Introduction to Software Engineering MCQs for ...

1. According to Sommerville, what does 'software' consist of? Understanding
   A. Only the executable program code
   B. Source code plus the compiler used to build it
   C. Computer programs plus associated documentation and configuration data
   D. Device drivers and the operating system

2. Which phrase is at the heart of the IEEE definition of software engineering? Understanding
   A. Writing code as quickly as possible to meet a deadline
   B. The mathematical theory of computation and algorithms
   C. Choosing and buying the hardware that software runs on
   D. A systematic, disciplined, quantifiable approach to the development, operation and maintenance of software

3. The term 'software engineering' was popularised, and the 'software crisis' widely discussed, at which event? Understanding
   A. The 2001 meeting that produced the Agile Manifesto
   B. Royce's 1970 paper on managing large software systems
   C. Boehm's 1988 paper introducing the spiral model
   D. The 1968 NATO Software Engineering Conference

4. Which was a typical symptom of the software crisis? Understanding
   A. Projects delivered late and over budget, with unreliable software that was hard to maintain
   B. Hardware becoming more expensive than the software running on it
   C. Too few programming languages being available to developers
   D. Software physically wearing out after a few years of use

5. Which list gives Sommerville's four essential attributes of good software? Understanding
   A. Portability, speed, low cost, attractive design
   B. Correctness, completeness, consistency, compactness
   C. Maintainability, dependability and security, efficiency, acceptability
   D. Modularity, low coupling, high cohesion, inheritance

6. For a generic software product, who owns the specification of what the software should do? Understanding
   A. A single customer who signs the development contract
   B. The developer, who decides on features and changes
   C. An independent national standards body
   D. The end users, through a formal vote on each release

7. Which of these is an example of a customised (bespoke) software product? Understanding
   A. A word processor sold to millions of users
   B. A mobile game published on an app store
   C. An air traffic control system built for one national aviation authority
   D. An off-the-shelf antivirus package

8. Which four activities does Sommerville say are fundamental to every software process? Understanding
    A. Communication, planning, modelling and construction
    B. Inception, elaboration, construction and transition
    C. Specification, development, validation and evolution
    D. Analysis, coding, debugging and release

9. What are the five framework activities in Pressman's generic process framework? Understanding
    A. Communication, planning, modelling, construction, deployment
    B. Specification, design, validation, evolution, retirement
    C. Inception, elaboration, construction, transition, production
    D. Requirements, design, coding, testing, maintenance

10. In Pressman's layered view of software engineering, which layer is described as the bedrock that supports the others? Understanding
    A. Tools
    B. Methods
    C. A quality focus
    D. Process

11. Which of these is an umbrella activity in Pressman's process framework? Understanding
    A. Construction
    B. Software configuration management
    C. Deployment
    D. Communication

12. Software does not wear out, yet it deteriorates over time. What is the main reason? Understanding
    A. Storage media slowly lose the bits of the program
    B. Compilers produce worse code as they age
    C. Processors execute old instructions more slowly
    D. Changes made during maintenance introduce new defects and erode the original design

13. Which statement is Brooks's law? Understanding
    A. Work expands to fill the time available for its completion
    B. Adding manpower to a late software project makes it later
    C. The number of transistors on a chip doubles about every two years
    D. Organisations design systems that mirror their own communication structures

14. Which principle of the ACM/IEEE Software Engineering Code of Ethics states that software engineers shall act consistently with the public interest? Understanding
    A. Public
    B. Client and Employer
    C. Product
    D. Self

15. In 'No Silver Bullet', which four properties did Brooks identify as essential difficulties of software? Understanding
    A. Cost, schedule, staffing and tooling
    B. Complexity, conformity, changeability and invisibility
    C. Syntax errors, logic errors, runtime errors and linker errors
    D. Coupling, cohesion, abstraction and encapsulation

16. How does Sommerville distinguish computer science from software engineering? Understanding
    A. Computer science covers theory and fundamentals; software engineering covers the practicalities of developing and delivering useful software
    B. They are two names for exactly the same discipline
    C. Computer science covers hardware, while software engineering covers only programming
    D. Computer science deals with project management, while software engineering deals with algorithms

17. A project is three weeks behind schedule. The manager says: 'No problem, we will just add more programmers and catch up.' Pressman classifies this belief as a: Application
    A. Customer myth
    B. Practitioner myth
    C. Proven scheduling principle
    D. Management myth

18. A client says: 'Here is a general statement of what we want. Start coding now and we will fill in the details later.' Which kind of software myth is this? Application
    A. Customer myth
    B. Management myth
    C. Practitioner myth
    D. Process myth

19. A developer says: 'Once the program runs and passes my demo, my job is done.' Which software myth does this reflect? Application
    A. Customer myth
    B. Management myth
    C. Practitioner myth
    D. Hardware myth

20. Using the pairwise formula n(n - 1)/2, how many communication paths exist in a team of 6 developers? Application
    A. 30
    B. 15
    C. 36
    D. 12

21. A team grows from 5 to 8 developers. By how many does the number of pairwise communication paths increase? Application
    A. 3
    B. 28
    C. 24
    D. 18

22. A university contracts a software house to build an exam-management system to its own specification. How is this product classified? Application
    A. Generic product
    B. Customised (bespoke) product
    C. Commercial off-the-shelf (COTS) product
    D. Shrink-wrapped product

23. The software controlling a car's anti-lock braking system belongs to which of Sommerville's application types? Application
    A. Batch processing system
    B. Interactive transaction-based application
    C. Embedded control system
    D. Data collection system

24. A bank's payroll program runs once a month, reading thousands of employee records and printing payslips without user interaction. Which application type is it? Application
    A. Batch processing system
    B. Interactive transaction-based application
    C. Embedded control system
    D. Entertainment system

25. Passengers search flights, choose seats and pay on an airline's website, and each request updates a central database. Which of Sommerville's application types is this? Application
    A. Batch processing system
    B. Interactive transaction-based application
    C. Embedded control system
    D. Modelling and simulation system

26. An engineer finds that a release could expose patients' records, but the employer wants to ship on time. Under the ACM/IEEE Code of Ethics, which principle takes priority? Application
    A. Client and Employer: act in the best interests of the employer
    B. Management: keep projects on schedule
    C. Public: act consistently with the public interest
    D. Colleagues: be fair to and supportive of colleagues

27. Users find a new hospital app confusing, say it does not fit their workflow, and refuse to use it. Which of Sommerville's essential attributes is most clearly lacking? Application
    A. Efficiency
    B. Maintainability
    C. Dependability and security
    D. Acceptability

28. In Pressman's framework, writing code and running the unit tests on it are both part of which activity? Application
    A. Modelling
    B. Construction
    C. Deployment
    D. Planning

29. A developer uses cryptic one-letter names everywhere, saying 'Only I will ever read this code.' Which of Hooker's seven principles (in Pressman) is violated? Application
    A. Be open to the future
    B. What you produce, others will consume
    C. Plan ahead for reuse
    D. Maintain the vision

30. A failure-rate chart for a product shows a spike after every new release, with the minimum failure rate creeping upward over the years. What does the chart describe? Application
    A. The hardware bathtub curve: wear-out at end of life
    B. The idealised software curve: failures fall to a flat line
    C. A sign the storage hardware is physically wearing out
    D. The actual software failure curve: deterioration caused by change

31. A startup building a small MVP says software engineering is pointless overhead because 'we're just writing code'. Which response best reflects Sommerville's view? Analysis
    A. Methods should suit the system type; even an MVP needs some specification, validation and evolution, with far less formality than a safety-critical system
    B. Software engineering is only for large projects, so the startup can skip it entirely
    C. The startup must follow a full waterfall process with complete documentation
    D. Software engineering is the same as programming, so nothing changes either way

32. A late project has 4 developers and 8 person-months of work left. The manager adds 4 new developers and expects it to finish in 1 month. What is the most likely outcome? Analysis
    A. It will finish in exactly 1 month, because 8 people x 1 month = 8 person-months
    B. It will finish in exactly 2 months, because new staff have no effect at all
    C. It will take longer than 1 month: newcomers need ramp-up, and communication paths rise from 6 to 28
    D. It will finish in under 1 month, because parallel work removes coordination costs

33. A manager suggests keeping spare copies of a module to swap in when it fails, as is done with hardware parts. Why will this not fix software failures? Analysis
    A. Software copies degrade during storage and become unreliable
    B. Swapping copies would work, but storing them is too expensive
    C. Every copy contains the same design or coding defect; software fails because of errors, not physical wear
    D. Only the original installed copy can run on the hardware

34. A developer accepts a contract to build a cryptographic payment system despite having no security experience, and tells the client they are an expert. Which professional responsibility from Sommerville is mainly breached? Analysis
    A. Confidentiality
    B. Intellectual property rights
    C. Computer misuse
    D. Competence

35. Why did the software crisis emerge in the 1960s? Analysis
    A. Hardware grew powerful enough for much larger, more complex systems, and informal development methods did not scale to them
    B. Programmers of the time were less capable than today's
    C. No high-level programming languages existed yet
    D. Hardware was too slow to run large programs at all

36. Product A is a mobile game sold to millions; Product B is a records system built for one hospital. A major requirements change is proposed for each. For which product does the customer decide whether the change is made? Analysis
    A. Product B, because a customised product's specification is owned by the customer
    B. Product A, because its many users own the specification
    C. Both, because users always own the specification
    D. Neither, because the developer always owns the specification

37. A team adopts a new IDE with better autocompletion. According to Brooks's 'No Silver Bullet', what should they expect? Analysis
    A. It removes essential complexity, so productivity will rise about tenfold
    B. It removes the conformity problem, so integration issues disappear
    C. It eliminates invisibility, so the design structure becomes obvious
    D. It reduces accidental difficulty, so it helps, but it will not give an order-of-magnitude productivity gain

38. After a release, a team cannot tell which version of a configuration file is running in production, and two developers overwrote each other's changes. Which umbrella activity has failed? Analysis
    A. Risk management
    B. Software configuration management
    C. Technical reviews
    D. Measurement

39. A team held one short call with a client, coded for a month, then learned at delivery that they had built the wrong product. Which of Pressman's framework activities were most neglected? Analysis
    A. Construction: the code was probably badly written
    B. Deployment: the delivery was poorly scheduled
    C. Planning alone: only the schedule estimate was wrong
    D. Communication and modelling: requirements were not properly elicited, and no analysis model was reviewed with the client

40. Option 1: one team of 9 developers. Option 2: three teams of 3, where the three team leads also talk to each other. How many pairwise communication paths does each option have? Analysis
    A. Option 1 has 36; option 2 has 12
    B. Option 1 has 81; option 2 has 27
    C. Option 1 has 36; option 2 has 9
    D. Option 1 has 18; option 2 has 6

41. Pressman describes software as having three parts. Which list matches his definition? Understanding
    A. Instructions (programs), data structures that let programs manipulate information, and descriptive information (documentation)
    B. Source code, compiled machine code and the compiler that produced it
    C. Programs, the hardware they run on and the operating system
    D. Executable instructions only; documentation is a separate product

42. What does the hardware 'bathtub curve' show? Understanding
    A. Software failure rate rising after every new release
    B. Hardware failure rate over time: high early failures, a long low period, then rising failures as components wear out
    C. The cost of fixing a defect in each successive development phase
    D. Team productivity falling as more developers are added

43. In Pressman's idealised failure curve for software, what happens after the early, undiscovered defects are corrected? Understanding
    A. The failure rate climbs steadily as the code ages
    B. The failure rate falls to exactly zero and stays there
    C. The failure rate flattens and stays level, because software is not affected by physical wear
    D. The failure rate follows the bathtub curve of the hardware it runs on

44. What does the phrase 'software is engineered, not manufactured' mean? Understanding
    A. Software must be built on a factory production line like hardware
    B. Software engineers do not need to follow a defined process
    C. Software quality is mainly decided during the copying of the final release
    D. Its costs are concentrated in development; producing extra copies costs almost nothing and does not add quality problems

45. Which four general issues does Sommerville say affect most types of software? Understanding
    A. Heterogeneity, business and social change, security and trust, and scale
    B. Complexity, conformity, changeability and invisibility
    C. Maintainability, dependability and security, efficiency and acceptability
    D. Specification, development, validation and evolution

46. Over the lifetime of a long-lived system, which cost does Sommerville say often exceeds the cost of developing it? Understanding
    A. The cost of the hardware it runs on
    B. The cost of changing the software after it goes into use
    C. The cost of writing the original requirements
    D. The cost of training the first group of users

47. Which of Hooker's seven principles (quoted by Pressman) says every design decision should be tested by asking whether it adds real value for users? Understanding
    A. Keep it simple, stupid
    B. Maintain the vision
    C. The reason it all exists
    D. Be open to the future

48. Which of Hooker's principles warns that without a clear, shared vision a system becomes a patchwork of incompatible designs? Understanding
    A. Think!
    B. Plan ahead for reuse
    C. What you produce, others will consume
    D. Maintain the vision

49. In Pressman's layered view of software engineering, which layer provides the technical 'how-to's' for building software, such as requirements analysis, design modelling and testing? Understanding
    A. Methods
    B. Tools
    C. Process
    D. Quality focus

50. Pressman bases the essence of software engineering practice on George Polya's problem-solving approach. What is the order of its four steps? Understanding
    A. Plan a solution, understand the problem, test the result, carry out the plan
    B. Understand the problem, plan a solution, carry out the plan, examine the result for accuracy
    C. Communicate, plan, model, construct
    D. Specify, develop, validate, evolve

51. How many principles does the ACM/IEEE Software Engineering Code of Ethics contain? Understanding
    A. Four
    B. Six
    C. Eight
    D. Ten

52. Which list gives Sommerville's four areas of professional responsibility for software engineers? Understanding
    A. Public, client and employer, product and judgment
    B. Honesty, punctuality, loyalty and teamwork
    C. Reliability, security, safety and availability
    D. Confidentiality, competence, intellectual property rights and computer misuse

53. Into which three groups does Pressman divide software myths? Understanding
    A. Management myths, customer myths and practitioner myths
    B. Hardware myths, software myths and network myths
    C. Requirement myths, design myths and testing myths
    D. Technical myths, legal myths and financial myths

54. Which of Pressman's software application domains covers compilers, editors, file-management utilities and operating-system components? Understanding
    A. Application software
    B. System software
    C. Engineering and scientific software
    D. Embedded software

55. Pressman notes that organisations keep old 'legacy' systems running for decades, despite poor design and documentation. What is the main reason? Understanding
    A. Legacy systems are always of higher quality than new systems
    B. Legacy code cannot legally be rewritten once it is in use
    C. They support core business functions and are indispensable, so replacing them is costly and risky
    D. Legacy systems never need to change, so they cost nothing to keep

56. According to Brooks in 'No Silver Bullet', which past breakthrough removed a large share of software's accidental difficulty? Understanding
    A. Detailed requirements documents, which eliminated complexity
    B. Formal specifications, which eliminated changeability
    C. Flowcharts, which eliminated invisibility
    D. High-level programming languages, which freed programmers from machine-level detail

57. Which of Brooks's essential difficulties means software has no natural geometric representation, so its structure cannot be captured in a single diagram? Understanding
    A. Invisibility
    B. Complexity
    C. Conformity
    D. Changeability

58. Under Sommerville's attribute 'dependability and security', good software should: Understanding
    A. Be easy to understand, change and extend as requirements evolve
    B. Not cause physical or economic damage when it fails, and stop malicious users accessing or damaging the system
    C. Use memory and processor cycles without waste
    D. Be understandable, usable and compatible with the users' other systems

59. Which umbrella activity in Pressman's framework defines criteria for reusing work products and sets up mechanisms to build reusable components? Understanding
    A. Software configuration management
    B. Work product preparation and production
    C. Reusability management
    D. Software quality assurance

60. What is a software process model, according to Sommerville? Understanding
    A. The source code of the tools used to build a system
    B. A UML class diagram of the system's architecture
    C. A legal contract between developer and customer
    D. A simplified representation of a software process, presented from a particular perspective

61. Which ACM/IEEE Code principle requires engineers to maintain integrity and independence in their professional judgment? Understanding
    A. Judgment
    B. Profession
    C. Self
    D. Product

62. Pressman says software plays a dual role. Which statement describes it? Understanding

A. It is a product when sold and a service only when it is free
B. It is a product that delivers computing potential, and a vehicle for delivering a product, by controlling computers, enabling communication and building other programs
C. It is hardware when embedded and software when installed on a PC
D. It is a document during design and a program after deployment

63. How does Sommerville relate system engineering to software engineering? Understanding

A. They are the same discipline under two different names
B. Software engineering is the broader field and includes hardware design
C. System engineering covers all aspects of computer-based systems, including hardware, software and process engineering; software engineering is part of it
D. System engineering deals only with operating systems

64. Why does Sommerville argue there is no single universal software engineering method? Understanding

A. Software engineering is too young a discipline to have any methods
B. Every company is legally required to invent its own method
C. Only the waterfall model is universal; the rest are optional
D. Different kinds of system need different techniques; a safety-critical controller and an online game have very different needs

65. A team has 10 developers, and every pair needs to communicate. How many communication paths are there? Application

A. 45
B. 100
C. 90
D. 20

66. Twelve developers are split into three teams of 4. Members talk only within their team, and the three team leads also talk to each other. How many communication paths are there? Application

A. 66
B. 21
C. 18
D. 24

67. Remaining work is estimated at 24 person-months, and a manager assigns 6 developers. What does the simple person-month calculation predict, and what is Brooks's objection? Application

A. 4 months, and Brooks says this is accurate for any software task
B. 144 months, because person-months must be multiplied by headcount
C. 4 months, but that holds only if the work can be split perfectly with no communication between people
D. 18 months, because six people subtract six months from the estimate

68. A manager says: 'We already have a book full of standards and procedures for building software. That gives my people everything they need.' Which kind of software myth is this? Application

A. Customer myth
B. Practitioner myth
C. Ethical myth
D. Management myth

69. A customer says: 'Our requirements will keep changing, but that's fine, because software is flexible and changes are easy to make.' Which kind of software myth is this? Application
    A. Customer myth
    B. Management myth
    C. Practitioner myth
    D. Hardware myth

70. A developer says: 'Until the program is running, I have no way of assessing its quality.' Which reality, according to Pressman, corrects this myth? Application
    A. Quality can only be measured by customers after release
    B. Technical reviews can assess quality from the start of the project and are one of the most effective quality filters
    C. Compiling the code without errors proves its quality
    D. Quality assessment belongs to the hardware team, not developers

71. A company director says: 'We've outsourced the system to a vendor, so we can relax and let them build it.' What does Pressman say about this belief? Application
    A. It is a customer myth, and outsourcing removes all project risk
    B. It is correct, because contracts transfer all responsibility to the vendor
    C. It is a management myth: an organisation that cannot manage software projects internally will struggle when it outsources them
    D. It is a practitioner myth about the vendor's programmers

72. A developer argues: 'Software engineering will make us write piles of unnecessary documents and slow us down.' Which response reflects Pressman? Application
    A. It is true; software engineering always trades speed for paperwork
    B. This is a customer myth, because customers write the documents
    C. It is true only for generic products, never customised ones
    D. This is a practitioner myth; software engineering aims at quality, and better quality reduces rework and so speeds up delivery

73. A photo-editing app is sold to anyone through an app store, and its company decides which features to add next. How is it classified? Application
    A. Generic product
    B. Customised (bespoke) product
    C. Embedded control system
    D. Batch processing system

74. A company buys an ERP system from a vendor and then spends months configuring its modules and workflows to match its own business processes. How does Sommerville describe this situation? Application
    A. It is a purely customised product written from scratch for the company
    B. It blurs the line between generic and customised products: a generic system is adapted to one customer's needs
    C. It is an embedded control system, because it is installed on site
    D. It stops being software and becomes a service contract

75. A national meteorological office runs software that represents the atmosphere mathematically to forecast tomorrow's weather. Which of Sommerville's application types is it? Application
    A. Data collection system
    B. Batch processing system
    C. Modelling and simulation system
    D. Entertainment system

76. Sensors along a river measure water levels and send the readings to a central flood-monitoring system. Which of Sommerville's application types describes the sensor software? Application
    A. Stand-alone application
    B. Modelling and simulation system
    C. Interactive transaction-based application
    D. Data collection system

77. A spreadsheet program installed on a laptop does everything locally and never needs a network connection. Which of Sommerville's application types is it? Application
    A. Stand-alone application
    B. Interactive transaction-based application
    C. Systems of systems
    D. Batch processing system

78. A payroll system must implement hundreds of tax rules exactly as the law states them, and match the file formats of three existing bank systems. Which of Brooks's essential difficulties does this illustrate? Application
    A. Invisibility
    B. Conformity
    C. Changeability
    D. Accidental complexity

79. Before any code is written, three engineers meet to examine the design document line by line and record every error they find. Which of Pressman's umbrella activities is this? Application
    A. Measurement
    B. Risk management
    C. Technical reviews
    D. Software project tracking and control

80. Every Friday a project lead compares actual progress with the schedule and reassigns work when a task slips. Which of Pressman's umbrella activities is this? Application
    A. Software quality assurance
    B. Reusability management
    C. Technical reviews
    D. Software project tracking and control

81. During an evaluation, a vendor offers a tester an expensive gift if their report recommends the vendor's tool. Which ACM/IEEE Code principle is most directly at stake? Application
    A. Judgment
    B. Self
    C. Colleagues
    D. Public

82. An engineer publishes a conference paper that presents a teammate's algorithm as their own work. Which ACM/IEEE Code principle is breached? Application
    A. Product
    B. Colleagues
    C. Client and Employer
    D. Management

83. An engineer copies a client's confidential customer database to a personal laptop to reuse in a side project. Which of Sommerville's professional responsibilities is breached? Application
    A. Competence
    B. Intellectual property rights
    C. Confidentiality
    D. Computer misuse

84. A team installs unlicensed copies of a commercial compiler on every developer's machine to save money. Which of Sommerville's professional responsibilities is breached? Application
    A. Confidentiality
    B. Competence
    C. Computer misuse
    D. Intellectual property rights

85. In Pressman's framework, estimating effort, drawing up a schedule and listing the risks to manage belong to which framework activity? Application
    A. Planning
    B. Communication
    C. Modelling
    D. Deployment

86. A team grows from 7 to 10 developers, all talking to each other. Headcount rises by about 43%. Which statement about communication paths is correct? Analysis
    A. Paths rise by about 43%, the same as headcount, because each person adds one path
    B. Paths rise from 21 to 45, about 114%, because paths grow roughly with the square of team size
    C. Paths rise from 49 to 100, about 104%, because paths equal n squared
    D. Paths rise from 21 to 30, about 43%, because three people add three paths each

87. A bank's app must run on Android, iOS and the web, and exchange data with a 30-year-old mainframe. Which of Sommerville's general issues does this illustrate most directly? Analysis
    A. Scale
    B. Business and social change
    C. Heterogeneity
    D. Security and trust

88. Twelve developers can be organised as (P) two teams of 6 with the two leads linked, or (Q) three teams of 4 with all three leads linked. Which statement is correct? Analysis
    A. P has 30 paths and Q has 18, because links between leads are not counted
    B. P has 32 paths and Q has 24, because each lead link is counted in both directions
    C. P and Q both have 66 paths, because all 12 developers are still on the project
    D. P has 31 paths and Q has 21, so Q needs less coordination, and both are far below the 66 of one team

89. A company plans one team of 15 developers. An alternative splits them into three teams of 5, with the three team leads also talking to each other. Which statement is correct? Analysis
    A. The single team has 105 paths and the split design has 33, so splitting removes 72 paths
    B. The single team has 225 paths and the split design has 75, so splitting removes 150 paths
    C. The single team has 105 paths and the split design has 30, so splitting removes 75 paths
    D. Both designs have 105 paths, because headcount is unchanged

90. A finance director says: 'Software doesn't wear out, so once it is delivered we don't need a maintenance budget.' What is the best critique? Analysis
    A. The director is right; delivered software never needs to change
    B. Software deteriorates as it is changed, and it must keep changing as the business and its environment change; lifetime change costs often exceed development costs
    C. Software wears out just like hardware, so worn modules must be replaced with spares
    D. Maintenance is only needed if the original developers made mistakes

91. A project manager tells the team to submit deliberately low cost estimates so the company wins a bid, planning to ask for more money later. Which ACM/IEEE Code principle is most directly violated by the manager? Analysis
    A. Self
    B. Colleagues
    C. Management
    D. Product

92. Which change reduces accidental difficulty rather than essential difficulty, in Brooks's sense? Analysis
    A. Asking regulators to simplify the tax rules a payroll system must follow
    B. Drawing more diagrams so that software stops being invisible
    C. Freezing requirements permanently so the software never needs to change
    D. Replacing manual, error-prone builds with an automated build and deployment pipeline

93. A company must choose between buying a generic CRM product and commissioning a custom one. Which comparison is accurate? Analysis
    A. The generic product is usually cheaper and faster to deploy, but the vendor controls its specification; the custom product fits the company's needs and its specification is the company's, at higher cost and time
    B. The company controls the generic product's specification, while the developer controls the custom one
    C. A custom product is always cheaper because there are no licence fees
    D. Generic products cannot be configured, so they never fit any business

94. System A crashes weekly and sometimes loses data. System B never fails, but each small change takes weeks because the code is tangled and undocumented. Which of Sommerville's attributes does each lack? Analysis
    A. A lacks efficiency; B lacks acceptability
    B. A lacks dependability and security; B lacks maintainability
    C. A lacks maintainability; B lacks dependability and security
    D. Both lack acceptability, since users dislike them

95. A team records no defect counts, review results or effort data, so after two years nobody can say whether quality is improving. Which umbrella activity is missing, and what is its consequence? Analysis
    A. Software configuration management; versions of files are being lost
    B. Risk management; the team cannot predict future risks
    C. Measurement; without data the team cannot assess or improve its process and product
    D. Reusability management; components are being rewritten

96. Which of these statements is a reality, not one of Pressman's software myths? Analysis
    A. Once we write the program and get it to work, our job is done
    B. A general statement of objectives is enough to begin writing programs
    C. If we fall behind schedule, we can add more programmers and catch up
    D. A working program is only one part of a software configuration, which also includes models, documents and plans

97. In which situation is adding people least likely to suffer from Brooks's law? Analysis
    A. Early in a project, adding staff to enter 10,000 independent records, a task that needs no coordination
    B. One week before release, adding newcomers to fix bugs in a tightly coupled core module
    C. Mid-integration, adding developers who must learn a complex, undocumented architecture
    D. Late in a project, adding people who must attend every design meeting to stay in sync

98. A developer writes a dense one-line function using clever tricks 'to show skill', and teammates cannot follow it. Which two of Hooker's principles does this violate most directly? Analysis
    A. Be open to the future, and plan ahead for reuse
    B. Keep it simple, stupid, and what you produce, others will consume
    C. The reason it all exists, and maintain the vision
    D. Think!, and plan ahead for reuse

99. Comparing an online multiplayer game with a hospital infusion-pump controller, which statement best reflects Sommerville's view of methods? Analysis
    A. Both must use exactly the same documented waterfall method
    B. The game needs formal safety proofs, while the pump can be prototyped freely
    C. The game suits rapid, iterative prototyping with player feedback, while the pump needs rigorous specification, safety analysis and documentation
    D. Neither needs software engineering, because both are small programs

100. Which statement correctly relates Pressman's construction activity to Sommerville's fundamental activities? Analysis
    A. Construction is the same as Sommerville's specification
    B. Construction matches Sommerville's evolution, because both happen after release
    C. Construction has no equivalent, since Sommerville does not include testing
    D. Construction combines code generation and testing, so it overlaps both Sommerville's development and validation

Software Process Models MCQs for the NSCT

101. What is the correct phase order of the waterfall model as Sommerville presents it? Understanding
    A. System and software design, requirements definition, implementation and unit testing, integration and system testing, operation and maintenance
    B. Requirements definition, implementation and unit testing, system and software design, integration and system testing, operation and maintenance
    C. Requirements definition, system and software design, implementation and unit testing, integration and system testing, operation and maintenance
    D. Requirements definition, system and software design, integration and system testing, implementation and unit testing, operation and maintenance

102. The waterfall model is most appropriate when: Understanding
    A. Requirements are well understood and unlikely to change much during development
    B. Requirements are expected to change every few weeks
    C. The project is exploratory research with an unknown outcome
    D. A startup needs rapid user feedback on early versions

103. Who proposed the spiral model of software development? Understanding
    A. Winston Royce
    B. Kent Beck
    C. Ken Schwaber
    D. Barry Boehm

104. What most distinguishes the spiral model from other process models? Understanding

A. Strictly sequential phases with no iteration
B. Explicit risk assessment and reduction in every loop
C. Fixed-length sprints of two weeks
D. Testing that happens only at the end of the project

105. In Sommerville's description of Boehm's spiral, what are the four sectors of each loop? Understanding

A. Requirements; design; coding; testing
B. Objective setting; risk assessment and reduction; development and validation; planning
C. Inception; elaboration; construction; transition
D. Communication; modelling; construction; deployment

106. In the V-model, what is paired with each development phase on the left side of the V? Understanding

A. A prototype that is shown to the customer
B. A corresponding testing phase on the right side
C. A risk-analysis step, as in the spiral
D. A sprint review with stakeholders

107. In the V-model, unit testing checks the work products of which phase? Understanding

A. Detailed (module or component) design
B. User requirements
C. Architectural (high-level) design
D. System requirements specification

108. Acceptance testing in the V-model corresponds to which development phase? Understanding

A. User requirements analysis
B. Detailed module design
C. Coding
D. Architectural design

109. Which description fits incremental development? Understanding

A. The whole system is specified, built and tested in one pass
B. A disposable model is built and then discarded
C. The system is built as a series of versions, each adding functionality to the previous one
D. Development follows a loop driven by risk analysis

110. What is a throwaway prototype? Understanding

A. An early version that is refined step by step into the final system
B. A quick model built to clarify or validate requirements, then discarded before the real system is built
C. The first increment delivered to the customer
D. A prototype kept as a reference copy of production code

111. Boehm summarised verification as which question? Understanding

A. Are we building the right product?
B. Is the product finished on schedule?
C. Are we building the product right?
D. Does the customer accept the price?

112. What are the four phases of the Rational Unified Process (RUP), in order? Understanding

A. Elaboration, inception, construction, transition
B. Inception, elaboration, construction, transition
C. Inception, construction, elaboration, deployment
D. Planning, analysis, construction, transition

113. Which are Sommerville's three general process models? Understanding

A. Scrum, Extreme Programming and Kanban
B. Waterfall, incremental development, and integration and configuration
C. Spiral, V-model and RAD
D. Prototyping, RUP and Cleanroom

114. Which statement about Winston Royce's 1970 paper is accurate? Understanding

A. He described the sequential phases but warned the simple version was risky, recommending feedback between phases and building a pilot first
B. He endorsed a strictly sequential process with no feedback at all
C. He named it the 'waterfall model' in the title of the paper
D. He proposed the risk-driven spiral model

115. According to Sommerville, what is a key problem with incremental development? Understanding

A. Customers see no working software until the end of the project
B. It cannot accommodate changes to requirements
C. The process is less visible, and the system's structure tends to degrade as new increments are added
D. All requirements must be frozen before any coding starts

116. Why is the spiral model often called a meta-model? Understanding

A. It is the only model that can be used for every project unchanged
B. It describes the model of the project team rather than the product
C. It is a model of how other models were historically invented
D. Each loop can use whatever process model fits the dominant risk, such as prototyping or waterfall-style development

117. An exam-board system has requirements fixed by regulation, a fixed-price contract and a stable, experienced team. Which process model fits best? Application

A. Waterfall
B. Throwaway prototyping
C. Spiral
D. Evolutionary prototyping

118. A medical infusion pump must be certified, and every requirement and design element needs a matching, documented test plan. Which model fits best? Application

A. V-model
B. Throwaway prototyping
C. Extreme Programming
D. Spiral with no documentation

119. A large defence project uses novel technology, has a large budget and faces many unknown technical risks. Which model fits best? Application

A. Waterfall
B. Spiral
C. V-model
D. Code-and-fix

120. A client cannot describe what the user interface should look like and wants to see screens before deciding. Which approach fits best? Application

A. Waterfall
B. V-model
C. Big-bang integration
D. Prototyping

121. An e-commerce startup wants checkout live in two months, then wishlists, then reviews, releasing each to real users as it is ready. Which model fits best? Application

    A. Waterfall
    B. V-model
    C. Incremental
    D. Throwaway prototyping

122. A waterfall project plans requirements (2 months), design (3), implementation and unit testing (4), and integration and system testing (2). When does the customer first receive working software? Application

    A. End of month 9
    B. End of month 4
    C. End of month 2
    D. End of month 11

123. In a spiral loop, three risks are assessed. A: probability 0.2, loss Rs 500,000. B: 0.5, Rs 150,000. C: 0.1, Rs 900,000. Using risk exposure = probability x loss, which should be tackled first? Application

    A. Risk B
    B. Risk C
    C. Risk A
    D. All three equally, because they are in the same loop

124. A test checks that the payments module and the orders module exchange data correctly through their interface. In the V-model this level is paired with: Application

    A. User requirements
    B. Detailed module design
    C. Coding
    D. Architectural (high-level) design

125. At the end of a spiral loop the team reviews progress and decides whether to commit to another loop. Which sector is this? Application

    A. Planning
    B. Objective setting
    C. Risk assessment and reduction
    D. Development and validation

126. Your client warns that business rules will change often during the project. Which model should you avoid? Application

    A. Incremental
    B. Evolutionary prototyping
    C. Spiral
    D. Waterfall

127. In a RUP project, the team builds an executable architecture baseline and tackles the major technical risks. Which phase are they in? Application

    A. Elaboration
    B. Inception
    C. Construction
    D. Transition

128. A company builds an HR system mainly by selecting existing packages and web services, configuring them and integrating them. Which of Sommerville's general models is this? Application
    A. Waterfall
    B. Incremental development
    C. Integration and configuration
    D. Spiral

129. Increment 1 has 20 person-weeks. Features (priority, effort in person-weeks): Login (must, 5), Payment (must, 10), Reports (should, 8), Chat (could, 6), Search (must, 4). Using priority-first incremental delivery, which set goes in increment 1? Application
    A. Login, Payment and Reports
    B. Payment, Reports and Search
    C. Login, Reports, Chat and Search
    D. Login, Payment and Search

130. A team mocks up screens in a design tool to confirm requirements with users, then writes the production system from scratch with a proper architecture. What kind of prototyping is this? Application
    A. Evolutionary prototyping
    B. Incremental delivery
    C. Throwaway prototyping
    D. Operational prototyping of the final code

131. A waterfall project first showed users the system in month 14, and they rejected 30% of the features. What is the root cause, and what would have helped most? Analysis
    A. The coding was poor; better unit testing would have prevented the rejections
    B. The team was too small; adding developers would have fixed it
    C. The V-model's acceptance tests were missing; adding them at month 14 would have helped
    D. Waterfall delays user feedback until late; an incremental or prototyping approach would have exposed misunderstandings early

132. A team chooses the full spiral model for a three-week internal script with clear requirements. What is the best critique? Analysis
    A. The spiral's risk-analysis overhead and need for risk expertise are not justified for a small, low-risk project
    B. The spiral model cannot be used for internal software
    C. The spiral model forbids prototyping, which the script needs
    D. The spiral model always takes at least a year

133. How does the V-model differ from the classic waterfall? Analysis
    A. The V-model is iterative and delivers working increments every month
    B. The V-model removes the need for unit testing
    C. Each test level is planned alongside its matching development phase, so test design starts early, though execution is still sequential
    D. The V-model executes all tests before any code is written

134. Team A ships login in release 1, payments in release 2 and reports in release 3. Team B first ships a rough version of all three features, then improves each in later releases. Which description is correct? Analysis
    A. Team A is working iteratively; Team B is working incrementally
    B. Team A is working incrementally; Team B is working iteratively
    C. Both teams are following the waterfall model
    D. Both teams are following the V-model

135. Mitigation X costs Rs 20,000 and cuts a risk's exposure from Rs 100,000 to Rs 20,000. Mitigation Y costs Rs 5,000 and cuts another risk's exposure from Rs 75,000 to Rs 50,000. Using risk reduction leverage = (RE before - RE after) / cost, which is better value? Analysis
    A. Y, with leverage 5 versus X's 4
    B. X, with leverage 4 versus Y's 5
    C. X, because it removes more exposure in absolute terms, so leverage does not matter
    D. Both are equal, with leverage 4

136. A regulated banking core system will be built by 200 developers in three countries, and the regulator demands full upfront specification documents. Which approach is best justified? Analysis
    A. Pure Scrum with no upfront documentation
    B. Code-and-fix to reach the market quickly
    C. Throwaway prototyping as the only process
    D. A plan-driven model, such as waterfall or the V-model, possibly with some agile practices inside teams

137. Stakeholders see a quick prototype built with shortcuts and demand that it be shipped as the product next week. What is the best response? Analysis
    A. Ship it as is, because the customer is always right
    B. Add more features to the prototype so it looks complete
    C. Abandon prototyping and switch to the waterfall model
    D. Explain it was built to clarify requirements and rebuild the product properly, having agreed this purpose at the start

138. After eight increments, a system's code has become tangled and every new feature takes longer. What is the most appropriate response? Analysis
    A. Switch to waterfall for the remaining work
    B. Plan time for refactoring to restore structure, since frequent change in incremental development degrades it
    C. Stop adding features and freeze the system for good
    D. Hire more developers to work faster on the tangled code

139. Acceptance testing shows that the system does exactly what its specification says, but users say it does not meet their needs. What kind of failure is this, and where did the defect originate? Analysis
    A. A verification failure, originating in coding
    B. A validation failure, originating in the requirements phase
    C. A verification failure, originating in unit testing
    D. A validation failure, originating in integration testing

140. Which pairing of process model and weakness is correct? Analysis
    A. Spiral: performs no risk analysis, so risks surface late
    B. Incremental: customers see nothing until the final release
    C. V-model: inflexible to requirement changes, because it inherits the waterfall's sequence
    D. Prototyping: cannot help clarify vague requirements

141. What is the 'code-and-fix' (build-and-fix) approach to software development? Understanding
    A. Writing code straight away and repairing problems as they appear, with no formal specification or design
    B. Building a throwaway prototype and then rewriting it with a proper design
    C. Planning test cases for every phase before writing any code
    D. Delivering the system in small, prioritised increments

142. Sommerville describes a four-stage process for developing a prototype. What is the correct order? Understanding

A. Develop the prototype, evaluate it, define its functionality, then set objectives
B. Establish prototype objectives, define prototype functionality, develop the prototype, evaluate the prototype
C. Requirements, design, implementation, testing
D. Inception, elaboration, construction, transition

143. What does the radial dimension (distance from the centre) represent in Boehm's spiral diagram? Understanding

A. The number of developers on the project
B. The number of defects found in testing
C. The cumulative cost incurred so far
D. The level of customer satisfaction

144. Pressman describes four types of process flow. In which one are one or more activities carried out at the same time as others? Understanding

A. Linear process flow
B. Iterative process flow
C. Evolutionary process flow
D. Parallel process flow

145. How does Pressman's concurrent development model represent software engineering activities? Understanding

A. Each activity exists in one of several states, such as under development, awaiting changes or done, and events trigger transitions between states
B. Each activity runs once in strict sequence and is then closed for good
C. All activities are skipped except construction, which runs continuously
D. Each activity is assigned to a separate company working in parallel

146. Sommerville describes the Rational Unified Process from three perspectives. Which are they? Understanding

A. Inception, elaboration and construction
B. Dynamic (phases over time), static (process activities or workflows) and practice (recommended good practices)
C. Technical, financial and legal
D. Requirements, design and testing

147. Which RUP phase moves the system from the development community to the user community and makes it work in the real environment? Understanding

A. Inception
B. Elaboration
C. Transition
D. Construction

148. Which of these is one of the good practices recommended by the Rational Unified Process? Understanding

A. Freeze all requirements at the end of inception
B. Avoid iteration so that each phase finishes only once
C. Leave quality checks until the final transition phase
D. Visually model the software using UML

149. Sommerville lists three types of software component commonly reused in integration-and-configuration development. Which list matches? Understanding
    A. Stand-alone application systems configured for an environment, collections of objects such as frameworks or packages, and web services
    B. Waterfall, spiral and V-model processes
    C. Unit tests, integration tests and acceptance tests
    D. Requirements documents, design documents and user manuals

150. According to Sommerville, what are the main trade-offs of the integration-and-configuration (reuse-based) approach? Understanding
    A. It is slower and more expensive than writing everything from scratch
    B. It cuts cost, risk and delivery time, but requirements may have to be compromised and the organisation loses control over how reused components evolve
    C. It guarantees that every requirement is met exactly as written
    D. It removes the need for any testing or integration work

151. Sommerville contrasts change anticipation with change tolerance. What does change tolerance mean? Understanding
    A. Predicting every possible change before development starts and designing for it
    B. Refusing all change requests once the requirements are signed off
    C. The process is designed so that changes can be accommodated at relatively low cost, typically through incremental development
    D. Tolerating defects in released software instead of fixing them

152. What name does Pressman also give the waterfall model? Understanding
    A. The spiral meta-model
    B. The concurrent model
    C. The unified process
    D. The classic life cycle

153. Pressman notes that linear, waterfall-style projects can fall into 'blocking states'. What does this mean? Understanding
    A. Some team members must wait for others to finish dependent tasks before they can start their own
    B. The customer blocks all requirement changes after sign-off
    C. The compiler blocks deployment until every test passes
    D. Risk analysis blocks the project from continuing to the next loop

154. In Sommerville's description of software validation, what are the three stages of testing? Understanding
    A. Alpha testing, beta testing and gamma testing
    B. Component testing, system testing and customer (acceptance) testing
    C. Requirements testing, design testing and documentation testing
    D. Black-box testing, white-box testing and grey-box testing

155. Which list gives the design activities Sommerville describes in the design process? Understanding
    A. Inception, elaboration, construction and transition
    B. Communication, planning, modelling and deployment
    C. Architectural design, database design, interface design, and component selection and design
    D. Component testing, system testing and customer testing

156. What are the framework activities of Humphrey's Personal Software Process (PSP), as Pressman presents them? Understanding
    A. Inception, elaboration, construction, transition and production
    B. Communication, planning, modelling, construction and deployment
    C. Objective setting, risk assessment, development and planning
    D. Planning, high-level design, high-level design review, development and postmortem

157. What are the five maturity levels of the staged CMMI model, from lowest to highest? Understanding
    A. Initial, managed, defined, quantitatively managed, optimising
    B. Initial, defined, managed, optimising, quantitatively managed
    C. Inception, elaboration, construction, transition, production
    D. Ad hoc, repeatable, iterative, incremental, agile

158. What is the key idea of the formal methods model (for example, cleanroom software engineering) described by Pressman? Understanding
    A. Code is written first and documented formally afterwards
    B. A mathematical specification is developed and verified, which removes ambiguity and allows defects to be found by analysis rather than testing alone
    C. Formal meetings replace all technical documents
    D. Customers write the code using a formal template

159. What problem does aspect-oriented software development (AOSD) address? Understanding
    A. Projects whose only risk is an uncertain user interface
    B. Teams that cannot agree on a release date
    C. Crosscutting concerns, such as security, logging or transaction handling, that span many components
    D. Hardware that must be designed in parallel with software

160. In which kind of system does Sommerville say the waterfall model is still useful? Understanding
    A. Consumer mobile apps whose features change weekly
    B. Start-up products whose requirements are still being discovered
    C. Small internal scripts that a single developer writes in a day
    D. Embedded systems, where software must interface with inflexible hardware that cannot be changed late

161. According to Sommerville, what is a benefit of incremental development compared with the waterfall? Understanding
    A. The cost of accommodating changing customer requirements is reduced, because less analysis and documentation has to be redone
    B. The process is more visible to managers, because every increment produces full documentation
    C. The system structure always improves as more increments are added
    D. Customers do not need to be involved until the final release

162. Why is identifying common facilities hard in incremental delivery, according to Sommerville? Understanding
    A. Increments cannot share code by law
    B. Many increments need the same basic facilities, but requirements are not defined in detail until each increment is built, so shared needs are hard to spot early
    C. Each increment must use a different programming language
    D. Common facilities are always provided by the operating system

163. In Pressman's presentation of the Unified Process, which fifth phase follows transition? Understanding

A. Maintenance
B. Retirement
C. Production
D. Elaboration

164. Which recommendation did Winston Royce make in his 1970 paper, often summarised as 'do it twice'? Understanding

A. Run the whole waterfall twice for every project, with no changes
B. Write every program in two different languages for safety
C. Test each module twice, once by the developer and once by the customer
D. Build a preliminary pilot version of the system first, so that the final version is informed by what was learned

165. A waterfall plan has requirements (2 months), design (3), implementation (5) and testing (2). At the end of testing a requirements defect is found, forcing 1 month of requirements rework, 1.5 months of redesign, 2 months of recoding and a full 2-month retest. When is the system now delivered? Application

A. End of month 18.5
B. End of month 12
C. End of month 16.5
D. End of month 24

166. Increment 1 of a system takes 4 months, and each later increment takes 2.5 months. When does the customer receive increment 3? Application

A. End of month 7.5
B. End of month 9
C. End of month 12
D. End of month 6.5

167. In one spiral loop, three risks are identified: R1 (probability 0.3, loss Rs 400,000), R2 (0.1, Rs 1,000,000) and R3 (0.6, Rs 150,000). What is the total risk exposure for the loop? Application

A. Rs 1,550,000
B. Rs 103,333
C. Rs 310,000
D. Rs 120,000

168. A mitigation costs Rs 60,000 and cuts a risk's exposure from Rs 150,000 to Rs 100,000. Using risk reduction leverage = (exposure before - exposure after) / cost, should the team adopt it? Application

A. Yes; leverage is 2.5, so the mitigation is highly cost-effective
B. Yes; leverage is 1.2, so it is just worth it
C. No; leverage is negative because the exposure is still above zero
D. No; leverage is about 0.83, below 1, so the mitigation costs more than the risk it removes

169. At the start of a spiral loop, earlier loops have resolved the user-interface and performance risks. The main remaining risk is integrating the sub-systems. According to Sommerville, which development model suits this loop? Application

A. A waterfall-style model
B. Throwaway prototyping
C. Formal transformations
D. Code-and-fix

170. A tester checks that the complete, integrated system meets the response-time, security and functional requirements written in the system specification. In the V-model, which level is this, and what is it planned from? Application

A. Unit testing, planned from the detailed design
B. System testing, planned from the system requirements specification
C. Integration testing, planned from the architectural design
D. Acceptance testing, planned from the module code

171. In a V-model project, when should the acceptance test plan be written? Application

A. After coding is finished, when the system can be run
B. During unit testing, alongside the unit test cases
C. During user requirements analysis, before design begins
D. After deployment, based on users' complaints

172. In a project following Pressman's concurrent model, the modelling activity is in the 'under development' state when the customer says the requirements must change. Which state does modelling move to? Application

A. Baselined
B. Done
C. Inactive
D. Awaiting changes

173. In a RUP project, the team writes the business case, identifies the external actors that interact with the system and makes an initial estimate of cost and schedule. Which phase is this? Application

A. Inception
B. Elaboration
C. Construction
D. Transition

174. A RUP team is running beta tests at customer sites, training users and converting data from the old system. Which phase are they in? Application

A. Construction
B. Transition
C. Inception
D. Elaboration

175. A small company starts coding a stock-control program the day it is requested, with no written requirements or design, and patches it every time users report a fault. Which approach is this? Application

A. Spiral model
B. V-model
C. Code-and-fix
D. Throwaway prototyping

176. A start-up builds a rough version of its app with real users, refines it every week from their feedback, and the refined code becomes the product. Which approach is this? Application

A. Throwaway prototyping
B. V-model
C. Classic waterfall
D. Evolutionary prototyping

177. Without a UI prototype, there is a 40% chance of user-interface rework costing Rs 1,000,000. A prototype costs Rs 150,000 and cuts the chance to 10%. What is the expected saving from prototyping? Application

A. Rs 150,000
B. Rs 300,000
C. Rs 250,000
D. Rs 400,000

178. Increment 1 has 18 person-weeks. Features (priority, effort): A (must, 6), B (must, 8), C (should, 5), D (could, 3), E (should, 4). Take features in priority order, then in listed order, skipping any that do not fit. Which features are in increment 1? Application

A. A, B and C
B. A, B and E
C. A, B, C and D
D. A, B and D

179. After evaluating three off-the-shelf HR packages, a team rewrites some of its original requirements so that they match what the chosen package can do. In Sommerville's integration-and-configuration process, which stage is this? Application

A. Software discovery and evaluation
B. Application system configuration
C. Requirements refinement
D. Component adaptation and integration

180. An engineer records the time and defects for each task, then at the end of the project compares the actual figures with the plan to improve future estimates. Which PSP activity is the comparison? Application

A. High-level design review
B. Planning
C. Development
D. Postmortem

181. An organisation has documented, standard processes that every project tailors and follows. Which staged CMMI level does this describe? Application

A. Level 3, defined
B. Level 1, initial
C. Level 2, managed
D. Level 5, optimising

182. One sub-team designs the reporting screens while another sub-team codes the order-processing component, whose design is already finished. Which of Pressman's process flows is this? Application

A. Linear process flow
B. Parallel process flow
C. Iterative process flow
D. Evolutionary process flow

183. Four increments are delivered at the ends of months 3, 6, 9 and 12, each producing Rs 100,000 a month in benefit once in use. A waterfall would deliver everything at month 12. How much benefit has the incremental approach produced by the end of month 12? Application

A. Rs 1,200,000
B. Rs 400,000
C. Rs 1,800,000
D. Rs 3,600,000

184. A banking firm buys a commercial insurance package, and three years later the vendor drops features the bank relies on. Which disadvantage of integration-and-configuration does this illustrate? Application

A. Higher development cost than writing the system from scratch
B. Slower delivery than a waterfall project
C. Inability to configure the package at all
D. Loss of control over the evolution of reused system elements

185. A spiral loop finds a 25% chance that a Rs 2,000,000 module will fail. A Rs 80,000 prototype would cut that chance to 5%. What is the expected cost with and without the prototype? Application

A. Rs 180,000 with the prototype, Rs 500,000 without
B. Rs 100,000 with the prototype, Rs 500,000 without
C. Rs 180,000 with the prototype, Rs 2,000,000 without
D. Rs 580,000 with the prototype, Rs 500,000 without

186. Which situation does Sommerville identify as particularly hard for incremental delivery? Analysis

A. A new consumer app whose features can be released one at a time
B. Replacing an existing system, because users will not switch until the new one does at least everything the old one did
C. A project whose customer is happy to try early partial versions
D. A system whose highest-priority features can be identified early

187. A government tender requires the complete system specification to be part of the fixed contract before any work starts. Which approach conflicts most with this procurement model? Analysis

A. The waterfall model
B. The V-model
C. Incremental delivery, where the specification is developed alongside the software
D. A plan-driven process with up-front specification

188. Which statement correctly contrasts the spiral model with incremental development? Analysis

A. The spiral delivers features to users in each loop, while incremental development never delivers early
B. Incremental development requires formal risk analysis, while the spiral avoids it
C. They are identical; 'spiral' is just a diagram of incremental development
D. The spiral is driven by risk analysis in every loop, while increments are usually chosen by feature priority

189. Which pairing of process model and its main driver is correct? Analysis

A. Spiral: risk; V-model: test planning matched to each phase; incremental: feature priority
B. Spiral: documentation; V-model: risk; incremental: test planning
C. Spiral: feature priority; V-model: user feedback; incremental: risk
D. Spiral: test planning; V-model: feature priority; incremental: documentation

190. A team plans to evolve a quick UI prototype into a medical device system expected to run for 20 years. What is the strongest objection, based on Sommerville? Analysis

A. Prototypes are always slower to build than the final system
B. Throwaway-style prototypes cannot easily be tuned for non-functional needs such as safety and reliability, are poorly documented and degrade in structure, so they make a weak base
C. Medical devices cannot have user interfaces
D. Regulators require prototypes to be written in a different language

191. Which statement about Pressman's concurrent development model is correct? Analysis

A. Activities run strictly one after another, like the waterfall
B. It can be used only for embedded systems
C. All activities exist at the same time in different states, so the model gives an accurate picture of the project's current state
D. It eliminates the need for any communication activity

192. A manager claims that in RUP, coding happens only in the construction phase and requirements work happens only in inception. What is the best correction? Analysis

A. The claim is correct; each workflow belongs to exactly one phase
B. Coding happens only in transition, after users approve the design
C. RUP has no requirements workflow; requirements are handled by agile methods
D. RUP's workflows are not tied to phases: requirements, implementation and test work can occur in every phase, at different intensities

193. After two spiral loops (Rs 170,000 spent), analysis shows a 60% chance the core algorithm cannot meet requirements, which would waste the remaining Rs 800,000 budget. A third loop could build a Rs 60,000 prototype to test the algorithm. What does spiral thinking recommend? Analysis

A. Run the prototype loop: it costs Rs 60,000 against an expected loss of Rs 480,000, and the project can be stopped if it fails
B. Proceed straight to full development, since Rs 170,000 is already spent
C. Cancel immediately, since a 60% risk means the project will fail
D. Skip risk analysis and switch to the waterfall to finish faster

194. System testing reveals that the billing sub-system sends amounts in paisa while the payments sub-system expects rupees. In the V-model, which test level should have caught this, and which phase produced the defect? Analysis

A. Unit testing should have caught it; the defect came from coding
B. Integration testing should have caught it; the defect came from architectural (interface) design
C. Acceptance testing should have caught it; the defect came from user requirements
D. System testing caught it on time; no earlier level is responsible

195. Which process-model recommendation is poorly justified? Analysis

A. Use the V-model for avionics software that needs documented traceability from requirements to tests
B. Use incremental delivery for an e-commerce site that wants to launch core checkout early
C. Use the full spiral for a small, well-understood data-entry app because spiral reduces risk
D. Use integration and configuration for a payroll system when a suitable package exists

196. Sommerville argues that plan-driven and agile approaches are not a strict either/or choice. Which statement best reflects this? Analysis

A. Every project must choose pure waterfall or pure agile
B. Plan-driven processes cannot include any iteration
C. Agile processes forbid any planning or documentation
D. Most practical processes combine elements of both, with the balance depending on the system, the team and the organisation

197. To avoid costly iterations, a waterfall team freezes its requirements early, even though users are still unsure what they need. What is the most likely consequence, according to Sommerville? Analysis

A. The delivered system may not do what users want, and later problems are worked around in the code, making it poorly structured
B. The system will be delivered early and exactly match users' needs
C. Testing becomes unnecessary because requirements are fixed
D. Users will lose the ability to report defects

198. Which statement about the V-model and the waterfall is correct? Analysis

A. The V-model removes the waterfall's sequence and runs all phases at once
B. The V-model keeps the waterfall's sequential flow but plans a test level for each development phase, so it is just as hard to change late
C. The V-model is risk-driven, like the spiral
D. The V-model delivers working increments to users after each phase

199. Comparing reuse with building from scratch for a hospital records system, which comparison is most accurate? Analysis

A. Building from scratch is always cheaper because there are no licence fees
B. Reuse guarantees that every original requirement is met
C. Reuse can deliver sooner and with less risk, but the hospital may have to accept a package's way of working instead of every original requirement
D. Building from scratch removes the need for testing because the team controls the code

200. A spiral loop lists three risks: A (0.5 x Rs 200,000), B (0.2 x Rs 600,000) and C (0.05 x Rs 3,000,000). A manager ranks them by probability alone. Which statement is correct? Analysis

A. By exposure the order is A, B, C, the same as by probability
B. By exposure the order is C, A, B, because losses alone decide
C. All three have the same exposure, so the order does not matter
D. By exposure the order is C (150,000), B (120,000), A (100,000), the reverse of the probability ranking

Agile Software Development MCQs for the NSCT

201. Complete the Agile Manifesto value: 'Working software over ...' Understanding

A. following a plan
B. contract negotiation
C. comprehensive documentation
D. processes and tools

202. In which year was the Manifesto for Agile Software Development written? Understanding

A. 1995
B. 1999
C. 2001
D. 2010

203. According to the 2020 Scrum Guide, a Scrum Team consists of: Understanding

A. One Scrum Master, one Product Owner and Developers
B. One project manager, one Scrum Master and a Development Team
C. One Product Owner, several Scrum Masters and testers
D. One team lead, one architect and Developers

204. In the 2020 Scrum Guide, what are the people who create the usable Increment each Sprint called? Understanding

A. The Development Team
B. Developers
C. Engineers
D. The Delivery Team

205. What is the maximum length of a Sprint in the Scrum Guide? Understanding

A. One month
B. Two weeks
C. Six weeks
D. Three months

206. What is the timebox of the Daily Scrum? Understanding

A. 30 minutes
B. 5 minutes
C. 1 hour
D. 15 minutes

207. Who is accountable for ordering the Product Backlog? Understanding

A. The Scrum Master
B. The Developers
C. The Product Owner
D. The stakeholders

208. Which pairing of Scrum artifacts with their commitments matches the 2020 Scrum Guide? Understanding

A. Product Backlog: Sprint Goal; Sprint Backlog: Product Goal; Increment: Definition of Done
B. Product Backlog: Definition of Done; Sprint Backlog: Sprint Goal; Increment: Product Goal
C. Product Backlog: Product Goal; Sprint Backlog: Definition of Done; Increment: Sprint Goal
D. Product Backlog: Product Goal; Sprint Backlog: Sprint Goal; Increment: Definition of Done

209. What are the three pillars of Scrum's empiricism? Understanding

A. Transparency, inspection and adaptation
B. Commitment, focus and courage
C. Planning, execution and review
D. Visualise, limit and manage

210. Which list gives the five Scrum values? Understanding

A. Communication, simplicity, feedback, courage and respect
B. Transparency, inspection, adaptation, focus and respect
C. Individuals, interactions, collaboration, change and software
D. Commitment, focus, openness, respect and courage

211. Which are core practices of the Kanban method? Understanding

A. Fixed-length sprints, sprint reviews and retrospectives
B. Pair programming, test-first development and refactoring
C. Visualise the workflow, limit work in progress and manage flow
D. Assigning a Product Owner and a Scrum Master

212. In Extreme Programming, what is the practice of two developers working together at one workstation called? Understanding

A. Mob review
B. Collective ownership
C. Spiking
D. Pair programming

213. What are the five values of Extreme Programming (second edition)? Understanding

A. Communication, simplicity, feedback, courage and respect
B. Commitment, focus, openness, respect and courage
C. Transparency, inspection, adaptation, simplicity and respect
D. Planning, testing, coding, listening and designing

214. According to the Scrum Guide, who has the authority to cancel a Sprint? Understanding

A. Only the Scrum Master
B. The Developers, by majority vote
C. Only the Product Owner
D. Any stakeholder who funds the product

215. How often does the Agile Manifesto say working software should be delivered? Understanding

A. Once, at the end of the project
B. Every day, without exception
C. Only when the full specification has been signed off
D. Frequently, from a couple of weeks to a couple of months, preferring the shorter timescale

216. For a one-month Sprint, what is the maximum timebox of the Sprint Retrospective in the 2020 Scrum Guide? Understanding

A. 3 hours
B. 4 hours
C. 8 hours
D. 1 hour

217. A Scrum team completed 21, 25 and 26 story points in its last three Sprints. What is its average velocity? Application

A. 26 points per Sprint
B. 72 points per Sprint
C. 23 points per Sprint
D. 24 points per Sprint

218. A team's average velocity is 24 points per Sprint, and 150 points remain in the Product Backlog. How many Sprints should it forecast to finish? Application

A. 6
B. 6.25
C. 7
D. 5

219. A team of 5 developers has a 10-working-day Sprint and plans 6 focused hours per person per day. One developer is on leave for 2 days. What is the team's capacity in hours? Application

A. 300 hours
B. 288 hours
C. 240 hours
D. 384 hours

220. A Sprint starts with 120 points of work and lasts 10 working days. On a straight ideal burndown line, how many points should remain at the end of day 4? Application

A. 48
B. 72
C. 80
D. 60

221. In the same 10-day, 120-point Sprint, 90 points actually remain at the end of day 4. What average daily burn is now needed to finish on time? Application

A. 12 points per day
B. 18 points per day
C. 9 points per day
D. 15 points per day

222. A Kanban team averages 15 items in progress and finishes 5 items per day. Using Little's law, what is the average lead time? Application

A. 3 days
B. 75 days
C. 0.33 days
D. 10 days

223. At the end of a Sprint, items of 5, 8, 3 and 5 points meet the Definition of Done. An 8-point item is 90% complete. What velocity should be recorded? Application

A. 28.2 points
B. 21 points
C. 29 points
D. 25 points

224. Last Sprint a team completed 24 points with 240 available hours. Next Sprint, holidays leave 200 hours. Assuming the same rate, what should it forecast? Application

A. 20 points
B. 24 points
C. 29 points
D. 16 points

225. A Product Owner wants to use the Daily Scrum to assign each developer's tasks for the day. What does the Scrum Guide say? Application

A. The Product Owner should run the Daily Scrum and assign tasks
B. The Scrum Master should assign tasks instead
C. The Daily Scrum is for the Developers, who self-manage their own plan for the day
D. Tasks should be assigned by the stakeholders at the Daily Scrum

226. Mid-Sprint, a senior stakeholder asks the Developers to add a new feature that would endanger the Sprint Goal. What should happen? Application

A. The Developers add it at once to keep the stakeholder happy
B. Send the request to the Product Owner, who can add and order it in the Product Backlog for a future Sprint
C. The Scrum Master cancels the Sprint and restarts with the new feature
D. The Developers drop testing to make room for the feature

227. A Kanban board's Testing column has a WIP limit of 3 and already holds 3 items. A developer finishes coding a new item. What should they do? Application

A. Push it into Testing anyway, since coding is finished
B. Raise the WIP limit to 4 for this item
C. Start coding another item while the first waits in Testing
D. Leave it in the development 'done' area, or help clear Testing, rather than push it into Testing

228. A developer writes a failing automated test for a new function before writing the function itself. Which XP practice is this? Application

A. Test-first development
B. Continuous integration
C. Refactoring
D. Small releases

229. Developers merge their work into the main branch several times a day, and an automated build runs all tests on each merge. Which XP practice is this? Application
    A. Pair programming
    B. Continuous integration
    C. Collective ownership
    D. On-site customer

230. Which artifact holds the Sprint Goal, the selected Product Backlog items and the plan for delivering them? Application
    A. The Product Backlog
    B. The Sprint Backlog
    C. The Increment
    D. The burndown chart

231. After management links bonuses to velocity, a team's velocity jumps from 24 to 40 points in one Sprint with no other change. What is the most likely explanation? Analysis
    A. The team became 67% more productive overnight
    B. Velocity rose because the Sprint was made shorter
    C. The Product Owner reordered the backlog more effectively
    D. Story-point inflation: estimates grew, so velocity rose without more value being delivered

232. A Sprint burndown stays flat for the first eight days, then drops to zero on the last day. What does this most likely show? Analysis
    A. The team is working at a perfectly steady pace
    B. Work is batched into large items that are marked done only at the end, hiding progress; items should be smaller
    C. The Product Owner added scope every day
    D. The Sprint was too short to measure

233. A Scrum Master assigns tasks to each developer every morning and reports their individual progress to management. What is wrong? Analysis
    A. Nothing; assigning tasks is the Scrum Master's main accountability
    B. The Product Owner should be the one assigning tasks
    C. It undermines the Developers' self-management; the Scrum Master should coach and remove impediments, not direct the work
    D. The tasks should be assigned weekly, not daily

234. At the end of a Sprint, one Product Backlog item is coded but has not been tested as the Definition of Done requires. What should happen to it? Analysis
    A. It is shipped, with testing scheduled for next Sprint
    B. It is not part of the Increment, cannot be presented as done at the Sprint Review, and returns to the Product Backlog
    C. It is counted as half-done in velocity and demonstrated
    D. The Sprint is automatically extended until the item is tested

235. A company makes a committee of three managers the 'Product Owner', and they often give the team conflicting priorities. What does the Scrum Guide say? Analysis
    A. The Product Owner is one person, not a committee; others can represent stakeholder needs, but one person decides the ordering
    B. A committee is fine as long as it votes on every item
    C. The Scrum Master should resolve the conflicts by ordering the backlog
    D. The Developers should choose whichever priority they prefer

236. An operations support team receives unpredictable tickets of varying urgency all day. Which approach fits best? Analysis
    A. Scrum with one-month Sprints, refusing new tickets until the next Sprint
    B. Waterfall, with all tickets gathered and specified up front
    C. XP pair programming alone, with no workflow control
    D. Kanban, because continuous flow with WIP limits handles unplanned, varying-priority work without fixed iterations

237. Team A's velocity is 30 points per Sprint and Team B's is 45. Management concludes that Team B is 50% more productive. What is the best critique? Analysis
    A. The conclusion is correct, since story points are a universal unit
    B. Team A must be more productive, since lower velocity means more careful work
    C. Story points are relative estimates calibrated within each team, so velocities cannot be compared across teams
    D. Productivity can be compared only if both teams use one-week Sprints

238. A Kanban team's average lead time is 10 days and its throughput is 2 items per day. It wants a 5-day lead time without changing throughput. Using Little's law, what should it do? Analysis
    A. Cut average WIP from 20 to 10 items
    B. Raise average WIP from 20 to 40 items
    C. Cut average WIP from 10 to 5 items
    D. Keep WIP at 20, since lead time does not depend on WIP

239. To go faster, an XP team stops writing automated tests but keeps refactoring heavily. Regressions soon multiply. What is the best diagnosis? Analysis
    A. Refactoring itself causes bugs and should be banned
    B. Safe refactoring depends on an automated test suite to catch behaviour changes, so test-first development should be restored
    C. Pair programming should be stopped to save time
    D. More frequent releases will fix the regressions

240. A manager insists that the Sprint burndown chart is a mandatory Scrum artifact. Which response matches the 2020 Scrum Guide? Analysis
    A. Correct: the burndown chart is the fourth Scrum artifact
    B. Correct: the burndown chart replaced the Sprint Backlog in 2020
    C. Burndowns are a useful forecasting practice, but the Guide's artifacts are only the Product Backlog, Sprint Backlog and Increment
    D. Burndowns are banned by the Scrum Guide

241. Which statement is one of the four values in the Manifesto for Agile Software Development? Understanding
    A. Customer collaboration over contract negotiation
    B. Working software over customer collaboration
    C. Following a plan over responding to change
    D. Processes and tools over individuals and interactions

242. The Agile Manifesto says it values the items on the left 'over' the items on the right. How should 'over' be read? Understanding
    A. The items on the right should be dropped entirely
    B. The items on the right still have value, but the items on the left are valued more
    C. Both sides are valued equally and teams choose freely
    D. The items on the right apply only to large, regulated projects

243. How many principles sit behind the Agile Manifesto? Understanding

A. 4
B. 5
C. 12
D. 10

244. According to the Agile principles, what is the primary measure of progress? Understanding

A. Story points completed
B. Percentage of the plan followed
C. Pages of documentation approved
D. Working software

245. In the 2020 Scrum Guide, what is the maximum timebox of Sprint Planning for a one-month Sprint? Understanding

A. 4 hours
B. 8 hours
C. 3 hours
D. 15 minutes

246. In the 2020 Scrum Guide, what is the maximum timebox of the Sprint Review for a one-month Sprint? Understanding

A. 8 hours
B. 2 hours
C. 4 hours
D. 1 day

247. Which three topics does Sprint Planning address in the 2020 Scrum Guide? Understanding

A. What was done yesterday, what will be done today, and what is blocking progress
B. What went well, what went badly, and what to improve
C. Who owns each task, how many hours each takes, and when each is due
D. Why is this Sprint valuable, what can be Done this Sprint, and how will the chosen work get done

248. How does the 2020 Scrum Guide describe the Sprint Review? Understanding

A. A working session to inspect the outcome of the Sprint and determine future adaptations
B. A formal sign-off where the Product Owner accepts or rejects each item
C. A presentation by the Developers to management on individual performance
D. A meeting held only when a release is planned

249. What does the 2020 Scrum Guide say about the Product Goal? Understanding

A. It is set fresh at every Sprint Planning and replaced at every Sprint Review
B. It is the commitment for the Increment and defines its quality standard
C. It is the long-term objective for the Scrum Team, and the team must fulfil or abandon one before taking on the next
D. It is optional and used only when several Scrum Teams share a product

250. According to the 2020 Scrum Guide, what happens if the organisation already has a Definition of Done as a standard? Understanding

A. Each Scrum Team must replace it with its own definition
B. The Product Owner may waive it when deadlines are tight
C. It applies only to the final release, not to each Increment
D. All Scrum Teams must follow it as a minimum

251. In the 2020 Scrum Guide, what is Product Backlog refinement? Understanding

A. The ongoing act of breaking down Product Backlog items and adding detail such as description, order and size
B. A timeboxed event held once per Sprint just before the Sprint Review
C. The Scrum Master's review of whether the Developers estimated correctly
D. Removing every item that will not fit in the next Sprint

252. In the 2020 Scrum Guide, who owns the Sprint Backlog and may update it during the Sprint? Understanding

A. The Product Owner, because they order all backlogs
B. The Developers, because it is a plan by and for the Developers
C. The Scrum Master, because they track progress
D. Key stakeholders, after the Sprint Review

253. In the 2020 Scrum Guide, the Scrum Master is accountable for what? Understanding

A. Maximising the value of the product
B. Creating a usable Increment each Sprint
C. Approving the Definition of Done
D. The Scrum Team's effectiveness

254. How is the Sprint Goal created in the 2020 Scrum Guide? Understanding

A. The Product Owner proposes how the product could increase in value, and the whole Scrum Team collaborates to define the Sprint Goal
B. The Product Owner writes it alone before Sprint Planning
C. The Developers write it alone after selecting their tasks
D. The Scrum Master derives it from the team's velocity

255. Which statement about Increments matches the 2020 Scrum Guide? Understanding

A. Exactly one Increment is created per Sprint, at the Sprint Review
B. Several Increments may be created within one Sprint, and one may be delivered to stakeholders before the Sprint ends
C. An Increment may be released only after the Sprint Review approves it
D. An Increment can include work that has not yet met the Definition of Done

256. According to the 2020 Scrum Guide, how large is a Scrum Team typically? Understanding

A. Exactly 7 people
B. 3 to 5 people
C. 10 or fewer people
D. Up to 20 people

257. According to the 2020 Scrum Guide, Scrum is founded on which two ideas? Understanding

A. Empiricism and lean thinking
B. Detailed upfront planning and phase gates
C. Pair programming and test-driven development
D. Earned value and critical path analysis

258. In Scrum, which event is the container for all the other events? Understanding

A. Sprint Planning
B. The Sprint
C. The Sprint Review
D. The Daily Scrum

259. What does it mean that Kanban is a 'pull' system? Understanding

A. The manager pulls work from the team at fixed dates
B. Customers pull the product from the store when it is released
C. A stage takes new work only when it has capacity, instead of having work pushed onto it
D. Each column pulls the oldest item regardless of its WIP limit

260. On a cumulative flow diagram, what does the vertical distance between the 'arrived' line and the 'done' line at a given date show? Understanding

A. The average lead time for an item
B. The team's velocity for that day
C. The number of defects found so far
D. The amount of work in progress at that date

261. In Extreme Programming, what does 'collective code ownership' mean? Understanding

A. Each module has one owner who must approve all changes
B. Any pair may change any part of the code base when needed
C. The customer owns the copyright of the code
D. Code is owned by whoever wrote the most lines

262. In Extreme Programming, what is a 'spike'? Understanding

A. A sudden increase in velocity after refactoring
B. A large story that cannot be split
C. A short, timeboxed experiment to explore a technical or design question so that a story can be estimated
D. A production failure that stops the release

263. Which XP practice says developers should not routinely work long overtime because tired people make more mistakes? Understanding

A. Small releases
B. Simple design
C. Coding standards
D. Sustainable pace

264. In the INVEST checklist for user stories, what does the 'I' stand for? Understanding

A. Independent
B. Incremental
C. Integrated
D. Iterative

265. A team's last three velocities were 18, 22 and 20 points. 100 points remain in the Product Backlog. Using the lowest and highest velocities as a range, how many more Sprints should it forecast? Application

A. 4 to 5 Sprints
B. Exactly 5 Sprints
C. 5 to 6 Sprints
D. 6 to 7 Sprints

266. In planning poker, three developers show 3, 5 and 13 for the same story. What should happen next? Application

A. The team records the average, 7 points
B. The Scrum Master picks the most common estimate
C. The highest estimate is taken to be safe
D. The people with the lowest and highest estimates explain their reasoning, then everyone votes again

267. A story 'Students can pay fees online' is too big to finish in one Sprint. Which split is best? Application

A. Thin end-to-end slices, such as 'pay by card' and then 'pay by bank transfer', each usable on its own
B. Separate stories for the database layer, the API layer and the UI layer
C. One story for coding and a second story for testing
D. Keep it whole and extend the Sprint until it is done

268. A Scrum Team wants to discuss how to make its slow build pipeline faster and agree on improvements. In which event does this belong? Application

A. The Sprint Review
B. The Sprint Retrospective
C. The Daily Scrum
D. Product Backlog refinement

269. Mid-Sprint, the Developers realise that the selected work is larger than they expected. What does the 2020 Scrum Guide say they should do? Application

A. Extend the Sprint until all selected items are finished
B. Lower the Definition of Done so the items can be finished
C. Cancel the Sprint and start planning again
D. Collaborate with the Product Owner to clarify and renegotiate the scope of the Sprint Backlog without endangering the Sprint Goal

270. The Developers want to replace the 'three questions' at their Daily Scrum with walking the board from right to left. What does the 2020 Scrum Guide say? Application

A. That is fine; the Developers may choose any structure that focuses on progress toward the Sprint Goal and produces a plan for the next day
B. Not allowed; the three questions are mandatory
C. Only allowed if the Scrum Master runs the meeting
D. Only allowed for Sprints shorter than two weeks

271. A Product Owner is also coding items from the Sprint Backlog this Sprint. How should they take part in the Daily Scrum, according to the 2020 Scrum Guide? Application

A. They must stay silent because the event is only for Developers
B. They participate as a Developer
C. They lead the event because they own the backlog
D. They must not attend while they hold the Product Owner accountability

272. A release burnup shows total scope of 200 points, 120 points done, a steady velocity of 20 points per Sprint, and scope growing by 5 points every Sprint. How many more Sprints until done? Application

A. 4
B. 5
C. 6
D. 7

273. A Kanban team keeps about 24 items in progress, and each item takes 8 days on average from start to finish. Using Little's law, what is its throughput? Application

A. 3 items per day
B. 32 items per day
C. 0.33 items per day
D. 192 items per day

274. A Kanban team records four items. A starts on day 2 and finishes on day 5; B runs from day 3 to day 9; C from day 4 to day 7; D from day 6 to day 12. Measuring cycle time as finish day minus start day, what is the average cycle time? Application

    A. 3 days
    B. 4.5 days
    C. 6 days
    D. 18 days

275. In a pair, one developer types the code while the other reviews each line and thinks about the overall design. What is the reviewing role called? Application

    A. The driver
    B. The tester
    C. The navigator
    D. The coach

276. A developer removes duplicated code and renames unclear methods. All existing tests still pass and the program's behaviour has not changed. What is this called? Application

    A. Debugging
    B. Re-engineering the requirements
    C. Regression testing
    D. Refactoring

277. An XP team keeps waiting days for email answers about business rules. Which XP practice addresses this? Application

    A. Collective code ownership
    B. Having a real customer available to the team full time (on-site customer)
    C. Continuous integration
    D. Coding standards

278. A Product Owner writes acceptance criteria as: 'Given a registered student, when she enters a wrong password three times, then her account is locked for 15 minutes.' Which approach is this? Application

    A. Planning poker
    B. MoSCoW prioritisation
    C. Behaviour-driven development (Gherkin-style scenarios)
    D. A burndown forecast

279. Using SAFe's Weighted Shortest Job First (cost of delay divided by job size), which item should be done first? A: cost of delay 20, size 10. B: 12, size 3. C: 30, size 15. D: 8, size 1. Application

    A. A
    B. B
    C. C
    D. D

280. Senior managers ask for help understanding how to adopt Scrum across several departments. Under the 2020 Scrum Guide, whose accountability includes this? Application

    A. The Scrum Master, who leads, trains and coaches the organisation in its Scrum adoption
    B. The Product Owner, who maximises the product's value
    C. The Developers, who self-manage their work
    D. No one, because Scrum covers only a single team

281. At the Sprint Review, stakeholders suggest a new feature that would open a new market. What is the usual outcome? Application
    A. The Developers add it to the current Sprint Backlog immediately
    B. It is rejected because the Sprint Goal is fixed
    C. The Product Backlog may be adjusted to include it
    D. The Scrum Master writes it into the Definition of Done

282. On day 5 of a two-week Sprint, a feature meets the Definition of Done and a client needs it urgently. What does the 2020 Scrum Guide allow? Application
    A. It must wait until the Sprint Review approves it
    B. It must wait until the end of the Sprint so velocity is counted correctly
    C. It can be released only if the Sprint is cancelled
    D. It may be delivered now, because an Increment can be released before the Sprint ends

283. A product's market and requirements are changing fast, and the team keeps building things that are no longer wanted by the time a one-month Sprint ends. What change does the Scrum Guide's reasoning support? Application
    A. Use shorter Sprints, to get more learning cycles and limit risk to a smaller time frame
    B. Use longer Sprints, so more is delivered before feedback
    C. Freeze the Product Backlog for three months
    D. Drop the Sprint Review to save time

284. A Kanban team has a throughput of 4 items per week, and 30 similar items are waiting. Roughly how many weeks should it forecast to finish them? Application
    A. 7 weeks
    B. 8 weeks
    C. 120 weeks
    D. 4 weeks

285. For a story that only needs CSV export, a developer builds a general plug-in framework 'in case we need more formats later'. Which XP principle does this break? Application
    A. Collective code ownership
    B. Continuous integration
    C. Sustainable pace
    D. Simple design (YAGNI: you aren't gonna need it)

286. A Scrum Team of nine has split itself into a UI sub-team, a back-end sub-team and a test sub-team, each with its own lead. What is the best assessment under the 2020 Scrum Guide? Analysis
    A. It conflicts with the Guide: a Scrum Team is a cohesive, cross-functional unit with no sub-teams or hierarchies
    B. It is fine, because nine people are too many to work as one group
    C. It is required, because Developers must specialise
    D. It is fine as long as the leads report to the Scrum Master

287. A team's Daily Scrum regularly runs 45 minutes because people solve technical problems in it. What is the best fix? Analysis
    A. Allow the extra time because solving problems is valuable
    B. Keep the Daily Scrum to 15 minutes and let the people involved meet straight afterwards to solve the problem
    C. Move the Daily Scrum to once a week so it can be longer
    D. Have the Scrum Master solve the problems before the meeting

288. A team holds a Retrospective every Sprint, lists many problems, but nothing ever changes. What does the 2020 Scrum Guide suggest is missing? Analysis
    A. Inviting more stakeholders to the Retrospective
    B. Extending the Retrospective to a full day
    C. Identifying the most helpful changes and acting on them, for example by adding them to the next Sprint Backlog
    D. Having the Scrum Master write a report for management

289. Every fourth Sprint, a team runs a 'hardening Sprint' that only does integration and testing of the previous three Sprints' work. What does this most likely show? Analysis
    A. The earlier Sprints did not produce truly Done, usable Increments, so the Definition of Done is too weak
    B. The team is following Scrum correctly by grouping testing
    C. The team's velocity is too high and should be reduced
    D. The Product Owner is ordering the backlog badly

290. A client wants a fixed scope, a fixed date and a fixed price, and also wants the team to 'be agile'. What is the most realistic agile response? Analysis
    A. Fix the scope and date, and let quality vary
    B. Fix the date and cost, and let scope vary, delivering the highest-value items first
    C. Fix all three and add overtime if the team falls behind
    D. Fix the scope, and let the date and cost grow as needed

291. A team's cumulative flow diagram shows the 'In Test' band getting steadily wider while the 'Done' line flattens. What is the best interpretation and action? Analysis
    A. The team is speeding up and should take more work
    B. Developers are idle and should start more items
    C. Testing is a bottleneck; reduce WIP upstream and have developers help with testing
    D. The chart is wrong because bands should always widen

292. A Product Owner is unavailable for a whole Sprint, so the Developers guess at the requirements and later redo much of the work. What is the best diagnosis under the 2020 Scrum Guide? Analysis
    A. The Developers were at fault for not writing the requirements themselves
    B. The Scrum Master should have become the Product Owner for the Sprint
    C. Rework is normal in Scrum and needs no change
    D. The Product Owner remains accountable for clear Product Backlog items; they may delegate the work, but must ensure someone can answer questions

293. A burndown chart has stayed almost flat for several Sprints even though the team finishes about 20 points each Sprint. What is the most likely cause, and which chart would show it better? Analysis
    A. The team is not working; a velocity chart would prove it
    B. New scope is being added as fast as work is completed; a burnup chart shows done work and total scope as separate lines
    C. The chart is plotting hours instead of points; a Gantt chart would fix it
    D. Estimates are too small; a cumulative flow diagram would correct them

294. A Kanban board has columns but no WIP limits, and lead times keep growing as more items are started. What is the best diagnosis? Analysis
    A. The board needs more columns to show more detail
    B. The team should switch to a Gantt chart
    C. Without WIP limits work is pushed, not pulled, so too many items are in progress; set and respect WIP limits
    D. Lead time grows naturally as a product matures

295. Two Scrum Teams build the same mobile app. Each has its own Product Owner and its own Product Backlog, and they often build overlapping or conflicting features. What does the 2020 Scrum Guide recommend? Analysis

    A. Each team keeps its own backlog, and the Scrum Masters merge the features
    B. Merge the two teams into one team of any size
    C. Let each Product Owner own half of the app's screens
    D. Teams working on the same product share one Product Goal, one Product Backlog and one Product Owner

296. A team is building control software for a medical device. The regulator requires a detailed requirements and safety analysis before the system is approved. Which approach does Sommerville's guidance favour? Analysis

    A. A plan-driven or hybrid approach, because regulation demands extensive up-front analysis and documentation
    B. Pure Scrum with minimal documentation, because agile is always faster
    C. Kanban, because regulated work arrives unpredictably
    D. XP alone, because pair programming replaces safety analysis

297. Two days before a Sprint ends, three items are unfinished, so the team proposes extending the Sprint by three days. What is the best response under the 2020 Scrum Guide? Analysis

    A. Extend the Sprint, because finishing the items matters most
    B. Extend only if the Scrum Master approves
    C. Do not extend; the Sprint ends on schedule and the unfinished items return to the Product Backlog for re-ordering
    D. Mark the items as Done and finish them in the next Sprint

298. A manager says pair programming halves productivity because two people work on one task. What is the best critique? Analysis

    A. The manager is right, so pairing should be banned
    B. Pairing doubles speed because two people type at once
    C. Productivity should be measured only by lines of code per person
    D. It ignores that pairs catch defects early, share knowledge and often need only somewhat more total effort, so overall cost can fall

299. Team A keeps 30 items in progress and finishes 5 per day. Team B keeps 12 in progress and finishes 3 per day. Which team has the shorter average lead time, by Little's law? Analysis

    A. Team B, with 4 days against Team A's 6 days
    B. Team A, because it finishes more items per day
    C. Team A, with 6 days against Team B's 4 days
    D. They are equal, because both have high WIP

300. A Sprint Backlog is a list of ten unrelated small items with no Sprint Goal. What is the main thing the team loses? Analysis

    A. Its velocity, because velocity needs a Sprint Goal to be measured
    B. Focus and flexibility: without a Sprint Goal there is no single objective to guide trade-offs when work turns out differently than planned
    C. Its Product Goal, which is replaced each Sprint
    D. Its Daily Scrum, which cannot happen without a Sprint Goal

Software Requirements Engineering MCQs for NSCT

301. Which of these is a functional requirement for a university admissions portal? Understanding
    A. Every page shall load within 2 seconds on a 4G connection.
    B. The system shall let an applicant download a fee voucher as a PDF.
    C. The portal shall be available 99.5% of the time during the admission window.
    D. The system shall be developed in Java to match the university's existing systems.

302. According to Sommerville, what do non-functional requirements describe? Understanding
    A. Constraints on the system's services and qualities such as performance, security and usability
    B. The specific services the system must provide to its users
    C. The order in which developers should write the modules
    D. The internal class structure chosen by the designers

303. Sommerville divides non-functional requirements into which three top-level categories? Understanding
    A. Functional, domain and user requirements
    B. Product, organisational and external requirements
    C. Usability, efficiency and dependability requirements
    D. Business, user and system requirements

304. A requirement says the system must comply with the country's personal data protection legislation. In Sommerville's classification, this is which kind of non-functional requirement? Understanding
    A. Product (efficiency) requirement
    B. Organisational (development) requirement
    C. External (legislative) requirement
    D. Domain functional requirement

305. How do user requirements differ from system requirements? Understanding
    A. User requirements are written only as UML class diagrams; system requirements are written only as code
    B. User requirements are high-level natural-language statements for customers; system requirements are detailed descriptions of functions and constraints for developers
    C. User requirements describe hardware; system requirements describe software
    D. User requirements are non-functional and system requirements are functional

306. Which elicitation technique involves an analyst observing people in their real working environment to see how they actually do their jobs? Understanding
    A. Questionnaires
    B. Ethnography
    C. Joint application development (JAD) workshops
    D. Throwaway prototyping

307. What is a well-known weakness of interviews as an elicitation technique? Understanding
    A. Stakeholders may leave out tacit domain knowledge that they take for granted
    B. Interviews cannot capture any stakeholder opinions
    C. Interviews always require a working prototype first
    D. Interviews can only gather non-functional requirements

308. In requirements engineering, what is an SRS? Understanding
    A. A Software Requirements Specification: the agreed document stating what the system should do
    B. A Software Release Schedule: the dates of each release
    C. A System Risk Survey: the list of project risks
    D. A Source Revision Snapshot: a baseline in version control

309. In IEEE 830, what does it mean for an SRS requirement to be verifiable? Understanding

A. The requirement has only one possible interpretation
B. The origin of the requirement is recorded and it can be referenced in later work
C. A finite, cost-effective process exists to check that the software meets the requirement
D. The requirement can be changed easily without breaking the structure of the document

310. What does requirements traceability mean? Understanding

A. Being able to link each requirement back to its source and forward to the design, code and tests that realise it
B. Being able to trace every bug back to the programmer who wrote it
C. Keeping a log of every meeting held with the client
D. Recording how long each requirement took to implement

311. In Sommerville's requirements engineering process, which order do the three core activities follow in each iteration? Understanding

A. Elicitation and analysis, then specification, then validation
B. Specification, then elicitation and analysis, then validation
C. Validation, then specification, then elicitation and analysis
D. Elicitation and analysis, then validation, then specification

312. What is the main purpose of requirements validation? Understanding

A. Checking that the code correctly implements the design
B. Collecting requirements from stakeholders for the first time
C. Estimating the cost and schedule of each requirement
D. Checking that the requirements actually define the system the customer really wants

313. Which statement best describes domain requirements? Understanding

A. Requirements that describe only the user interface of the system
B. Requirements that list the internet domain names the system will use
C. Requirements that apply only during the maintenance phase
D. Requirements derived from the application domain, such as a rule or formula from the field the system works in

314. What distinguishes enduring requirements from volatile requirements? Understanding

A. Enduring requirements are non-functional; volatile requirements are functional
B. Enduring requirements are written by developers; volatile requirements are written by users
C. Enduring requirements are relatively stable and come from the organisation's core activity; volatile requirements are likely to change
D. Enduring requirements apply only to embedded systems

315. In the MoSCoW prioritisation technique, what do the letters M, S, C and W stand for? Understanding

A. Main, Secondary, Conditional, Waived
B. Mandatory, Suggested, Considered, Withdrawn
C. Must test, Should test, Can test, Will test
D. Must have, Should have, Could have, Won't have (this time)

316. In a UML use case diagram, what does an «extend» relationship from use case B to base use case A mean? Understanding

A. B adds optional behaviour to A at an extension point, only when a condition holds
B. B's behaviour is always inserted into A every time A runs
C. B is a generalisation of A and inherits all of A's steps
D. A cannot run unless B has finished first

317. Classify this requirement: "95% of search queries shall return results within 3 seconds." Application

A. Functional requirement
B. Domain requirement
C. Organisational (process) requirement
D. Non-functional (product, performance) requirement

318. During a review, an analyst asks whether a requirement for real-time face recognition at every campus gate can be built with current technology inside the project's budget. Which validation check is this? Application

A. Consistency check
B. Completeness check
C. Validity check
D. Realism check

319. A requirement states that a student portal must have 99.9% availability. What is the maximum downtime this allows in a 30-day month? Application

A. 4.32 minutes
B. 432 minutes
C. 8.76 hours
D. 43.2 minutes

320. A client writes: "The system should be easy to use." Which rewrite best turns this into a verifiable non-functional requirement? Application

A. The system shall be very user-friendly and intuitive for all users.
B. After four hours of training, a new clerk shall be able to process an admission form with an average of no more than two errors per hour.
C. The system shall use a modern front-end framework to improve usability.
D. Users shall be satisfied with how easy the system is to use.

321. Users of a new mobile banking app cannot describe what they want from a feature they have never seen before. Which elicitation technique is most suitable? Application

A. Sending a written questionnaire
B. Reading the bank's existing policy documents
C. Building a throwaway prototype for users to try
D. Freezing a detailed SRS and asking users to sign it

322. You need opinions on a proposed timetable feature from about 5,000 students across several campuses, and you want results you can count. Which elicitation technique fits best? Application

A. An online questionnaire
B. One-to-one interviews with each student
C. Ethnographic observation of each class
D. A single requirements workshop with all students

323. An SRS contains the requirement: "The system shall respond quickly to user actions." Which IEEE 830 quality does it most clearly fail? Application

A. Traceable
B. Modifiable
C. Verifiable
D. Ranked for importance

324. An SRS says in section 3.2 "Reports shall be exported only as PDF" and in section 5.4 "Reports shall be exported only as Excel files." What kind of problem is this? Application

A. An incompleteness (missing requirement)
B. A traceability gap
C. A verifiability problem
D. An inconsistency (conflicting requirements)

325. A requirement reads: "The system shall be developed using the company's standard Scrum process and delivered by 30 June." In Sommerville's classification it is which type? Application

A. Product requirement
B. External requirement
C. Organisational requirement
D. Functional requirement

326. A reliability requirement specifies POFOD = 0.001 for a request to unlock a lab door. What does it mean? Application

A. The door system may fail once every 1,000 hours
B. The system must recover from a failure within 0.001 seconds
C. On average, no more than 1 in 1,000 unlock requests may fail
D. The door system must be available 99.9% of the time

327. A server's reliability requirement is ROCOF = 2 failures per 1,000 operational hours. If it runs 24 hours a day for a 30-day month, how many failures should you expect at most? Application

A. 0.06
B. 1.44
C. 2
D. 14.4

328. In the user story "As a student, I want to reset my password so that I can log in if I forget it", which part states the benefit? Application

A. so that I can log in if I forget it
B. As a student
C. I want to reset my password
D. The whole story is the benefit

329. A requirements traceability matrix shows that requirement R7 has a design element and code module linked to it but no test case. What does this tell you? Application

A. R7 is complete and can be closed
B. R7 should be deleted because it has no tests
C. R7 is a non-functional requirement
D. R7 may never be verified, so a test case needs to be written for it

330. Which statement belongs in an SRS rather than in a design document? Application

A. The SMS gateway shall be implemented as a Singleton class.
B. The system shall send an SMS reminder to the patient 24 hours before an appointment.
C. Pending reminders shall be stored in a hash map keyed by patient ID.
D. A Python cron job shall run every minute to dispatch reminders.

331. A system passed every test written from its SRS, yet after delivery the client says it does not support how their staff actually work. Which RE activity most likely failed? Analysis

A. Coding: the programmers made logic errors
B. Unit testing: the tests had too little coverage
C. Configuration management: the wrong version was released
D. Requirements validation: nobody checked that the SRS described what the client really needed

332. The SRS for a payroll system contains a single line: "The system shall be secure." What is the best analysis and fix? Analysis
    A. It is fine as written, because testers can decide what secure means
    B. It should be deleted, because security belongs only in the design document
    C. It is a vague non-functional goal; refine it into specific, testable requirements such as locking an account after five failed logins and encrypting salary data at rest
    D. It should be rewritten as "The system shall be very secure" to show its priority

333. A wrong requirement found after release usually costs far more to fix than the same error found in a requirements review. What best explains this? Analysis
    A. Programmers charge more after release
    B. The error has spread into design, code, tests and documentation built on it, all of which must be changed
    C. Requirements reviews are always free
    D. Released software cannot be modified at all

334. Which requirement is best written: unambiguous, verifiable and stating a single need? Analysis
    A. The system shall generate the monthly attendance report within 10 seconds for up to 2,000 students.
    B. The system shall generate reports quickly and should support a large number of students.
    C. The system shall generate the attendance report, email it to parents and let teachers edit marks.
    D. The system shall use database indexes so that reports are generated fast.

335. A hospital system has two requirements: doctors must open a patient record within 2 seconds, and every record access must be protected by two-factor authentication. Doctors say entering a code each time makes the 2-second target impossible. What is the best requirements engineering response? Analysis
    A. Drop the security requirement, because performance always has priority
    B. Keep both requirements unchanged and let developers find a way
    C. Remove the performance requirement, because it is non-functional
    D. Treat it as a requirements conflict and negotiate with stakeholders, for example two-factor login once per shift with a short time-out, then record the agreed trade-off

336. In an ATM use case model, "Withdraw Cash" must always check the PIN, and it prints a receipt only if the customer asks for one. Which modelling is correct? Analysis
    A. Withdraw Cash «extend» Authenticate PIN; Print Receipt «include» Withdraw Cash
    B. Withdraw Cash «include» Authenticate PIN; Print Receipt «extend» Withdraw Cash
    C. Both Authenticate PIN and Print Receipt are «include» relationships
    D. Both Authenticate PIN and Print Receipt are «extend» relationships

337. A start-up's market changes every few weeks, and its requirements change with it. Which requirements approach fits best? Analysis
    A. Write a complete SRS up front and freeze it before any coding
    B. Skip requirements entirely and let developers build what they think is useful
    C. Keep a prioritised backlog of user stories, refine it every iteration, and validate with customers after each increment
    D. Collect requirements only at the end of the project during acceptance testing

338. A data protection regulation changes. The team must find every part of the system affected. Which traceability information is needed? Analysis
    A. Only the bug tracker's history for the last release
    B. Only links from test cases to the testers who ran them
    C. Links from the regulation to the requirements derived from it, and forward links from those requirements to design, code and tests
    D. Only the version-control commit messages

339. A requirement says: "The system shall log every login attempt with the time and IP address." It was written to meet a security goal. How should it be classified? Analysis
    A. Functional: it specifies something the system must do, even though it comes from a security non-functional goal
    B. Non-functional, because anything related to security is non-functional
    C. Domain requirement, because logging is common in every domain
    D. Organisational requirement, because the company asked for it

340. A requirement demands availability of at least 99.9%. Three designs are proposed: X (MTTF 999 h, MTTR 1 h), Y (MTTF 2,000 h, MTTR 4 h) and Z (MTTF 400 h, MTTR 0.25 h). Using availability = MTTF / (MTTF + MTTR), which designs meet it? Analysis
    A. Y only
    B. X and Z
    C. X and Y
    D. All three

341. In Sommerville's requirements elicitation and analysis process, which sequence of activities is correct? Understanding
    A. Requirements discovery, classification and organisation, prioritisation and negotiation, specification
    B. Specification, discovery, validation, prioritisation
    C. Design, discovery, coding, negotiation
    D. Prioritisation, discovery, specification, classification

342. In requirements engineering, who is a stakeholder? Understanding
    A. Only the customer who pays for the system
    B. Anyone who is affected by the system or has a direct or indirect influence on its requirements
    C. Only the end users who log in to the system
    D. Only the developers who build the system

343. According to Sommerville, which set of items does a requirements scenario usually include? Understanding
    A. The class diagram, sequence diagram and deployment diagram
    B. The cost estimate, schedule and risk register
    C. The starting situation, the normal flow of events, what can go wrong, concurrent activities and the state when it finishes
    D. The database schema, API list and test data

344. In a UML use case model, what is an actor? Understanding
    A. A class inside the system that performs a use case
    B. A developer who implements the use case
    C. A screen in the user interface
    D. A role played by a person or an external system that interacts with the system being modelled

345. IEEE 830 says a good SRS should be 'ranked for importance and/or stability'. What does this mean? Understanding

A. Requirements are listed in alphabetical order
B. Each requirement carries an identifier showing how important it is or how likely it is to change
C. Requirements are sorted by the cost of implementing them
D. Only the most important requirements are written down

346. In IEEE 830, what makes an SRS 'modifiable'? Understanding

A. Anyone can edit it without approval
B. It is written in a word processor rather than on paper
C. Its structure and style let changes be made easily, completely and consistently, without redundant copies of requirements
D. Its requirements may change after release without notice

347. According to IEEE 830, which condition is part of an SRS being 'complete'? Understanding

A. It contains no requirement that could ever change
B. It lists the source code modules that implement each requirement
C. It includes the project budget and staffing plan
D. It defines the software's responses to all realisable classes of input data, in both valid and invalid situations

348. Which standard superseded IEEE 830 for software requirements specifications? Understanding

A. ISO/IEC/IEEE 29148
B. ISO 9001
C. IEEE 1012
D. ISO/IEC 25010

349. In Sommerville's requirements change management process, which sequence of stages is correct? Understanding

A. Change implementation, then costing, then approval
B. Design change, then code change, then requirement change
C. Problem analysis and change specification, then change analysis and costing, then change implementation
D. Requirements validation, then elicitation, then specification

350. In Pressman's requirements engineering tasks, which one comes first? Understanding

A. Elaboration
B. Negotiation
C. Validation
D. Inception

351. In the Kano model, what is a 'basic' (must-be) requirement? Understanding

A. A feature users take for granted; its absence causes strong dissatisfaction, but its presence adds little satisfaction
B. A feature that delights users who did not expect it
C. A feature whose satisfaction rises in direct proportion to how well it performs
D. A feature users do not care about either way

352. What is the main purpose of a feasibility study at the start of requirements engineering? Understanding

A. To write the detailed SRS
B. To decide whether the proposed system is worth building with the available technology, budget and schedule
C. To design the system architecture
D. To produce the user manual

353. In requirements documents, what does the word 'shall' conventionally signal? Understanding

A. An optional, nice-to-have feature
B. A design suggestion for developers
C. A requirement that has been rejected
D. A mandatory requirement

354. Sommerville lists problems with natural-language requirements. What is 'requirements amalgamation'? Understanding

A. Several different requirements expressed together as one requirement
B. Functional and non-functional requirements mixed up
C. Two requirements that contradict each other
D. A requirement copied into several places

355. In a structured (form-based) requirements specification, which fields are typically recorded for each function? Understanding

A. Class names, attributes and methods
B. Inputs and their source, outputs and their destination, the action, pre-conditions, post-conditions and side effects
C. Programmer name, estimated hours and code reviewer
D. Release date, marketing message and price

356. Which of these is a standard requirements validation technique? Understanding

A. Code coverage analysis
B. Load balancing
C. Test-case generation
D. Refactoring

357. In elicitation, what is tacit knowledge? Understanding

A. Knowledge people use so routinely that they do not think to mention it
B. Knowledge recorded in the organisation's procedure manuals
C. Knowledge the analyst brings from previous projects
D. Knowledge that has been formally approved by management

358. Why does Sommerville say non-functional requirements are often more critical than individual functional requirements? Understanding

A. They are easier to test than functional requirements
B. Failing to meet one, such as reliability, can make the whole system unusable
C. They are always written first in the SRS
D. They apply only to small parts of the system

359. What is 'scope creep' in a software project? Understanding

A. A planned reduction in features to meet a deadline
B. A technique for estimating effort
C. The uncontrolled growth of requirements after the scope has been agreed
D. The gradual slowdown of a system's response time

360. Sommerville distinguishes a goal from a verifiable non-functional requirement. Which is a goal? Understanding

A. Medical staff shall be able to use all system functions after four hours of training
B. The system shall respond to 95% of queries within 2 seconds
C. The system shall be available 99.5% of the time during working hours
D. The system should be easy for medical staff to use

361. Sommerville suggests metrics for specifying non-functional requirements. Which metric fits robustness? Understanding

A. Processed transactions per second
B. Time to restart after failure
C. Training time
D. Number of target-dependent statements

362. How does a user story mainly differ from a use case? Understanding

A. A user story is a UML diagram; a use case is plain text
B. A user story is used only in waterfall projects; a use case only in agile
C. A user story is a short statement of a need written to prompt conversation; a use case describes the interaction flow between actors and the system in more detail
D. There is no difference; the terms are synonyms

363. What is a requirements baseline? Understanding

A. The first rough list of ideas from a brainstorming session
B. The minimum hardware needed to run the system
C. The set of requirements that the testers have already passed
D. A reviewed and agreed set of requirements that can then be changed only through a formal change process

364. Why might a team use a formal specification language such as Z for some requirements? Understanding

A. To remove ambiguity by stating requirements mathematically so they can be analysed precisely
B. To make requirements easier for non-technical customers to read
C. To avoid writing any natural-language requirements
D. To generate the user interface automatically

365. Classify this requirement: 'The system shall not disclose any personal information about patients to staff, apart from their name and reference number.' In Sommerville's scheme, what is it? Application

A. A product (performance) requirement
B. An organisational (development) requirement
C. An external (ethical) requirement
D. A functional requirement for data entry

366. Classify this requirement: 'The mobile app shall use no more than 50 MB of storage on the device.' Application

A. Organisational (operational) requirement
B. External (legislative) requirement
C. Functional requirement
D. Product (efficiency/space) requirement

367. Classify this requirement: 'Users of the clinic system shall authenticate themselves using their health authority identity card.' In Sommerville's scheme, what is it? Application

A. Organisational (operational) requirement
B. Product (security) requirement
C. External (legislative) requirement
D. Domain requirement

368. A release has a fixed 10-week timebox. Without one feature, the release would break a data protection law. In MoSCoW, how should that feature be ranked? Application

A. Should have
B. Must have
C. Could have
D. Won't have this time

369. Users were delighted by a feature that automatically fills in their address from a postcode, though nobody had asked for it. In the Kano model, what is this? Application

A. A basic (must-be) attribute
B. A performance attribute
C. An indifferent attribute
D. An excitement (delighter) attribute

370. A service must have availability of at least 0.999. Repairs take 2 hours on average (MTTR). Using availability = MTTF / (MTTF + MTTR), what is the minimum MTTF? Application

A. 1,998 hours
B. 2,000 hours
C. 1,000 hours
D. 999 hours

371. During acceptance testing, a door-control system failed 4 times out of 2,000 unlock requests. What is its observed POFOD? Application

A. 0.004
B. 0.002
C. 0.0005
D. 500

372. A chemical plant's emergency shutdown system is rarely called on, but a failure when it is called could be disastrous. Which reliability metric is most suitable for its requirement? Application

A. ROCOF (rate of occurrence of failures)
B. Availability
C. POFOD (probability of failure on demand)
D. Throughput in transactions per second

373. A traceability matrix shows that 34 of 40 requirements have at least one linked test case. What is the test coverage of the requirements? Application

A. 85%
B. 15%
C. 34%
D. 6%

374. A release has a budget of 10 effort units. Candidate requirements (value, cost) are A (9, 5), B (7, 3), C (6, 4) and D (4, 2). Which set gives the most total value within budget? Application

A. A and C
B. A, B and D
C. B, C and D
D. A and B

375. An analyst must replace a 15-year-old payroll system. Its users have left, but its manuals, forms and reports remain. Which elicitation technique should come first? Application

A. Ethnography with the current users
B. A brainstorming session with the original developers
C. Document analysis of the existing manuals, forms and reports
D. An online questionnaire to the original users

376. The admissions, finance and examinations departments each want different rules for the same student portal, and emails have not settled it. Which elicitation technique fits best? Application
    A. Separate interviews with each department, repeated until they agree
    B. Ethnography in one department only
    C. A questionnaire sent to all students
    D. A facilitated joint workshop (JAD) with key representatives of all three departments

377. Which requirement is ambiguous? Application
    A. The system shall display reports created between the 1st and the last day of the previous calendar month.
    B. The system shall display all reports for the last month.
    C. The system shall display reports created in the 30 days before today's date.
    D. The system shall display at most 50 reports per page.

378. Which requirement is compound and should be split into separate requirements? Application
    A. The system shall email the receipt to the student within 5 minutes of payment.
    B. The system shall update the ledger within 5 seconds of payment.
    C. The system shall email the receipt to the student and update the ledger within 5 seconds.
    D. The system shall keep each receipt for 7 years.

379. During a requirements review, a checker asks: 'Do any two requirements contradict each other?' Which validation check is this? Application
    A. Completeness check
    B. Realism check
    C. Verifiability check
    D. Consistency check

380. An online exam system sends fee payments to an external bank payment gateway and receives a confirmation. How should the gateway appear in the use case diagram? Application
    A. As an actor outside the system boundary
    B. As a use case inside the system boundary
    C. As a class in the use case diagram
    D. It should not appear because it is not a person

381. In the use case 'Withdraw Cash', the main flow assumes the card is valid. Where should 'the bank rejects the card' be described? Application
    A. In the main success scenario
    B. In a separate actor
    C. In an alternative or exception flow of the use case
    D. In the system's class diagram

382. A backlog contains: 'As a developer, I want to set up the database schema.' Which INVEST quality does it most clearly lack? Application
    A. Negotiable
    B. Small
    C. Estimable
    D. Valuable

383. A requirement says the payment service shall handle at least 500 transactions per second. A load test processes 28,800 transactions in one minute. Does it meet the requirement? Application
    A. No, it achieved 480 transactions per second
    B. Yes, it achieved 28,800 transactions per second
    C. Yes, it achieved 576 transactions per second
    D. No, it achieved 48 transactions per second

384. Traceability links show: requirement R3 is realised by design modules D1 and D2. D1 is tested by T1, T2 and T3, and D2 by T3 and T4. If R3 changes, how many distinct tests should be re-run? Application

A. 5
B. 4
C. 3
D. 2

385. A project baselined 120 requirements. Over one quarter, 30 of them were changed. What is the requirements volatility, and what does it suggest? Application

A. 4%; requirements are stable, so no action is needed
B. 25%; this is normal, so the baseline can be dropped
C. 90%; the project should be cancelled
D. 25%; high churn, so the team should strengthen change control and consider more iterative delivery

386. An SRS states: 'The system shall be available 24/7 with zero downtime.' What is the best analysis? Analysis

A. It is unrealistic and hard to verify; specify a measurable availability, such as 99.9% excluding agreed maintenance windows
B. It is a good requirement because it is strict
C. It is a functional requirement and needs no change
D. It should simply be removed because availability cannot be specified

387. Requirements for a ward management system were collected only from hospital managers. After rollout, nurses refuse to use it because it does not fit how they hand over patients. What went wrong? Analysis

A. The managers gave wrong information on purpose
B. Key stakeholders, the nurses who use it every day, were left out of elicitation
C. The developers coded the requirements incorrectly
D. The SRS should have been longer

388. An SRS has 'TBD' (to be determined) in twelve places, and the team plans to start design anyway. What is the best course of action? Analysis

A. Ignore the TBDs because developers can decide later
B. Delete the TBD sections so the SRS looks complete
C. Treat the SRS as incomplete: give each TBD an owner and a resolution date, and avoid designing the parts that depend on them
D. Stop the whole project until every TBD is resolved

389. A client tries a throwaway prototype built quickly to explore the user interface, likes it, and asks to put it straight into production. What is the best response? Analysis

A. Explain that a throwaway prototype ignores non-functional requirements such as security, reliability and maintainability, so the real system must be built properly
B. Agree, because the client is happy with it
C. Agree, but add more features first
D. Refuse to use anything learned from the prototype

390. Testers find features in the system that no requirement asked for. Which practice exposes this, and what is the problem called? Analysis

A. Forward traceability exposes scope creep, which is good practice
B. Backward traceability from code to requirements exposes gold plating, which is work no stakeholder requested
C. Regression testing exposes technical debt
D. Code review exposes requirements volatility

391. In interviews, clerks say they always check a form against a second register. Observation shows they skip this check when queues are long. What does this tell the analyst? Analysis
    A. The clerks lied, so their input should be ignored
    B. The register check should be removed from the requirements
    C. Stated procedures can differ from actual practice, so observation is needed to capture real work before writing requirements
    D. Interviews are always more accurate than observation

392. An SRS contains: 'The system shall process card payments using Stripe API v3.' A reviewer says this is design, not a requirement. When is it a legitimate requirement? Analysis
    A. Never; an SRS may not name any technology
    B. Always, because more detail is always better
    C. Only if the developers prefer Stripe
    D. When a stakeholder genuinely mandates it, for example because of an existing merchant contract, making it a design constraint

393. In a MoSCoW-prioritised release, the stakeholders have marked every single requirement as 'Must have'. What is the problem? Analysis
    A. Nothing, because it shows the requirements are important
    B. Prioritisation has lost its meaning, leaving no contingency; DSDM advises Must haves take no more than about 60% of the effort
    C. The release will finish early, because all items are clear
    D. The Won't-have items are now mandatory

394. A team plans to use only a UML use case diagram as the complete requirements specification for a banking app. What is the main weakness? Analysis
    A. Use case diagrams cannot show external systems
    B. Use case diagrams are only for agile projects
    C. The diagram shows actors and use case names but not flows, business rules or non-functional requirements
    D. Use case diagrams contain too much detail for customers

395. The user requirements for a library system are written with database table names and API calls. Library staff say they cannot review them. What is the best fix? Analysis
    A. Train the library staff in SQL
    B. Skip review by the library staff
    C. Merge user and system requirements into one technical document
    D. Rewrite the user requirements in plain language for the staff, and keep technical detail in the system requirements

396. A tax system is built to the SRS, but its tax calculations are wrong because the domain experts never mentioned a rounding rule they considered obvious. What does this illustrate? Analysis
    A. A domain requirement was missed because experts assumed it was common knowledge
    B. A product performance requirement was wrong
    C. The SRS was too long
    D. Prototyping caused the error

397. An agile team writes only functional user stories. In production, the app is slow and crashes under exam-day load. What is the best change to its requirements practice? Analysis
    A. Write more functional user stories
    B. Remove load testing to save time
    C. Capture non-functional requirements explicitly, as constraints, acceptance criteria or Definition of Done items, with measurable targets
    D. Leave non-functional needs to the operations team after release

398. A traceability matrix shows: R1 has no design link; R2 has design, code and test links; R3 has a code link but no design link; test T7 links to no requirement. Which observation is correct? Analysis
    A. Only R2 has a problem, because it has too many links
    B. T7 is fine because extra tests are always good
    C. R1 is complete because it has been written down
    D. R1 may be unimplemented, R3 skipped design documentation, and T7 may test behaviour nobody asked for

399. A government department outsources a system on a fixed-price contract to a separate vendor. Why does this situation favour a detailed SRS rather than a minimal backlog? Analysis
    A. The SRS acts as the contract baseline, so both sides need an agreed, detailed statement of what will be delivered
    B. Agile methods are illegal in government projects
    C. Detailed SRSs remove the need for testing
    D. A minimal backlog cannot be prioritised

400. An online questionnaire on a new timetable feature got 900 replies from the main campus and 12 from a smaller campus. Several questions began 'Wouldn't you agree that...'. What is the main risk? Analysis
    A. Too much data to analyse
    B. Biased results: one campus dominates and leading questions push answers, so the findings may not reflect real needs
    C. Questionnaires cannot be used for requirements
    D. The smaller campus has no requirements

Software Project Management MCQs for NSCT

401. Pressman describes effective software project management as focusing on the "four Ps". What are they? Understanding
    A. Planning, programming, profiling and patching
    B. People, product, process and project
    C. People, price, place and promotion
    D. Product, performance, portability and privacy

402. What does COCOMO stand for? Understanding
    A. Computed Code Model
    B. Constructive Cost Model
    C. Cost of Code Maintenance and Operation
    D. Collaborative Cost Monitoring

403. Basic COCOMO (1981) classifies projects into which three modes? Understanding
    A. Simple, average and complex
    B. Small, medium and large
    C. Application composition, early design and post-architecture
    D. Organic, semi-detached and embedded

404. Which project best fits the basic COCOMO organic mode? Understanding
    A. A small, experienced team building a familiar in-house payroll application with flexible requirements
    B. Flight-control software that must meet strict hardware and real-time constraints
    C. A medium-sized system built by a team with mixed experience and some rigid requirements
    D. An air-traffic system developed by several contractors under strict regulations

405. In basic COCOMO, in what unit is the effort E reported? Understanding

A. Lines of code per day
B. Person-months
C. Calendar months
D. Function points

406. What does Brooks's law state? Understanding

A. Effort grows linearly with the number of lines of code
B. Adding people to a late software project makes it later
C. Work expands to fill the time available for it
D. A project's cost doubles for every month it is late

407. What does a Gantt chart show? Understanding

A. The dependencies between activities as a network of nodes only
B. Project activities as horizontal bars against a calendar timeline
C. The class structure of the software
D. The number of defects found in each module

408. In the critical path method (CPM), what is the critical path? Understanding

A. The shortest path through the activity network
B. The longest path through the activity network, which sets the minimum project duration
C. The path containing the most activities
D. The path with the most expensive activities

409. Which three time estimates does PERT use for each activity? Understanding

A. Optimistic, most likely and pessimistic
B. Minimum, average and maximum cost
C. Earliest start, latest start and slack
D. Planned, earned and actual

410. How is the total float (slack) of an activity calculated? Understanding

A. Latest start − earliest start (equivalently latest finish − earliest finish)
B. Earliest finish − earliest start
C. Duration − latest start
D. Latest finish − earliest start

411. What is a work breakdown structure (WBS)? Understanding

A. A chart of the organisation's reporting lines
B. A hierarchical decomposition of the project's work into smaller deliverables and work packages
C. A list of defects that broke the build
D. A diagram of the software's module dependencies

412. In earned value management, what is earned value (EV)? Understanding

A. The actual money spent so far
B. The budgeted cost of the work planned to be done by now
C. The profit the project will make
D. The budgeted cost of the work actually completed so far

413. Which statement describes the Wideband Delphi estimation technique? Understanding

A. One manager gives a single estimate based on the deadline
B. Effort is calculated from lines of code with a fixed formula
C. The cost of a previous project is copied without adjustment
D. Several experts estimate independently, discuss the differences, and re-estimate over several rounds until estimates converge

414. Which COCOMO II sub-model is used once the system architecture has been designed and more detail about the components is known? Understanding
    A. Application composition model
    B. Early design model
    C. Post-architecture model
    D. Basic organic model

415. Why are software cost estimates made at the start of a project less accurate than those made later? Understanding
    A. Estimation formulas only work after coding has finished
    B. Managers deliberately inflate early estimates
    C. Requirements and design are still uncertain, so the range of possible outcomes is wide and narrows as the project progresses
    D. Early estimates use a different unit of effort

416. What is the difference between a milestone and a deliverable? Understanding
    A. A milestone is a document; a deliverable is a meeting
    B. They are two names for the same thing
    C. A deliverable has no date; a milestone always costs money
    D. A milestone is a checkpoint for judging progress; a deliverable is a work product handed over to the customer

417. Using basic COCOMO organic mode ($E = 2.4 \times KLOC^{1.05}$), what is the estimated effort for a 32 KLOC project? Application
    A. About 76.8 person-months
    B. About 91.3 person-months
    C. About 145.5 person-months
    D. About 13.9 person-months

418. A basic COCOMO organic project has effort E = 91.33 person-months. With $TDEV = 2.5 \times E^{0.38}$, what is the average team size? Application
    A. About 13.9 people
    B. About 7.6 people
    C. About 36.5 people
    D. About 6.6 people

419. Using basic COCOMO semi-detached mode ($E = 3.0 \times KLOC^{1.12}$), what is the estimated effort for a 50 KLOC system? Application
    A. 150.0 person-months
    B. About 145.9 person-months
    C. About 239.9 person-months
    D. About 17.0 person-months

420. An activity has optimistic time 2 days, most likely 4 days and pessimistic 12 days. What is its PERT expected time? Application
    A. 6 days
    B. 4 days
    C. 5 days
    D. 7 days

421. An activity has PERT estimates o = 3, m = 6 and p = 15 weeks. What is its variance? Application
    A. 2 weeks^2
    B. 12 weeks^2
    C. 4 weeks^2
    D. 24 weeks^2

422. Activities (duration in days, predecessors): A (3, none), B (4, A), C (2, A), D (5, B), E (6, C), F (2, D and E). What is the minimum project duration? Application

A. 13 days
B. 22 days
C. 12 days
D. 14 days

423. Activities (duration in days, predecessors): A (3, none), B (4, A), C (2, A), D (5, B), E (6, C), F (2, D and E). What is the total float of activity E? Application

A. 1 day
B. 0 days
C. 2 days
D. 6 days

424. At the status date, planned value PV = 50,000, earned value EV = 40,000 and actual cost AC = 50,000. What is the cost performance index (CPI)? Application

A. 1.25
B. 1.0
C. 0.2
D. 0.8

425. A project has PV = 80, EV = 60 and AC = 75 (in thousands). What is the schedule performance index (SPI)? Application

A. 0.80
B. 1.33
C. 1.07
D. 0.75

426. A project with budget at completion BAC = 200,000 has a CPI of 0.8. If the current cost performance continues, what is the estimate at completion (EAC = BAC / CPI)? Application

A. 160,000
B. 240,000
C. 250,000
D. 200,000

427. A team estimates a system at 12,000 lines of code, and its historical productivity is 400 LOC per person-month. What is the estimated effort? Application

A. 3 person-months
B. 48 person-months
C. 30 person-months
D. 300 person-months

428. A path has three activities with PERT estimates (o, m, p) of (2, 4, 12), (3, 6, 15) and (1, 3, 5) days. What is the standard deviation of the path's duration? Application

A. About 2.69 days
B. About 4.33 days
C. About 7.22 days
D. About 1.44 days

429. An activity has earliest start 5, duration 4 and latest finish 12 (days). What is its total float? Application

A. 4 days
B. 3 days
C. 7 days
D. 0 days

430. A project has EV = 60 and AC = 75 (in thousands). What is the cost variance (CV = EV − AC), and what does it mean? Application

A. +15: the project is under budget
B. −15: the project is behind schedule
C. −15: the project is over budget
D. +15: the project is ahead of schedule

431. Activities (duration in days, predecessors): A (3, none), B (4, A), C (2, A), D (5, B), E (6, C), F (2, D and E). If activity C is delayed by 2 days, what happens to the project duration? Analysis

A. It becomes 15 days
B. It stays 14 days
C. It becomes 16 days
D. It becomes 13 days

432. Activities: P (5 days), Q (3, after P), R (7, after P), S (4, after Q), T (2, after R and S). The manager wants a shorter project and proposes crashing R by 2 days. What is the result? Analysis

A. No gain: P-Q-S-T is also 14 days long, so the project stays at 14 days unless Q, S, P or T is shortened too
B. The project drops to 12 days, because R is on the critical path
C. The project drops to 13 days, because only half the saving counts
D. The project grows, because crashing adds risk

433. At a review, PV = 250,000, EV = 200,000 and AC = 180,000. What is the project's status? Analysis

A. Over budget and behind schedule
B. Over budget but ahead of schedule
C. Under budget and ahead of schedule
D. Under budget but behind schedule

434. In basic COCOMO organic mode (E = 2.4 × KLOC^1.05), a 32 KLOC estimate is 91.3 person-months and a 64 KLOC estimate is 189.1 person-months. What does this show? Analysis

A. Economies of scale: larger projects need less effort per line
B. Effort is exactly proportional to size
C. The model is wrong, because effort should double exactly
D. A diseconomy of scale: doubling size multiplies effort by about 2.07, because the exponent is greater than 1

435. A 5-person team is a month behind. The manager adds 5 new developers for the final month. Counting communication paths as n(n − 1) / 2, what is the most likely outcome? Analysis

A. The project finishes on time, because doubling staff halves the remaining time
B. The project slips further: paths rise from 10 to 45, and existing staff lose time training the newcomers
C. The project finishes early, because communication paths fall
D. Nothing changes, because new people never affect experienced ones

436. A project's critical path has PERT expected duration 15 weeks and standard deviation 2.69 weeks. Assuming a normal distribution, what is the approximate probability of finishing within 18 weeks? Analysis

A. About 87%
B. About 50%
C. About 13%
D. About 97%

437. COCOMO estimates a system at 91 person-months over about 14 months. A client asks why 91 developers cannot finish it in one month. What is the best answer? Analysis

A. Person-months are not interchangeable: tasks have sequential dependencies and communication overhead grows with team size
B. COCOMO forbids teams larger than 10 people
C. 91 developers would finish in one month, so the client is right
D. Effort in COCOMO is measured in calendar months, not person-months

438. At project inception there is only a short vision document, no detailed requirements and no code. Which estimation approach is most sensible? Analysis

A. Run basic COCOMO with an exact KLOC figure and commit to the single number
B. Refuse to estimate until coding is finished
C. Estimate by analogy or expert judgement as a range, then re-estimate when the requirements and architecture are known
D. Use the deadline the customer wants as the estimate

439. Activities: A (4 days, none), B (6, none), C (3, after A), D (5, after A and B), E (4, after C), F (2, after D and E). Shortening which single activity by 1 day shortens the whole project? Analysis

A. B
B. D
C. F
D. A

440. A team works heavy overtime. At review, PV = 100, EV = 110 and AC = 130 (in thousands). What is the best reading? Analysis

A. Ahead of schedule (SPI 1.10) but over budget (CPI about 0.85)
B. Ahead of schedule and under budget
C. Behind schedule but under budget
D. Behind schedule and over budget

441. In earned value management, what does planned value (PV) represent? Understanding

A. The budgeted cost of the work scheduled to be done by the status date
B. The budgeted cost of the work actually completed by the status date
C. The money actually spent on the work completed by the status date
D. The total budget approved for the whole project

442. In earned value management, what is actual cost (AC)? Understanding

A. The budgeted cost of the work completed up to the status date
B. The cost the plan expected to have spent by the status date
C. The cost really incurred for the work performed up to the status date
D. The forecast cost of the remaining work

443. A status report shows a schedule performance index (SPI) of 1.15. What does this mean? Understanding

A. The project has spent 15% more than its budget
B. More work has been completed than was planned by this date
C. The project is 15% behind schedule
D. The project has spent 15% less than its budget

444. How is schedule variance (SV) calculated in earned value management, and what does a negative value mean? Understanding

A. SV = EV − AC; a negative value means the project is over budget
B. SV = PV − EV; a negative value means the project is behind schedule
C. SV = AC − PV; a negative value means the project is ahead of schedule
D. SV = EV − PV; a negative value means the project is behind schedule

445. What does the budget at completion (BAC) represent? Understanding

A. The forecast total cost based on performance so far
B. The total planned budget for all the work in the project
C. The money still needed to finish the remaining work
D. The difference between the budget and the forecast cost

446. What is the free float of an activity in a project network? Understanding

A. How long it can be delayed without delaying the project's finish date
B. How long it can be delayed without delaying the earliest start of any following activity
C. The difference between its pessimistic and optimistic duration estimates
D. The time between the project start and the activity's earliest start

447. In a critical path network, what is true of every activity on the critical path? Understanding

A. It has the longest duration of all activities
B. It has no predecessors
C. Its total float is zero
D. It has the most resources assigned to it

448. In project scheduling, what does crashing an activity mean? Understanding

A. Running it in parallel with an activity it was planned to follow
B. Shortening its duration by adding resources, usually at extra cost
C. Removing it from the project scope to save time
D. Delaying it until resources become free, so that no one is over-allocated

449. What is fast tracking in schedule compression? Understanding

A. Adding extra people to critical activities to shorten them
B. Removing all float from non-critical activities
C. Performing activities in parallel that were planned to run one after another
D. Replacing the three PERT estimates with a single most likely estimate

450. Which statement about resource levelling is correct? Understanding

A. It always shortens the project by spreading work across more people
B. It never changes the critical path or the project end date
C. It shifts activity start dates to remove resource over-allocation and can lengthen the project
D. It assigns every activity the same number of resources

451. What do Albrecht's function points measure? Understanding

A. The functionality a system delivers to its users, independent of the programming language
B. The number of lines of code written per function
C. The number of defects found per function during testing
D. The cyclomatic complexity of each function in the code

452. Which five components are counted in function point analysis? Understanding

A. External inputs, external outputs, external inquiries, internal logical files and external interface files
B. Classes, methods, attributes, interfaces and packages
C. Requirements, design documents, modules, test cases and releases
D. Statements, branches, loops, functions and comments

453. Which size measure does the COCOMO II application composition model use? Understanding

A. Thousands of source lines of code (KLOC) after reuse adjustment
B. Unadjusted function points converted to KLOC
C. Story points from the product backlog
D. Application points (object points), based on screens, reports and components

454. How does intermediate COCOMO (Boehm, 1981) refine basic COCOMO? Understanding

A. It replaces KLOC with function points as the size measure
B. It adds a fourth project mode for web applications
C. It estimates the schedule first and derives effort from it
D. It multiplies the nominal effort by an effort adjustment factor built from 15 cost drivers

455. What does detailed (advanced) COCOMO add to intermediate COCOMO? Understanding

A. A Monte Carlo simulation of the three-point estimates
B. Phase-sensitive effort multipliers, so cost drivers are applied to each development phase separately
C. A switch from person-months to person-hours as the effort unit
D. Separate formulas for organic and embedded maintenance projects only

456. In agile estimation, what is planning poker? Understanding

A. The product owner assigns hours to each task and the team accepts them
B. Team members privately pick a story-point card, reveal at the same time, and discuss differences
C. Each developer bets on how many defects a release will contain
D. The team estimates only the largest story and scales the others from it

457. What is timeboxing in project planning? Understanding

A. Fixing the scope and extending the time until all of it is done
B. Recording the exact hours each developer spends on each task
C. Grouping activities in boxes on a Gantt chart by phase
D. Fixing the time available for a piece of work and adjusting the scope to fit it

458. Which three constraints make up the classic project management "iron triangle"? Understanding

A. People, process and product
B. Requirements, design and testing
C. Scope, time and cost
D. Risk, quality and communication

459. What is scope creep? Understanding

A. The gradual loss of team members during a long project
B. The delay caused when the critical path changes
C. The reduction of features to meet a deadline
D. Uncontrolled growth in a project's requirements without matching changes to time or budget

460. How is a milestone shown in a project schedule? Understanding

A. As the longest activity on the critical path
B. As an event with zero duration that marks the completion of significant work
C. As an activity with a large amount of float
D. As a bar that spans the whole project on the Gantt chart

461. In an activity-on-arrow (AOA) network, what is a dummy activity? Understanding

A. An activity whose duration has not yet been estimated
B. A zero-duration arrow used only to show a dependency, drawn as a dashed line
C. An activity added to absorb float at the end of the project
D. A critical activity that uses no resources

462. Which team structure places one senior chief programmer in charge of design and key code, supported by a backup programmer and a librarian? Understanding
    A. Chief programmer team
    B. Egoless (democratic) team
    C. Self-organising Scrum team
    D. Matrix team

463. What does the to-complete performance index (TCPI) measure? Understanding
    A. The cost efficiency needed on the remaining work to finish within the budget
    B. The schedule efficiency achieved on the work completed so far
    C. The percentage of the budget already spent
    D. The ratio of remaining work to work already completed

464. Classical PERT assumes which distribution for each activity's duration, and which for the total length of a long path? Understanding
    A. Normal for each activity; beta for the path
    B. Uniform for each activity; uniform for the path
    C. Beta for each activity; approximately normal for the path, by the central limit theorem
    D. Exponential for each activity; Poisson for the path

465. Using basic COCOMO organic mode (E = 2.4 × KLOC^1.05), what is the estimated effort for a 10 KLOC project? Application
    A. About 24.0 person-months
    B. About 25.2 person-months
    C. About 26.9 person-months
    D. About 10.5 person-months

466. Using basic COCOMO embedded mode (E = 3.6 × KLOC^1.20), what is the estimated effort for a 100 KLOC system? Application
    A. About 360 person-months
    B. About 904 person-months
    C. About 432 person-months
    D. About 251 person-months

467. A semi-detached project is estimated at E = 120 person-months. With basic COCOMO TDEV = 2.5 × E^0.35, what is the development time? Application
    A. About 13.4 months
    B. About 42.0 months
    C. About 48.0 months
    D. About 5.3 months

468. A 20 KLOC organic project has a basic COCOMO effort of about 55.8 person-months. What is the implied productivity? Application
    A. About 2.8 LOC per person-month
    B. About 359 LOC per person-month
    C. About 1,116 LOC per person-month
    D. About 1,740 LOC per person-month

469. An intermediate COCOMO estimate gives a nominal effort of 60 person-months. Three cost drivers are rated with multipliers 1.15 (reliability), 0.85 (analyst capability) and 1.30 (complexity); the rest are nominal (1.00). What is the adjusted effort? Application
    A. About 78.0 person-months
    B. About 66.0 person-months
    C. About 198.0 person-months
    D. About 76.2 person-months

470. An activity has PERT estimates o = 6, m = 9 and p = 18 days. What is its expected time? Application

A. 10 days
B. 11 days
C. 9 days
D. 12 days

471. An activity has PERT estimates o = 5, m = 10 and p = 23 days. What is its standard deviation? Application

A. 9 days
B. 2.17 days
C. 3 days
D. 18 days

472. A critical path has a PERT expected length of 40 days and a standard deviation of 4 days. What is the Z value for a deadline of 46 days, and the approximate probability of meeting it? Application

A. Z = 6, almost 100%
B. Z = 0.375, about 65%
C. Z = 1.5, about 7%
D. Z = 1.5, about 93%

473. A project's critical path has a PERT expected length of 30 weeks and a standard deviation of 3 weeks. Which deadline gives roughly an 84% chance of on-time completion? Application

A. 30 weeks
B. 36 weeks
C. 39 weeks
D. 33 weeks

474. Activities (duration in days, predecessors): A (4, none), B (3, none), C (5, A), D (2, A and B), E (6, C), F (3, D), G (2, E and F). What is the minimum project duration? Application

A. 11 days
B. 25 days
C. 17 days
D. 13 days

475. Activities (duration in days, predecessors): A (4, none), B (3, none), C (5, A), D (2, A and B), E (6, C), F (3, D), G (2, E and F). What is the free float of activity B? Application

A. 1 day
B. 7 days
C. 0 days
D. 3 days

476. Activities (duration in days, predecessors): A (4, none), B (3, none), C (5, A), D (2, A and B), E (6, C), F (3, D), G (2, E and F). What is the latest start time of activity D? Application

A. Day 4
B. Day 10
C. Day 6
D. Day 12

477. At the status date PV = 120,000 and EV = 90,000. What is the schedule variance? Application

A. +30,000: ahead of schedule
B. −30,000: over budget
C. 0.75: behind schedule
D. −30,000: behind schedule

478. BAC = 500,000, EV = 200,000 and AC = 250,000. The overspend came from a one-off hardware price rise that will not recur, so remaining work is expected to run at budgeted rates. What is the estimate at completion? Application

A. 625,000
B. 450,000
C. 750,000
D. 550,000

479. BAC = 900,000, EV = 300,000 and AC = 360,000. Assuming the current CPI continues, what is the estimate to complete (ETC = EAC − AC)? Application

A. 720,000
B. 1,080,000
C. 600,000
D. 540,000

480. A project has BAC = 900,000 and a forecast EAC of 1,080,000. What is the variance at completion (VAC = BAC − EAC), and what does it mean? Application

A. −180,000: the project is expected to finish over budget
B. +180,000: the project is expected to finish under budget
C. −180,000: the project is expected to finish late
D. 1.2: the project is expected to cost 20% more

481. A project has BAC = 400,000. At review the work is 35% complete and the actual cost is 175,000. What is the CPI? Application

A. 1.25
B. 0.35
C. 0.8
D. 2.29

482. How many communication paths exist in a team of 6 people, using n(n − 1) / 2? Application

A. 30
B. 6
C. 15
D. 36

483. A system has 6 external inputs, 5 external outputs, 4 external inquiries, 3 internal logical files and 2 external interface files, all of average complexity (weights 4, 5, 4, 10 and 7). The 14 general system characteristics sum to 42. With FP = UFP × (0.65 + 0.01 × ΣF), what is FP? Application

A. 109
B. About 45.8
C. About 120.0
D. About 116.6

484. A project is estimated at 36 person-months, and one person-month costs PKR 250,000. What is the estimated labour cost? Application

A. PKR 6,944
B. PKR 250,036
C. PKR 9,000,000
D. PKR 3,000,000

485. A Scrum team's average velocity is 30 story points per two-week sprint, and the remaining product backlog is 240 points. Ignoring new work, about how long will it take to finish? Application

A. 8 weeks
B. 16 sprints, about 32 weeks
C. 4 sprints, about 8 weeks
D. 8 sprints, about 16 weeks

486. BAC = 500,000, EV = 200,000 and AC = 250,000. The sponsor insists the original budget must still be met. What is the to-complete performance index, and what does it imply? Analysis

A. TCPI = 1.2: remaining work must be done 20% more efficiently than planned
B. TCPI = 0.8: the team can afford to slow down on remaining work
C. TCPI = 1.25: this equals 1 / CPI, so no change is needed
D. TCPI = 0.6: only 60% of the budget remains, so the project is on track

487. Two projects report: X has PV = 100, EV = 70, AC = 74; Y has PV = 100, EV = 98, AC = 93 (in thousands). Which reading is correct? Analysis

A. X's main problem is schedule (SPI 0.70); Y is close to plan on both measures
B. X's main problem is cost (CPI 0.70); Y is far over budget
C. Both projects are behind schedule by the same amount
D. Y is in worse shape because it has spent more money

488. Activities: A (2 days), B (6, after A), C (6, after A), D (3, after B and C). Crash cost per day: A 500, B 300, C 250, D 700. What is the cheapest way to cut the project by 1 day? Analysis

A. Crash C for 250
B. Crash B and C together for 550
C. Crash D for 700
D. Crash A for 500

489. Activities (duration in days, predecessors): A (4, none), B (3, none), C (5, A), D (2, A and B), E (6, C), F (3, D), G (2, E and F). Activity F is delayed by 4 days. What happens to the project? Analysis

A. Nothing: F has 6 days of total float, so the project still takes 17 days
B. The project becomes 21 days, because every delay adds to the end date
C. The project becomes 18 days, because F uses up G's float
D. The critical path switches to A-D-F-G

490. A path has four activities with PERT estimates (o, m, p) in days: P (2, 4, 6), Q (3, 5, 13), R (4, 6, 8), S (1, 8, 9). Which activity adds the most uncertainty to the path's finish date? Analysis

A. S
B. R
C. Q
D. P

491. Path 1 has a PERT expected length of 20 weeks ($\sigma = 1$). A parallel near-critical path 2 has 19 weeks ($\sigma = 3$). The deadline is 22 weeks. Assuming independence, what is the best analysis? Analysis

A. Only path 1 matters, because it is the critical path; the chance is about 98%
B. Path 2 is safer because it is shorter, so the project chance is about 98%
C. The chance is the average of the two paths, about 91%
D. Path 2 is the bigger risk (about 84% vs 98% on-time), so the combined chance is only about 82%

492. A team plans to estimate a 50 KLOC avionics flight-control system with the basic COCOMO organic formula (E = 2.4 × KLOC^1.05) instead of the embedded formula (E = 3.6 × KLOC^1.20). What is the effect? Analysis

A. Effort is overestimated, because organic projects have more inexperienced staff
B. Effort is underestimated by exactly 1.5 times, the ratio of 3.6 to 2.4
C. Effort is underestimated by a factor of about 2.7 (about 146 vs 394 person-months)
D. There is little difference, because the exponents 1.05 and 1.20 are almost equal

493. A project planned for 10 months with BAC = 1,000,000 finally finishes in month 13 having delivered all its scope. What is the SPI at completion, and what does this show? Analysis

A. 0.77: the project took 13 months against a plan of 10
B. 1.0: classic SPI always reaches 1 at completion, so it cannot reveal a late finish by itself
C. 1.3: the project ran 30% over schedule
D. 0: no planned value remains at completion

494. Activities A (4 days) and B (3 days) both start the project and both need the only database specialist. C (2 days) follows A and B. D (7 days) runs independently. Both C and D lead to the end. What is the shortest duration once the specialist constraint is respected? Analysis

A. 7 days
B. 9 days
C. 6 days
D. 16 days

495. To win a contract, a company quotes the budget the client said it could afford rather than an estimate of the work. Which estimation problem does Sommerville call this, and what is the main risk? Analysis

A. Parkinson's law; the team will finish early and waste the remaining budget
B. Pricing to win; the project is likely to overrun or cut quality when the real effort exceeds the price
C. Estimation by analogy; it is accurate because the client knows the market
D. Algorithmic costing; the price is too high to win future work

496. Which situation is Brooks's law least likely to apply to? Analysis

A. Adding testers at the start of a project to run independent, well-documented test cases
B. Adding three developers to a tightly integrated module two weeks before release
C. Adding new staff who need weeks of training from the existing team near a deadline
D. Doubling the team on a late project whose remaining tasks are strictly sequential

497. Basic COCOMO gives a 20 KLOC organic project a TDEV of about 11.5 months. The client demands delivery in 6 months with the same scope. What is the best response? Analysis

A. Explain that compressing the schedule this far raises effort, cost and risk, and negotiate scope or date
B. Double the team, because effort is fixed and time will halve
C. Accept, because COCOMO estimates are usually pessimistic
D. Keep the team and ask everyone to work twice as fast

498. A manager's plan is only a Gantt chart with bars but no dependency links. A task slips, and she cannot tell whether the release date is affected. What does she need? Analysis

A. A longer Gantt chart that starts earlier
B. An activity network with dependencies, so the critical path and float can be found
C. A function point count of the system
D. A larger management reserve in the budget

499. A task has been reported as "90% complete" for six weekly reviews in a row. What is the best interpretation? Analysis
    A. The task is nearly done, so its earned value should be recorded as 90%
    B. The task is off the critical path, so its status does not matter
    C. The team is ahead of schedule, because 90% exceeds the planned value
    D. Progress is being judged subjectively; use measurable milestones or 0/100 credit to get true earned value

500. A release plan has 240 story points over 6 sprints. After 3 sprints, the burndown chart shows 150 points remaining. If velocity stays the same, what is the forecast? Analysis
    A. On time, because half the sprints are done
    B. About 7 sprints in total, 1 sprint late
    C. About 8 sprints in total, 2 sprints late
    D. About 5 sprints in total, 1 sprint early

Software Design MCQs for NSCT

501. In software design, what does cohesion measure? Understanding
    A. How closely the elements inside a single module belong together and work towards one purpose
    B. How many modules call a given module
    C. How strongly one module depends on other modules
    D. How many lines of code a module contains

502. In software design, what does coupling measure? Understanding
    A. The degree to which elements inside one module are related
    B. The number of classes in the system
    C. The speed at which modules exchange messages
    D. The degree of interdependence between modules

503. Which combination do Pressman and Sommerville recommend for a good modular design? Understanding
    A. High cohesion and high coupling
    B. High cohesion and low coupling
    C. Low cohesion and low coupling
    D. Low cohesion and high coupling

504. Which type of cohesion is considered the strongest (best)? Understanding
    A. Functional cohesion
    B. Sequential cohesion
    C. Communicational cohesion
    D. Logical cohesion

505. Which type of cohesion is considered the weakest (worst)? Understanding
    A. Temporal cohesion
    B. Logical cohesion
    C. Procedural cohesion
    D. Coincidental cohesion

506. Which type of coupling is considered the tightest (worst)? Understanding
    A. Data coupling
    B. Stamp coupling
    C. Control coupling
    D. Content coupling

507. Which type of coupling is the loosest (best) of these? Understanding

A. Common coupling
B. Control coupling
C. Data coupling
D. Stamp coupling

508. The Gang of Four (GoF) book groups its 23 design patterns into which three categories? Understanding

A. Creational, structural and behavioural
B. Creational, functional and procedural
C. Static, dynamic and hybrid
D. Architectural, structural and deployment

509. According to the GoF, what is the intent of the Singleton pattern? Understanding

A. Create families of related objects without naming their concrete classes
B. Ensure a class has only one instance and provide a global point of access to it
C. Let a class have exactly one method
D. Allow only one thread to run at a time

510. According to the GoF, what is the intent of the Observer pattern? Understanding

A. Provide a surrogate that controls access to another object
B. Define a family of algorithms and make them interchangeable
C. Convert the interface of a class into another interface clients expect
D. Define a one-to-many dependency so that when one object changes state, all its dependents are notified and updated automatically

511. According to the GoF, what is the intent of the Adapter pattern? Understanding

A. Attach additional responsibilities to an object dynamically
B. Provide a unified, simpler interface to a set of interfaces in a subsystem
C. Convert the interface of a class into another interface that clients expect
D. Separate an abstraction from its implementation so the two can vary independently

512. Which of these is a UML structural diagram? Understanding

A. Class diagram
B. Sequence diagram
C. Activity diagram
D. State machine diagram

513. In a UML class diagram, how does composition differ from aggregation? Understanding

A. Composition is strong ownership where the part cannot exist without the whole (filled diamond); aggregation is a weaker whole-part link where the part can exist independently (hollow diamond)
B. Composition uses a hollow diamond and aggregation uses a filled diamond
C. Composition means inheritance and aggregation means association
D. There is no difference; the two terms are interchangeable in UML

514. What does Parnas's principle of information hiding say? Understanding

A. Each module should hide a design decision that is likely to change behind a stable interface
B. All data should be stored in encrypted form
C. Source code should not be shared with the customer
D. Every module should expose all of its variables so others can reuse them

515. What is communicational cohesion? Understanding

A. The elements are grouped because the output of one is the input of the next
B. The elements are grouped because they run at the same time, such as at start-up
C. The elements of a module are grouped because they operate on the same input or output data
D. The elements are grouped only because they are in the same general category

516. According to the GoF, what is the intent of the Decorator pattern? Understanding

A. Ensure that a class has only one instance
B. Attach additional responsibilities to an object dynamically, as a flexible alternative to subclassing
C. Compose objects into tree structures to represent part-whole hierarchies
D. Define the skeleton of an algorithm and let subclasses redefine some steps

517. A function startup() opens the log file, resets all counters and loads the configuration. These tasks are unrelated except that they all run when the program starts. What kind of cohesion does it have? Application

A. Functional cohesion
B. Sequential cohesion
C. Temporal cohesion
D. Coincidental cohesion

518. A module named utils contains print_date(), compress_file() and compute_gpa(), which have nothing in common. What kind of cohesion is this? Application

A. Coincidental cohesion
B. Logical cohesion
C. Functional cohesion
D. Communicational cohesion

519. Module A calls B(record, flag), and B uses the flag to decide which of its internal branches to run. What type of coupling exists between A and B? Application

A. Data coupling
B. Control coupling
C. Content coupling
D. External coupling

520. Six modules all read and write the same global variables current_user and cart_total. What type of coupling is this? Application

A. Data coupling
B. Stamp coupling
C. Control coupling
D. Common coupling

521. A function print_label(student) receives a whole Student record but only uses student.roll_no. What type of coupling is this? Application

A. Data coupling
B. Common coupling
C. Content coupling
D. Stamp coupling

522. An application must use exactly one shared database connection pool, and every part of the code must reach the same object. Which GoF pattern fits? Application
    A. Prototype
    B. Singleton
    C. Builder
    D. Observer

523. Your code expects a PaymentProcessor interface with pay(amount), but a third-party library offers makePayment(rupees, currency) and you cannot change it. Which GoF pattern lets you use the library? Application
    A. Facade
    B. Adapter
    C. Decorator
    D. Strategy

524. A sorting component must let the user switch between quick sort, merge sort and heap sort at run time, without changing the code that calls it. Which GoF pattern fits best? Application
    A. Singleton
    B. Strategy
    C. Composite
    D. Abstract Factory

525. A video-conversion library has dozens of classes for codecs, buffers and bit-rate settings. You want client code to call a single convert(file, format) method. Which GoF pattern fits best? Application
    A. Adapter
    B. Proxy
    C. Bridge
    D. Facade

526. A UML class diagram shows Department 1 —— 1..* Teacher. What does this multiplicity mean? Application
    A. Each department has one or more teachers, and each teacher belongs to exactly one department
    B. Each department has exactly one teacher, and each teacher works in one or more departments
    C. Each department has zero or more teachers
    D. Each teacher belongs to at least one department and at most two

527. You need to show the order in which a User, LoginController and Database exchange messages during login. Which UML diagram is most suitable? Application
    A. Class diagram
    B. Deployment diagram
    C. Component diagram
    D. Sequence diagram

528. What does this Python code print? Application

```
class Config:
    _instance = None
    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)
            cls._instance.value = 0
        return cls._instance

a = Config()
a.value = 5
b = Config()
b.value += 1
print(a is b, a.value)
```

A. False 1
B. True 1
C. False 5
D. True 6

529. What does this Python code print? Application

```
class Tea:
    def cost(self): return 100
    def desc(self): return "Tea"

class Extra:
    def __init__(self, drink): self.drink = drink

class Milk(Extra):
    def cost(self): return self.drink.cost() + 30
    def desc(self): return self.drink.desc() + " + Milk"

class Honey(Extra):
    def cost(self): return self.drink.cost() + 20
    def desc(self): return self.drink.desc() + " + Honey"

order = Honey(Milk(Milk(Tea())))
print(order.desc(), order.cost())
```

A. Tea + Honey + Milk + Milk 180
B. Tea + Milk + Honey 150
C. Tea + Milk + Milk + Honey 180
D. Honey + Milk + Milk + Tea 100

530. A House is made up of Rooms, and if a House object is destroyed its Rooms no longer exist. How should this be drawn in a UML class diagram? Application

A. Aggregation: a hollow diamond at the Room end
B. Generalisation: a hollow triangle pointing to House
C. Composition: a filled diamond at the House end of the association
D. Dependency: a dashed arrow from House to Room

531. A class ReportManager fetches data from the database, formats it as HTML, emails the report and writes audit logs. What is the best assessment and fix? Analysis

A. It has high cohesion because all its methods are about reports
B. It has high coupling but perfect cohesion, so no change is needed
C. It has low cohesion; split it into classes with one responsibility each, such as a repository, formatter, mailer and logger
D. It should be merged with other classes to reduce the number of files

532. Module pair X passes only an integer roll number. Pair Y shares global variables. Pair Z passes a flag that selects the callee's logic. Order them from loosest to tightest coupling. Analysis

    A. X, Y, Z
    B. Z, X, Y
    C. X, Z, Y
    D. Y, Z, X

533. What does this Python code print, and what kind of coupling does the mode parameter create? Analysis

```
def process(data, mode):
    if mode == 1:
        return sum(data)
    elif mode == 2:
        return max(data)
    return len(data)

print(process([4, 9, 2], 2) + process([4, 9, 2], 3))
```

    A. 12; data coupling
    B. 18; control coupling
    C. 12; control coupling
    D. 15; stamp coupling

534. A drawing app has shapes (Circle, Square, Triangle) and renderers (Vector, Raster). Subclassing gives VectorCircle, RasterCircle and so on, and the number of classes grows with every new shape or renderer. Which GoF pattern addresses this? Analysis

    A. Adapter
    B. Bridge
    C. Decorator
    D. Composite

535. A coffee shop sells one base coffee with four optional extras, each either added or not. How many classes would you need to cover every combination by subclassing, compared with the Decorator pattern? Analysis

    A. 8 subclasses versus 4 decorators
    B. 16 subclasses versus 1 base class plus 4 decorators
    C. 4 subclasses versus 16 decorators
    D. 24 subclasses versus 1 decorator

536. A file manager must treat single files and folders (which hold files and other folders) the same way, for example calling size() on either. Which GoF pattern fits? Analysis

    A. Composite
    B. Iterator
    C. Observer
    D. Facade

537. A team uses a Logger Singleton that every class calls directly. Unit tests become hard to isolate. What is the best analysis? Analysis

    A. The problem is that the Singleton has too many instances
    B. The Singleton acts as hidden global state; injecting a logger through constructors (dependency injection) makes the dependency explicit and replaceable in tests
    C. The tests are wrong; Singletons never affect testing
    D. Replacing the Singleton with more global variables will fix it

538. A class diagram shows Car with a filled diamond to Engine, and Car with a hollow diamond to Passenger. Which statement is correct? Analysis

A. An Engine is owned by one Car and is destroyed with it; Passengers exist independently of the Car
B. Passengers are destroyed when the Car is destroyed; Engines are not
C. Both Engine and Passenger are subclasses of Car
D. Both relationships mean exactly the same thing

539. What does this Python code print? Analysis

```
class Subject:
    def __init__(self):
        self.observers = []
    def attach(self, o): self.observers.append(o)
    def detach(self, o): self.observers.remove(o)
    def notify(self, x):
        for o in self.observers:
            o.update(x)

class Counter:
    def __init__(self): self.total = 0
    def update(self, x): self.total += x

s = Subject()
c1, c2 = Counter(), Counter()
s.attach(c1); s.attach(c2)
s.notify(3)
s.detach(c1)
s.notify(4)
s.attach(c1)
s.notify(5)
print(c1.total, c2.total)
```

A. 12 12
B. 3 12
C. 8 9
D. 8 12

540. Six modules read the shared global variables db_host and db_port. Which change most reduces their coupling? Analysis

A. Add more global variables so each module has its own copy
B. Let each module modify the globals directly when it needs a different value
C. Pass the needed configuration values to each module as parameters (or through a small config object injected into it)
D. Merge all six modules into one large module

541. What is logical cohesion? Understanding

A. Elements are grouped because they all run at the same time
B. Elements perform similar kinds of task and one is selected, usually by a flag passed in
C. Each element's output is the next element's input
D. Every element contributes to one single, well-defined task

542. What is procedural cohesion? Understanding

A. Elements are grouped because they operate on the same input or output data
B. Elements are grouped because each one's output feeds the next
C. Elements are grouped because they follow a fixed order of execution, even though they work on different data
D. Elements are grouped only because they are executed at program start-up

543. What is sequential cohesion? Understanding

A. The elements share global variables with other modules
B. The elements are grouped because they must run in a fixed order but share no data
C. The elements are unrelated helper functions in one file
D. The output of one element in the module is the input to the next

544. What is external coupling? Understanding

A. Modules communicate only through simple parameters
B. One module modifies the internal data of another
C. A module passes a flag that controls another module's logic
D. Modules depend on an externally imposed data format, communication protocol or device interface

545. In software design, what is abstraction? Understanding

A. Focusing on the essential features of something while hiding details that are not needed at that level
B. Splitting the system into the largest possible number of modules
C. Writing all code in an abstract class
D. Removing comments to make the code shorter

546. What is stepwise refinement, as described by Wirth? Understanding

A. Starting from a high-level statement of function and elaborating it into more detail step by step
B. Repeatedly testing a program until no defects are found
C. Improving code structure without changing its behaviour
D. Releasing a product in small increments to users

547. According to Fowler, what is refactoring? Understanding

A. Adding new features to a system after release
B. Rewriting a system in a different programming language
C. Fixing a defect reported by a user
D. Changing a system's internal structure without changing its external behaviour

548. Pressman argues there is a "right" number of modules for a design. Why does over-modularising a system raise total cost? Understanding

A. The cost of each module falls, but the cost of integrating and interfacing many modules rises
B. Small modules cannot be tested individually
C. Each module must be written in a separate programming language
D. Small modules always have lower cohesion than large ones

549. Pressman describes functional independence as the result of which two design qualities? Understanding

A. High coupling and low cohesion
B. High cohesion and low coupling
C. Deep inheritance and many global variables
D. Large modules and few interfaces

550. What does the interface segregation principle state? Understanding

A. Clients should not be forced to depend on methods they do not use
B. Each interface should be implemented by exactly one class
C. Interfaces and implementations must be in separate files
D. A class should implement as many interfaces as possible

551. What does the dependency inversion principle state? Understanding

A. Low-level modules should call high-level modules instead of the reverse
B. Dependencies should be resolved at compile time rather than run time
C. High-level modules should not depend on low-level modules; both should depend on abstractions
D. A subclass should never depend on its base class

552. How are the 23 Gang of Four patterns divided across the three categories? Understanding

A. 7 creational, 5 structural, 11 behavioural
B. 8 creational, 8 structural, 7 behavioural
C. 5 creational, 7 structural, 11 behavioural
D. 5 creational, 11 structural, 7 behavioural

553. According to the GoF, what is the intent of the Factory Method pattern? Understanding

A. Define an interface for creating an object, but let subclasses decide which class to instantiate
B. Ensure a class has only one instance
C. Create families of related objects without naming their concrete classes
D. Build a complex object step by step

554. According to the GoF, what is the intent of the Abstract Factory pattern? Understanding

A. Let one subclass decide which single class to instantiate
B. Provide an interface for creating families of related or dependent objects without specifying their concrete classes
C. Create new objects by copying a prototypical instance
D. Provide a simplified interface to a complex subsystem

555. According to the GoF, what is the intent of the Builder pattern? Understanding

A. Attach responsibilities to an object dynamically
B. Separate the construction of a complex object from its representation, so the same process can create different representations
C. Define a one-to-many dependency between objects
D. Ensure only one object of a class can be built

556. According to the GoF, what is the intent of the Proxy pattern? Understanding

A. Convert one interface into another that clients expect
B. Provide a surrogate or placeholder for another object to control access to it
C. Compose objects into tree structures
D. Let an object change its behaviour when its state changes

557. According to the GoF, what is the intent of the Template Method pattern? Understanding

A. Define the skeleton of an algorithm in an operation, deferring some steps to subclasses
B. Define a family of algorithms and make them interchangeable at run time
C. Encapsulate a request as an object
D. Provide a way to access elements of a collection sequentially

558. Besides purpose, the GoF classify patterns by scope. What distinguishes class-scope patterns such as Factory Method and Template Method? Understanding

A. They use inheritance between classes and subclasses, fixed at compile time
B. They deal with object relationships that can change at run time
C. They can only be used with static methods
D. They apply to exactly one class in a system

559. In a UML class diagram, what does a # in front of an attribute mean? Understanding

A. The attribute is private
B. The attribute is public
C. The attribute is static
D. The attribute is protected

560. In a UML class diagram, how is a class realising (implementing) an interface drawn? Understanding

A. A solid line with a hollow triangle pointing to the interface
B. A solid line with a filled diamond at the interface end
C. A dashed line with an open arrowhead pointing to the class
D. A dashed line with a hollow triangle pointing to the interface

561. In a structure chart, what are a module's fan-in and fan-out? Understanding

A. Fan-in is the number of parameters it receives; fan-out is the number of values it returns
B. Fan-in is the number of modules that call it; fan-out is the number of modules it calls
C. Fan-in is its lines of code; fan-out is its number of comments
D. Fan-in is the number of modules it calls; fan-out is the number that call it

562. In structured design, when is transaction analysis used instead of transform analysis to derive a structure chart? Understanding

A. When data flows in a single line from input, through a central transform, to output
B. When the system has no input data
C. When one input triggers one of several different processing paths, chosen at a dispatch (transaction) centre
D. When the design is object-oriented rather than structured

563. What does this Python code print? Application

```
class Subject:
    def __init__(self):
        self.observers = []
    def attach(self, o):
        self.observers.append(o)
    def detach(self, o):
        self.observers.remove(o)
    def notify(self, msg):
        for o in self.observers:
            o.update(msg)

class Display:
    def __init__(self):
        self.seen = []
    def update(self, msg):
        self.seen.append(msg)

s = Subject()
a, b = Display(), Display()
s.attach(a)
s.attach(b)
s.notify(1)
s.detach(a)
s.notify(2)
s.notify(3)
print(len(a.seen), len(b.seen))
```

A. 3 3
B. 1 3
C. 1 1
D. 0 3

564. What does this Python code print? Application

```
class Percent:
    def __init__(self, p):
        self.p = p
    def apply(self, total):
        return total - total * self.p // 100

class Flat:
    def __init__(self, amt):
        self.amt = amt
    def apply(self, total):
        return max(0, total - self.amt)

class Cart:
    def __init__(self, strategy):
        self.strategy = strategy
    def checkout(self, total):
        return self.strategy.apply(total)

c = Cart(Percent(10))
x = c.checkout(500)
c.strategy = Flat(600)
y = c.checkout(500)
print(x, y)
```

A. 450 0
B. 450 -100
C. 500 0
D. 50 0

565. What does this Python code print? Application

```
class Importer:
    def run(self):
        return self.read() + "|" + self.clean() + "|" + self.save()
    def read(self):
        return "read"
    def clean(self):
        return "trim"
    def save(self):
        return "db"

class CsvImporter(Importer):
    def read(self):
        return "csv"

class SafeCsvImporter(CsvImporter):
    def clean(self):
        return "trim+dedupe"

print(SafeCsvImporter().run())
```

A. read|trim|db
B. csv|trim|db
C. csv|trim+dedupe|db
D. read|trim+dedupe|db

566. What does this Python code print? Application

```
class File:
    def __init__(self, size):
        self._size = size
    def size(self):
        return self._size

class Folder:
    def __init__(self, *items):
        self.items = list(items)
    def size(self):
        return sum(i.size() for i in self.items)

docs = Folder(File(10), File(20))
photos = Folder(File(100), Folder(File(5), File(5)))
root = Folder(docs, photos, File(1))
photos.items.append(File(50))
print(root.size(), len(root.items))
```

A. 191 3
B. 141 3
C. 191 6
D. 141 2

567. What does this Python code print? Application

```
class Locked:
    def coin(self, t):
        t.state = Unlocked()
        return "unlock"
    def push(self, t):
        return "blocked"

class Unlocked:
    def coin(self, t):
        return "refund"
    def push(self, t):
        t.state = Locked()
        return "pass"

class Turnstile:
    def __init__(self):
        self.state = Locked()
    def coin(self):
        return self.state.coin(self)
    def push(self):
        return self.state.push(self)

t = Turnstile()
log = [t.push(), t.coin(), t.coin(), t.push(), t.push()]
print(" ".join(log))
```

A. blocked unlock refund pass blocked
B. blocked unlock unlock pass pass
C. pass unlock refund pass blocked
D. blocked unlock refund pass pass

568. What does this Python code print? Application

```
class RealImage:
    loads = 0
    def __init__(self, name):
        RealImage.loads += 1
        self.name = name
    def show(self):
        return "show " + self.name

class ImageProxy:
    def __init__(self, name):
        self.name = name
        self.real = None
    def show(self):
        if self.real is None:
            self.real = RealImage(self.name)
        return self.real.show()

gallery = [ImageProxy("a"), ImageProxy("b"), ImageProxy("c")]
gallery[0].show()
gallery[0].show()
gallery[2].show()
print(RealImage.loads)
```

A. 2
B. 3
C. 0
D. 1

569. What does this Python code print? Application

```
class Add:
    def __init__(self, n):
        self.n = n
    def do(self, acc):
        return acc + self.n
    def undo(self, acc):
        return acc - self.n

class Mul:
    def __init__(self, n):
        self.n = n
    def do(self, acc):
        return acc * self.n
    def undo(self, acc):
        return acc // self.n

acc, history = 1, []
for cmd in [Add(4), Mul(3), Add(2)]:
    acc = cmd.do(acc)
    history.append(cmd)
acc = history.pop().undo(acc)
acc = history.pop().undo(acc)
print(acc, len(history))
```

A. 15 1
B. 5 2
C. 17 3
D. 5 1

570. What does this Python code print? Application

```
class Approver:
    def __init__(self, name, limit, nxt=None):
        self.name, self.limit, self.nxt = name, limit, nxt
    def handle(self, amount):
        if amount <= self.limit:
            return self.name
        if self.nxt:
            return self.nxt.handle(amount)
        return "board"

chain = Approver("lead", 1000,
                 Approver("manager", 5000,
                          Approver("director", 20000)))
print([chain.handle(a) for a in (800, 5000, 7500, 25000)])
```

A. ['lead', 'director', 'director', 'board']
B. ['lead', 'manager', 'director', 'board']
C. ['lead', 'manager', 'director', 'director']
D. ['manager', 'manager', 'director', 'board']

571. A function print_document(kind) contains separate code to print an invoice, a payslip or a transcript, and the kind argument picks which block runs. What kind of cohesion does it have? Application

A. Functional cohesion
B. Temporal cohesion
C. Sequential cohesion
D. Logical cohesion

572. A module reads raw exam marks, computes each student's GPA from those marks, and then assigns a grade from the GPA. What kind of cohesion is this? Application

A. Coincidental cohesion
B. Temporal cohesion
C. Sequential cohesion
D. Logical cohesion

573. A routine save_and_exit() first checks whether the printer is online, then saves the open document, then plays a closing sound. The three steps use different data but must happen in this order. What kind of cohesion does it have? Application

A. Sequential cohesion
B. Procedural cohesion
C. Communicational cohesion
D. Functional cohesion

574. Class Billing directly changes the private field _balance of class Account, bypassing Account's methods. What type of coupling is this? Application

A. Content coupling
B. Data coupling
C. Stamp coupling
D. Control coupling

575. Two modules both read and write files in a bank's fixed-width statement format, which the bank can change at any time. What type of coupling links them? Application

A. Data coupling
B. Content coupling
C. Stamp coupling
D. External coupling

576. A GUI library must support Light and Dark themes. Each theme needs its own Button, Checkbox and Menu, and a screen must never mix Light and Dark widgets. Which GoF pattern fits best? Application

A. Singleton
B. Abstract Factory
C. Decorator
D. Observer

577. An HTTP request object has a URL plus many optional parts: headers, query parameters, a body, a timeout and retries. You want readable code that sets only the parts needed and then creates the object. Which GoF pattern fits best? Application

A. Adapter
B. Composite
C. Observer
D. Builder

578. A game creates enemies whose set-up is expensive (loading models and computing stats). Many enemies are almost identical, with only their position differing. Which GoF pattern fits best? Application

A. Facade
B. Strategy
C. Prototype
D. Iterator

579. At an airport, aircraft do not talk to each other directly about runway use; each talks only to the control tower, which coordinates them. Which GoF pattern does this model? Application

A. Composite
B. Prototype
C. Adapter
D. Mediator

580. A text editor displays millions of characters. Storing a full font, size and glyph image in every character object uses too much memory, but most characters share the same few styles. Which GoF pattern fits best? Application

A. Builder
B. Chain of Responsibility
C. Flyweight
D. Bridge

581. A custom LinkedPlaylist class must let client code visit every song in order without knowing whether songs are stored in nodes, an array or a tree. Which GoF pattern fits? Application

A. Singleton
B. Iterator
C. Decorator
D. Command

582. A UML class diagram shows Person 1 —— 0..1 Passport. Which statement is correct? Application

A. A person has exactly one passport, and a passport may belong to no one
B. A person has zero or more passports, and each passport belongs to one person
C. A person has at most one passport, and each passport belongs to exactly one person
D. Each passport belongs to at most one person, and each person has exactly one passport

583. In a structure chart, main calls read, process and write; read calls log; process calls validate and log; write calls log. What are the fan-in of log and the fan-out of main? Application

A. Fan-in of log = 3; fan-out of main = 3
B. Fan-in of log = 0; fan-out of main = 3
C. Fan-in of log = 3; fan-out of main = 6
D. Fan-in of log = 1; fan-out of main = 1

584. What does this Python code print, and what does it show about the Decorator pattern? Analysis

```
class Text:
    def __init__(self, s):
        self.s = s
    def render(self):
        return self.s

class Bold:
    def __init__(self, inner):
        self.inner = inner
    def render(self):
        return "<b>" + self.inner.render() + "</b>"

class Upper:
    def __init__(self, inner):
        self.inner = inner
    def render(self):
        return self.inner.render().upper()

print(Bold(Upper(Text("hi"))).render(), Upper(Bold(Text("hi"))).render())
```

A. HI HI; decorators always produce the same result whatever the order
B. <B>HI</B> HI; the outer decorator always runs first
C. HI <B>HI</B>; the order in which decorators wrap an object changes the result
D. hi <B>HI</B>; Upper is ignored when it is inside Bold

585. What does this Python code print, and which design principle does Square break? Analysis

```
class Rectangle:
    def __init__(self, w, h):
        self.w, self.h = w, h
    def set_width(self, w):
        self.w = w
    def set_height(self, h):
        self.h = h
    def area(self):
        return self.w * self.h

class Square(Rectangle):
    def __init__(self, s):
        super().__init__(s, s)
    def set_width(self, w):
        self.w = self.h = w
    def set_height(self, h):
        self.w = self.h = h

def stretch(r):
    r.set_width(5)
    r.set_height(4)
    return r.area()

print(stretch(Rectangle(1, 1)), stretch(Square(1)))
```

A. 20 16; the Liskov substitution principle
B. 20 20; no principle is broken
C. 20 16; the open/closed principle
D. 16 16; the interface segregation principle

586. A function area(shape) uses if/elif on shape.kind to handle "circle", "square" and "triangle". Every new shape means editing this function. Which principle is broken, and what is the best fix? Analysis

A. Single responsibility; split area() into three functions with the same if/elif
B. Open/closed; give each shape class its own area() method and call shape.area() polymorphically
C. Interface segregation; add an empty area() to an unrelated interface
D. Liskov substitution; make every shape inherit from Circle

587. class OrderService creates self.db = MySQLDatabase() inside its constructor and calls it directly. Unit tests need a real MySQL server, and moving to PostgreSQL means editing OrderService. What is the best analysis? Analysis

A. It breaks Liskov substitution; MySQLDatabase must subclass OrderService
B. It is a good design, because concrete classes are faster than interfaces
C. It breaks interface segregation; MySQLDatabase has too few methods
D. It breaks dependency inversion; depend on a Repository interface and inject the database

588. An interface Worker declares work(), eat() and sleep(). A Robot class must implement it, so it writes eat() and sleep() methods that just raise errors. What is the best redesign? Analysis

A. Keep Worker and have Robot return None from eat() and sleep()
B. Split Worker into smaller interfaces such as Workable and Feedable, and let Robot implement only Workable
C. Make Robot a subclass of a Human class
D. Merge every interface in the system into one so all classes match

589. Code in InvoicePrinter calls order.get_customer().get_address().get_city(). What is the main design concern? Analysis

    A. It breaks the open/closed principle because getters cannot be overridden
    B. It shows high cohesion, because all the calls are on one line
    C. It is content coupling, because it calls public methods
    D. It breaks the Law of Demeter: InvoicePrinter now depends on Customer and Address internals as well as Order

590. Design A: a document's publish() behaviour changes automatically as it moves through Draft, Review and Published, and each state object decides the next state. Design B: the user picks a compression algorithm (ZIP, RAR, 7z) before saving. Which patterns are these? Analysis

    A. A is Strategy; B is State
    B. A is State; B is Strategy
    C. Both are Strategy, because each swaps one object for another
    D. A is Observer; B is Template Method

591. A class SecureReport has exactly the same interface as Report. Its methods check the user's role and only then forward the call to a real Report object. Which pattern is this? Analysis

    A. Adapter
    B. Facade
    C. Proxy (protection proxy)
    D. Composite

592. Module P runs unrelated set-up tasks at start-up. Module Q parses input, validates the parsed data and stores the validated result. Module R holds random helpers with nothing in common. Order them from strongest to weakest cohesion. Analysis

    A. P, Q, R
    B. R, P, Q
    C. Q, P, R
    D. Q, R, P

593. A function send_reminder(student) receives a whole Student record with 30 fields but only uses student.email and student.name. Which change most reduces the coupling? Analysis

    A. Make Student a global variable that send_reminder reads
    B. Add a flag parameter to choose between email and SMS
    C. Change it to send_reminder(email, name), passing only the values it needs
    D. Let send_reminder edit Student's private fields directly

594. A form has 6 widgets, and every widget talks directly to every other. A redesign routes all interaction through one FormMediator. How many communication links are there before and after? Analysis

    A. 30 before, 6 after
    B. 15 before, 1 after
    C. 15 before, 6 after
    D. 6 before, 15 after

595. A Duck base class has a fly() method. RubberDuck and WoodenDuck subclasses must override fly() to do nothing, and a new RocketDuck flies in yet another way. What is the best redesign? Analysis

    A. Add more levels of inheritance until every combination has its own subclass
    B. Move fly() into a global function that checks the duck's class name
    C. Make Duck a Singleton so only one duck can fly
    D. Favour composition: give Duck a FlyBehaviour object (Strategy) that each duck is configured with

596. An app attaches each newly opened ChartWindow as an observer of a PriceFeed. Windows are closed but never detached. Over a long session memory grows and closed windows still receive updates. What is the cause and fix? Analysis

A. The subject still holds references to closed observers; detach them on close or use weak references
B. Observer always leaks memory, so replace it with a Singleton
C. PriceFeed is notifying too slowly; notify more often
D. ChartWindow has low cohesion; merge it into PriceFeed

597. In a structure chart, module format_date has a fan-in of 25, and module run_everything has a fan-out of 25. What is the best assessment? Analysis

A. Both are equally bad, because any high fan value means tight coupling
B. format_date should be copied into each caller to reduce its fan-in
C. format_date's high fan-in shows useful reuse; run_everything's high fan-out suggests it controls too much and should be split
D. run_everything is ideal, because high fan-out means high cohesion

598. The same 12-line input-validation block is copied into four classes. A bug is found in it. What is the best design response? Analysis

A. Fix the bug in the class where it was reported and leave the other three
B. Extract the block into one shared method or class and call it from all four places
C. Add a comment in each copy reminding developers to keep them in sync
D. Move all four classes into one large class

Software Architecture MCQs for NSCT

599. Which statement best matches the definition of software architecture given by Bass, Clements and Kazman? Understanding

A. The detailed algorithm and data-structure design inside each class
B. The set of structures needed to reason about a system: software elements, the relations among them and the properties of both
C. The choice of programming language, IDE and build tool for a project
D. The physical layout of servers and network cables in a data centre

600. In a strict (closed) layered architecture, which layer may a given layer call? Understanding

A. Only the layer directly below it
B. Any layer below it
C. Any layer above or below it
D. Only the layer directly above it

601. Which statement is true of the client-server architectural style? Understanding

A. Every node has an equal role and acts as both client and server
B. Clients initiate requests and servers provide services in response
C. Servers must know the identity of every client before it connects
D. Clients communicate only by writing to a shared central data store

602. In the Model-View-Controller (MVC) pattern, which component receives user input and turns it into operations on the model? Understanding

A. View
B. Controller
C. Model
D. Repository

603. Which property is central to a microservices architecture? Understanding

A. All services share one database schema to keep data consistent
B. Each service can be deployed independently and owns its own data
C. The whole application is built and released as a single unit
D. Every service must be written in the same programming language

604. In Sommerville's repository architecture, how do the sub-systems mainly exchange data? Understanding

A. By sending events to subscribers they do not know
B. By passing data streams from one filter to the next
C. Through a shared central repository that all components read and write
D. By calling the layer immediately below them

605. In Kruchten's 4+1 view model, which view shows how software components are mapped onto hardware nodes? Understanding

A. Logical view
B. Process view
C. Development view
D. Physical view

606. In Kruchten's 4+1 view model, what does the "+1" refer to? Understanding

A. A security view added for networked systems
B. A data view describing the database schema
C. Scenarios (use cases) that tie the other four views together
D. The source code that implements the architecture

607. Which of these is an architectural pattern rather than an object-level design pattern from the Gang of Four catalogue? Understanding

A. Singleton
B. Pipe and filter
C. Adapter
D. Factory Method

608. Bass, Clements and Kazman call properties such as availability, modifiability, performance and security: Understanding

A. Quality attributes
B. Functional requirements
C. Architectural views
D. Deployment constraints

609. A heartbeat, where one component sends a periodic "I am alive" message to a monitor, is a tactic for which quality attribute? Understanding

A. Modifiability
B. Usability
C. Testability
D. Availability

610. What is the main benefit of the publish-subscribe (event-driven) style? Understanding

A. Every event is guaranteed to be processed in the same order on every machine
B. Publishers and subscribers are loosely coupled because publishers do not know who receives their events
C. Components call each other directly, so control flow is easy to trace
D. It removes the need for any middleware or message broker

611. In the Architecture Tradeoff Analysis Method (ATAM), what is a tradeoff point? Understanding

A. A decision that is critical for achieving one particular quality attribute
B. An architectural decision that affects two or more quality attributes in opposite directions
C. A decision whose consequences for the quality goals are not yet understood
D. The point in the schedule when the architecture is frozen

612. In a two-tier fat-client system, where does the application (business) logic run? Understanding

A. Entirely on the server, with the client only displaying screens
B. On a separate middle tier between the client and the database
C. Inside the database's stored procedures only
D. Mostly on the client machine, with the server mainly managing data

613. In classic MVC, how does a view usually find out that the model's data has changed? Understanding

A. The model notifies registered views, as in the Observer pattern
B. The view polls the database directly on a timer
C. The controller copies the new data into the view's fields
D. The model redraws the view itself

614. What does an API gateway do in a microservices architecture? Understanding

A. It is a shared database that every service writes its data to
B. It is a single entry point that routes client requests to the right services and can handle concerns such as authentication
C. It compiles all services into one deployable package
D. It replaces each service's internal business logic

615. A log-processing script runs `cat app.log | grep ERROR | sort | uniq -c`. Which architectural style does this command line follow? Application

A. Client-server
B. Repository
C. Pipe and filter
D. Model-View-Controller

616. An online banking system has a browser front end, an application server holding the business rules and a separate database server. How is this best described? Application

A. Peer-to-peer architecture
B. Pipe-and-filter architecture
C. Two-tier fat-client architecture
D. Three-tier client-server architecture

617. The TCP/IP protocol stack, where each level uses only the services of the level beneath it, is a classic example of which style? Application

A. Microservices
B. Repository
C. Layered architecture
D. Publish-subscribe

618. Temperature sensors in a factory send readings to named topics. A dashboard, an alarm service and an archive service each choose which topics they receive, and sensors do not know who is listening. Which style is this? Application

A. Publish-subscribe (event-driven)
B. Client-server
C. Pipe and filter
D. Strict layered

619. In an IDE, the editor, compiler, debugger and refactoring tool all read and update one shared abstract syntax tree and symbol table. Which style is this? Application

A. Repository
B. Peer-to-peer
C. Pipe and filter
D. Microservices

620. A Django app maps a URL to a function that loads `Course` objects and renders `course_list.html`. In classic MVC terms, which component is the HTML template? Application

A. Model
B. Controller
C. Router
D. View

621. An e-commerce firm's checkout must scale tenfold during sales, while the product catalogue rarely changes. Separate teams own checkout and catalogue and want to release on their own schedules. Which architecture fits best? Application

A. A single monolith scaled by running more copies of the whole application
B. A repository architecture with one shared database for all modules
C. Microservices, with checkout and catalogue as separate services
D. A pipe-and-filter architecture processing orders in batches

622. A three-person start-up must launch a minimum viable product in eight weeks. The domain is still unclear and traffic will be small at first. Which architecture is the most sensible starting point? Application

A. Twenty microservices, each with its own database
B. A peer-to-peer network of client nodes
C. A well-layered (modular) monolith
D. A pipe-and-filter batch system

623. A strict layered system has layers UI → Business Logic → Data Access → Database. The UI needs a customer record. Which call path follows the architecture? Application

A. UI → Database directly with an SQL query
B. UI → Data Access → Database, skipping Business Logic
C. UI → Business Logic → Data Access → Database
D. Database → Data Access → UI

624. In a file-sharing network, every computer downloads pieces of a file from others and also uploads pieces it already has. No central server stores the file. Which style is this? Application

A. Peer-to-peer
B. Client-server
C. Three-tier
D. Model-View-Controller

625. A company's 500 desktop PCs run a fat-client app containing all the pricing rules, and every rule change means reinstalling software on each PC. Which change best addresses this? Application

A. Add more RAM to each desktop PC
B. Move the business rules to a server tier so clients only handle presentation
C. Duplicate the pricing rules into the database and the client
D. Merge the client and database into one desktop application

626. In an Orders microservice you need the customer's name, which belongs to the Customers service. Each service has its own database. Which approach respects the database-per-service rule? Application

A. Join directly against the Customers service's tables from the Orders database
B. Give the Orders service write access to the Customers database
C. Get it through the Customers service's API, or keep a local copy updated from its events
D. Merge the two services into one database schema

627. A server has a mean time between failures (MTBF) of 990 hours and a mean time to repair (MTTR) of 10 hours. What is its steady-state availability? Application

A. 98%
B. 99%
C. 99.9%
D. 1.01%

628. A request must pass through two services in series. Each is independently available 99.9% of the time. What is the availability of the request path? Application

A. 99.9%
B. 99.99%
C. About 99.95%
D. About 99.80%

629. A team says it uses microservices, but all 12 services read and write one shared database schema, and any schema change forces all of them to be redeployed together. What is the main architectural problem? Analysis

A. There are too few services; splitting into 24 would fix it
B. The services should be written in different programming languages
C. The system needs a pipe-and-filter layer in front of the database
D. The services are coupled through the shared schema, so they have lost independent deployment (a distributed monolith)

630. A five-layer system is easy to change, but each request passes through every layer and response time is poor. Which architectural trade-off does this show? Analysis

A. Layering improves modifiability and portability at some cost in performance
B. Layering improves performance but makes changes harder
C. Layering always improves every quality attribute together
D. The poor response time proves the system is not really layered

631. Following Sommerville's guidance, security is the most critical requirement for a banking core. Which architectural decision best supports it? Analysis

A. A layered structure with the most critical assets in the innermost layers and validation at each layer
B. Large-grain components on one machine to cut communication between them
C. Fine-grained, replaceable components spread across many servers
D. Replicating all customer data to as many nodes as possible

632. Performance is the critical requirement for a trading engine that must respond within microseconds. Following Sommerville, which decision fits best? Analysis

A. Split each operation into a separate microservice behind an API gateway
B. Add extra layers so each concern can be optimised separately
C. Localise critical operations in a few large-grain components deployed on the same computer
D. Store all data in a shared repository accessed over the network

633. In an MVC web app, each controller method contains SQL queries, discount calculations and validation rules, and the same rules are copied across several controllers. What is the best fix? Analysis

A. Move the SQL and rules into the HTML templates (views)
B. Merge all controllers into one large controller
C. Move business rules and data access into the model so controllers only coordinate input and output
D. Remove the model layer since the controllers already contain the logic

634. In an event-driven payment system, a consumer crashes after charging a card but before acknowledging the event. The broker redelivers it and the customer is charged twice. What is the best fix? Analysis

A. Make the consumer idempotent, for example by recording processed event IDs and ignoring repeats
B. Replace the broker with direct synchronous calls between services
C. Add more consumer instances to process events faster
D. Increase how long the broker keeps events before deleting them

635. A CASE tool built on the repository pattern works well, but the team now needs to change the shared data model, and every tool must be modified and retested. Which weakness of the repository style does this show? Analysis

A. Components communicate too much directly with each other
B. The repository makes data sharing between tools impossible
C. Repository systems cannot store large amounts of data
D. All components depend on the repository's data model, so changing it affects every component

636. A mobile app screen makes 30 separate calls to backend microservices, and on a 200 ms mobile network it takes several seconds to load. Which change gives the biggest improvement? Analysis

A. Add an API gateway or backend-for-frontend that aggregates the calls into one request
B. Split the backend into even smaller services
C. Move every service's database onto the mobile device
D. Make each call use a slower but more reliable protocol

637. Why is pipe and filter a poor choice for the core of an interactive drawing editor? Analysis

A. Filters cannot be reused in other programs
B. It needs a central repository that drawing tools cannot access
C. It forces every component into a strict hierarchy of layers
D. It suits batch transformation of data streams, not systems that react to frequent user events

638. Service A calls service B synchronously. B becomes very slow, A's threads all wait on B, and soon A stops responding to every request, including those that do not need B. Which tactic best prevents this cascade? Analysis

A. Raising A's timeout to wait longer for B
B. Merging A and B into one shared database
C. Adding more logging inside B
D. A circuit breaker with timeouts that fails fast when B is unhealthy

639. In a relaxed (open) layered architecture, what may a layer do that a strict (closed) layered architecture forbids? Understanding

A. Call the layer directly above it
B. Call any layer below it, not only the one directly beneath
C. Share global variables with every other layer
D. Skip the interface and edit another layer's data structures

640. In ATAM, what is a sensitivity point? Understanding

A. A decision that improves one quality attribute while harming another
B. A scenario that stakeholders rank as the highest priority
C. A property of one or more components that is critical for achieving a particular quality attribute response
D. A component that has not yet been assigned to any architectural view

641. In Kruchten's 4+1 view model, which view shows how the software is organised into modules, packages and libraries for the programmers who build it? Understanding

A. Physical view
B. Process view
C. Development view
D. Scenarios

642. Which concerns does the process view of Kruchten's 4+1 model mainly address? Understanding

A. The mapping of components to physical servers and network links
B. The class structure that implements functional requirements
C. The order in which the development team delivers releases
D. Concurrency, distribution and runtime interaction between processes and threads

643. In Kruchten's 4+1 model, which view mainly supports the functional requirements, showing key abstractions such as classes and their relationships? Understanding

A. Development view
B. Physical view
C. Process view
D. Logical view

644. In a two-tier thin-client model, what does the client mainly do? Understanding

A. Application logic and data management, with the server only storing files
B. Data management only, while the server handles presentation
C. Nothing until it downloads the full database to run locally
D. Presentation only, while application logic and data management run on the server

645. In the broker architectural pattern, as used by middleware such as CORBA, what does the broker do? Understanding

A. It stores all shared data in a central repository that components read directly
B. It locates servers and forwards client requests and replies, so clients need not know where services run
C. It renders the user interface and passes events to a controller
D. It splits the data into a stream that passes through a chain of filters

646. Bass, Clements and Kazman define an architectural tactic as: Understanding

A. A design decision that influences how a system responds to a quality attribute stimulus
B. A complete architectural style, such as layered or pipe and filter
C. A project management technique for scheduling architecture work
D. A test case that checks a functional requirement

647. A quality attribute scenario in Bass, Clements and Kazman has six parts. Which item is NOT one of them? Understanding

A. Cost estimate
B. Stimulus
C. Environment
D. Response measure

648. Which of these is a modifiability tactic? Understanding

A. Sending periodic heartbeat messages to a monitor
B. Authenticating every user before access
C. Placing an intermediary or interface between modules to reduce coupling
D. Keeping a hot spare server running in parallel

649. Which of these is a performance tactic? Understanding

A. Splitting a module to increase cohesion
B. Recording an audit trail of user actions
C. Using a voting scheme among redundant components
D. Keeping multiple copies of data, such as a cache, to cut response time

650. Requiring users to log in with a password and a one-time code before reaching account data is a tactic for which quality attribute? Understanding

A. Availability
B. Modifiability
C. Security
D. Performance

651. In a microservices system, service instances start and stop as load changes, so their network addresses keep changing. Which mechanism lets clients find a current instance? Understanding

A. A shared database schema for all services
B. Service discovery through a service registry
C. Hard-coding each instance's IP address in the client
D. A pipe-and-filter chain between services

652. In microservices, how does the saga pattern keep data consistent across services that each own a database? Understanding

A. It locks every service's database with a two-phase commit for the whole business process
B. It moves all services back onto one shared schema
C. It runs a sequence of local transactions, and if one step fails it runs compensating transactions to undo the earlier steps
D. It retries the failed step forever until it succeeds

653. What does Conway's law state? Understanding

A. The number of defects grows with the square of the number of modules
B. Adding people to a late project makes it later
C. Organisations design systems whose structure mirrors their own communication structure
D. Every layer added to a system doubles its response time

654. What is the strangler fig pattern? Understanding

A. Rewriting the whole legacy system from scratch and switching over in one release
B. Wrapping every new service inside the legacy monolith's codebase
C. Gradually replacing parts of a legacy system with new services behind a routing layer until the old system can be removed
D. Deleting unused legacy features without replacing them

655. Sommerville lists three advantages of designing and documenting an explicit software architecture. Which set is correct? Understanding

A. Stakeholder communication, system analysis and large-scale reuse
B. Faster compilation, smaller binaries and fewer test cases
C. Lower hardware cost, automatic scalability and zero downtime
D. Code generation, version control and continuous integration

656. Sommerville distinguishes architecture in the small from architecture in the large. What does architecture in the large concern? Understanding

A. How an individual program is decomposed into components
B. The internal design of the classes inside one component
C. Complex enterprise systems that include other systems, programs and components, often distributed across computers and owned by different organisations
D. The layout of user interface screens

657. According to Sommerville, what is a disadvantage of the MVC pattern? Understanding

A. It can add extra code and complexity when the data model and interactions are simple
B. It does not allow more than one view of the same data
C. It forces the user interface and data to be stored in the same class
D. It cannot be used for web applications

658. Which is a recognised weakness of the pipe-and-filter style? Understanding

A. Filters cannot be reused in a different pipeline
B. Filters must agree a data format, and each filter may have to parse and unparse data, which adds overhead
C. New filters cannot be added without rewriting the existing ones
D. It only works when all filters run on the same thread

659. According to Sommerville, what is a disadvantage of the client-server model? Understanding

A. Clients and servers must always run on the same computer
B. New servers cannot be added once the system is deployed
C. Each service can be a single point of failure, and performance depends on the network as well as the server
D. Clients must know the internal data structures of every server

660. What is horizontal scaling? Understanding

A. Adding more CPU and memory to one existing machine
B. Moving code from the client tier to the server tier
C. Splitting one database table into more columns
D. Adding more machines or instances and spreading the load across them

661. Why are stateless services easier to scale horizontally? Understanding

A. They never need a database
B. They run faster on a single CPU
C. They do not need a network connection
D. Any instance can handle any request because no session data is stored inside a particular instance

662. What does CQRS (Command Query Responsibility Segregation) separate? Understanding

A. The client tier from the server tier
B. The development view from the physical view
C. Synchronous calls from asynchronous messages
D. The model that handles updates (commands) from the model that handles reads (queries)

663. A service runs on two independent servers in parallel; the service is up if at least one server is up. Each server is available 99% of the time. What is the service's availability? Application

A. 99%
B. 98.01%
C. 99.5%
D. 99.99%

664. A system must reach 99.9% availability, and its mean time to repair (MTTR) is 1 hour. What mean time between failures (MTBF) is needed? Application

A. 999 hours
B. 99.9 hours
C. 1,000 hours exactly
D. 9,990 hours

665. A service promises 99.9% availability over a 365-day year. About how much downtime does that allow? Application

A. About 52.6 minutes
B. About 3.65 days
C. About 87.6 hours
D. About 8.76 hours

666. A checkout request must pass through three services in series, each independently available 99.5% of the time. What is the availability of the request path, to two decimal places? Application

A. 99.50%
B. 98.51%
C. 98.50%
D. 99.83%

667. In a strict layered design UI → Business Logic → Data Access → Database, a developer lets the UI call Data Access directly for one fast report. What has the design become? Application

A. A relaxed (open) layered architecture
B. A pipe-and-filter architecture
C. A peer-to-peer architecture
D. A repository architecture

668. An architect draws a diagram showing the order-processing threads, the message queue between them and how they synchronise at runtime. Which 4+1 view is this? Application

A. Logical view
B. Process view
C. Development view
D. Physical view

669. A team needs to move from MySQL to PostgreSQL. In their layered system all SQL lives in the Data Access layer behind an interface. Where must changes mainly be made? Application

A. In every layer, because they all use the database
B. In the UI layer, because it displays the data
C. In the Data Access layer only
D. In the Business Logic layer, because it owns the rules

670. A spreadsheet shows the same sales figures as a table and as a bar chart. Editing a cell instantly updates the chart. Which pattern does this best illustrate? Application

A. MVC, with one model and several views
B. Pipe and filter
C. Peer-to-peer
D. Broker

671. When an order is placed, the email, inventory and loyalty-points services must all react, and new reacting services will be added later without changing the order service. Which style fits? Application

A. Strict layering, with the order service calling each service in turn
B. A repository that every service polls for new orders
C. A thin-client model with all logic on the server
D. Publish-subscribe, with the order service publishing an OrderPlaced event

672. Quality scenario: "When a server crashes during normal operation, the monitor detects it and the system resumes service within 30 seconds." Which part of the scenario is "within 30 seconds"? Application

A. Stimulus
B. Response measure
C. Environment
D. Artifact

673. A product catalogue page runs the same costly database query for every visitor, though prices change only once a day. Which tactic most directly improves response time? Application

A. Cache the query result and refresh it when prices change
B. Add a heartbeat between the web server and the database
C. Split the catalogue into more layers
D. Require users to authenticate before viewing the catalogue

674. In a microservices shop, the Order service creates an order, then the Payment service declines the card. Each service has its own database. Using the saga pattern, what should happen next? Application

A. Run a compensating transaction that cancels the order in the Order service
B. Roll back both databases with a single local transaction
C. Leave the order as placed and retry payment forever
D. Delete the Payment service's database records for the customer

675. What does this pipe-and-filter style program print? Application

```
def source():
    for x in [3, 8, 5, 12, 7]:
        yield x

def keep_odd(xs):
    for x in xs:
        if x % 2:
            yield x

def square(xs):
    for x in xs:
        yield x * x

print(list(square(keep_odd(source()))))
```

A. [9, 25, 49]
B. [9, 64, 25, 144, 49]
C. [64, 144]
D. [3, 5, 7]

676. This code uses the observer link from classic MVC. What does it print? Application

```
class Model:
    def __init__(self):
        self.views = []
        self.value = 0
    def attach(self, v):
        self.views.append(v)
    def set(self, value):
        self.value = value
        for v in self.views:
            v.update(self)

class TextView:
    def update(self, m):
        print("text", m.value)

class BarView:
    def update(self, m):
        print("bar", "#" * m.value)

m = Model()
m.attach(TextView())
m.attach(BarView())
m.set(3)
```

A. bar ### text 3
B. text 3 bar ###
C. text 0 bar
D. text 3

677. This code is a minimal publish-subscribe event bus. What does it print? Application

```
subs = {}

def subscribe(topic, fn):
    subs.setdefault(topic, []).append(fn)

def publish(topic, msg):
    for fn in subs.get(topic, []):
        fn(msg)

log = []
subscribe("order", lambda m: log.append("mail:" + m))
subscribe("order", lambda m: log.append("stock:" + m))
subscribe("payment", lambda m: log.append("bank:" + m))
publish("order", "A1")
publish("refund", "A1")
print(log)
```

A. ['mail:A1', 'stock:A1', 'bank:A1']
B. ['mail:A1']
C. ['stock:A1', 'mail:A1']
D. ['mail:A1', 'stock:A1']

678. This code sketches a circuit breaker. What does it print? Application

```
class Breaker:
    def __init__(self, limit):
        self.limit, self.fails, self.state = limit, 0, "CLOSED"
    def call(self, ok):
        if self.state == "OPEN":
            return "fast-fail"
        if ok:
            self.fails = 0
            return "ok"
        self.fails += 1
        if self.fails >= self.limit:
            self.state = "OPEN"
        return "error"

b = Breaker(2)
print([b.call(ok) for ok in [False, True, False, False, True]])
```

A. ['error', 'ok', 'error', 'fast-fail', 'fast-fail']
B. ['error', 'ok', 'error', 'error', 'fast-fail']
C. ['error', 'fast-fail', 'fast-fail', 'fast-fail', 'fast-fail']
D. ['error', 'ok', 'error', 'error', 'ok']

679. A round-robin load balancer sends 10 requests to servers A, B and C, starting with A. How many requests does A receive? Application

A. 3
B. 5
C. 4
D. 10

680. A bank wants to move off a 15-year-old monolith without a risky one-day switchover. New features should go live in new services while old screens keep working. Which approach fits? Application

A. The strangler fig pattern, routing more features to new services over time
B. A big-bang rewrite released in one weekend
C. Adding all new features inside the old monolith
D. Converting the monolith to a pipe-and-filter design

681. A company has three teams: front-end, back-end and database. It plans a product with five business capabilities. What does Conway's law predict about the resulting architecture? Application

A. It will produce exactly five independent microservices
B. It will tend to split into three technical layers matching the teams, not five capability-based services
C. It will become peer-to-peer because all teams are equal
D. Team structure will have no effect on the architecture

682. A ride-hailing app runs its Pricing service on containers that are added and removed every few minutes. The Booking service keeps failing because it calls stale IP addresses. What should be added? Application

A. A service registry so Booking looks up live Pricing instances (service discovery)
B. A larger fixed list of IP addresses in Booking's config file
C. A shared database between Booking and Pricing
D. A strict layered structure inside the Pricing service

683. A primary server processes requests while a standby receives every state update and can take over within milliseconds. In Bass's availability tactics, what is this? Application

A. Passive redundancy (warm spare)
B. Cold spare
C. Active redundancy (hot spare)
D. Heartbeat

684. In a strict four-layer system, most requests pass through the Business layer without any processing, only forwarding calls to Data Access. Each extra hop adds latency. What is the best response? Analysis

A. Add a fifth layer to separate the pass-through calls
B. Consider opening the Business layer for those pass-through requests, accepting some extra coupling for speed
C. Merge all four layers into one class
D. Replace the layers with a pipe-and-filter chain

685. In an event-driven system, a customer's order was cancelled, and developers cannot tell which of 14 subscribers triggered it, because no component calls another directly. Which weakness of the style is this, and what helps? Analysis

A. Control flow is hard to trace; adding correlation IDs and central event logging helps
B. Coupling is too tight; merging subscribers into one service fixes it
C. The style cannot scale; switching to a thin client fixes it
D. Events are too slow; replacing them with a shared database fixes it

686. One user request goes through six microservices in a synchronous chain. Latency is high, and an outage in any one of them fails the request. Which redesign best addresses both problems? Analysis

A. Add a seventh service to coordinate the other six synchronously
B. Raise every service's timeout to 60 seconds
C. Give all six services one shared database
D. Reduce the synchronous chain, for example by merging over-split services or using asynchronous events where an immediate answer is not needed

687. A team caches product prices for 24 hours to speed up pages. Now customers sometimes see an old price at checkout. What is the best analysis? Analysis

A. The cache is too small; doubling its size will fix stale prices
B. Caching has no downside, so the database must be wrong
C. Remove the database and serve prices only from the cache
D. Caching traded consistency for performance; invalidate the cache when a price changes or re-check the price at checkout

688. A set of CASE tools (editor, analyser, report generator) all work on a large shared design model. Tools are added rarely, but they constantly need the latest version of the whole model. Which style fits better: repository or publish-subscribe? Analysis

A. Publish-subscribe, because events are always faster than shared data
B. Publish-subscribe, because the model never changes
C. Neither; pipe and filter is the only option for tools
D. Repository, because the tools share a large body of data that all of them need

689. In an MVC app, a view template opens a database connection and updates a student's grade when a page is shown. What is the main problem? Analysis

A. The view should open two connections for speed
B. The view is doing model work, which breaks MVC's separation and spreads business logic into presentation
C. Templates are not allowed to display data at all
D. The controller should be removed to simplify the design

690. Encrypting every call between internal services improves security but adds measurable latency to each request. In ATAM terms, what is the decision to encrypt all internal traffic? Analysis

A. A sensitivity point only, because it affects just one attribute
B. A risk-free decision, because security always outranks performance
C. A tradeoff point, because it affects security and performance in opposite directions
D. A non-risk, because it has no effect on any quality attribute

691. A small clinic system has one application server. When it reboots for updates, every client is cut off. Which change best fixes this weakness of client-server? Analysis

A. Move all clients to a fat-client model with no server
B. Run at least two server instances behind a load balancer so one can serve while the other restarts
C. Add more client machines
D. Make the server's response messages shorter

692. Field inspectors use tablets in rural areas with no internet for hours at a time, and must still apply inspection rules and save results. Which client model suits this best? Analysis

A. A thin client that runs all logic on the server
B. A pure web page that reloads from the server on every click
C. A fat (rich) client with local logic and storage that syncs with the server when online
D. A peer-to-peer network between the inspectors' tablets

693. A monolithic photo site is fine except for image resizing, which uses 90% of CPU at peak and forces the whole app to be scaled up. What is the most targeted change? Analysis

A. Split the whole monolith into 40 microservices at once
B. Move image resizing into the browser's HTML template
C. Extract image resizing into a separate service that can be scaled on its own
D. Buy a larger single server for the whole application

694. Ten microservices each use a shared "common-models" library that holds their data classes. Any change to the library means rebuilding and releasing all ten together. What is the main problem? Analysis

A. Ten services are too many for one library to hold
B. The library should be rewritten in a faster language
C. The services need a pipe-and-filter layer in front of the library
D. The shared library couples the services, so they have lost independent deployment

695. A three-stage pipe-and-filter pipeline processes items with filters taking 2 ms, 10 ms and 3 ms per item; each filter runs in parallel on its own stream. What limits throughput, and roughly what is it? Analysis

A. The 10 ms filter, giving about 100 items per second
B. The sum of all filters, giving about 67 items per second
C. The 2 ms filter, giving about 500 items per second
D. The 3 ms filter, giving about 333 items per second

696. A price service publishes PriceChanged events, and a consumer sometimes receives "price = 90" after "price = 100" even though 100 was set later. Displayed prices end up wrong. What is the best fix? Analysis

A. Give each event a version or sequence number per product and ignore events older than the last one applied
B. Publish each event twice so it is less likely to be lost
C. Make the consumer process events faster
D. Switch the consumer to polling the price service every hour

697. In a layered app, the Business Logic layer imports a UI dialog class to show error pop-ups. What is wrong, and what is the better design? Analysis

A. Nothing is wrong; layers may call in any direction
B. A lower layer depends on a higher one; Business Logic should raise errors or return results that the UI layer displays
C. The UI should be moved below Business Logic
D. The dialog class should be copied into Business Logic

698. Analysts propose replacing a working layered monolith with microservices. The team has six developers, the load is steady, and no module needs separate scaling. Which conclusion is best supported? Analysis

A. Migrate at once, because microservices are always more modern and reliable
B. Keep the modular monolith, because microservices would add network, monitoring and consistency costs without a matching benefit
C. Migrate, because microservices are required for a layered design
D. Replace the monolith with a peer-to-peer network to remove the server

User Interface Design MCQs for NSCT

699. Which Nielsen heuristic says the system should keep users informed about what is going on through timely feedback? Understanding

A. Match between system and the real world
B. Aesthetic and minimalist design
C. Help and documentation
D. Visibility of system status

700. Nielsen's heuristic "Recognition rather than recall" mainly aims to: Understanding

A. Make the interface look attractive and uncluttered
B. Let expert users work faster with shortcuts
C. Provide searchable online documentation
D. Reduce the user's memory load by keeping options and information visible

701. Which of these is one of Shneiderman's Eight Golden Rules of interface design? Understanding

A. Use as many colours as possible
B. Hide system status from novices
C. Permit easy reversal of actions
D. Require users to memorise commands

702. In Nielsen's five quality components of usability, "learnability" means: Understanding

A. How quickly experienced users can perform tasks
B. How easily users re-establish skill after a period of not using it
C. How pleasant the design is to use
D. How easily users can complete basic tasks the first time they meet the design

703. What is a low-fidelity prototype? Understanding

A. A fully working version of the final product
B. A prototype written in the final production language
C. A prototype with final visuals but no navigation
D. A rough, quick representation such as paper sketches or simple wireframes

704. According to the international standard ISO 9241-11, usability is the extent to which specified users can achieve specified goals with: Understanding

A. Speed, colour and layout on a specified device
B. Effectiveness, efficiency and satisfaction in a specified context of use
C. Security, reliability and availability on a specified network
D. Low cost, short schedule and small team size

705. Which Nielsen heuristic is applied when an app uses a shopping-cart icon and words like "checkout" that users already know from real shops? Understanding

A. Match between system and the real world
B. Consistency and standards
C. Error prevention
D. Flexibility and efficiency of use

706. Pressman, following Theo Mandel, gives three "golden rules" of interface design. Which set is correct? Understanding

A. Use colour; use animation; use sound
B. Prevent errors; write documentation; train users
C. Minimise cost; minimise screens; minimise testing
D. Place the user in control; reduce the user's memory load; make the interface consistent

707. Shneiderman's rule "Design dialogs to yield closure" means: Understanding

A. Close every dialog box automatically after a few seconds
B. Group action sequences with a clear beginning, middle and end, with feedback when a group completes
C. Never let users cancel a sequence once it starts
D. Put every option on one screen so no further dialogs are needed

708. In Norman's terms, what is a signifier? Understanding

A. The physical possibility of an action, whether or not it is visible
B. The relationship between controls and their effects in the world
C. A perceivable cue that shows where or how to act, such as a label or a raised button edge
D. A limitation that stops users doing the wrong thing

709. In Nielsen's 0 to 4 severity scale for usability problems, what does a rating of 4 mean? Understanding

A. Cosmetic problem: fix only if time allows
B. Usability catastrophe: imperative to fix before release
C. Minor problem: low priority
D. Not a usability problem at all

710. How does a horizontal prototype differ from a vertical prototype? Understanding

A. Horizontal is on paper; vertical is always coded
B. Horizontal shows many features shallowly; vertical implements a few features in depth
C. Horizontal is for mobile screens; vertical is for desktop screens
D. Horizontal is thrown away; vertical is always kept

711. What distinguishes evolutionary prototyping from throwaway prototyping? Understanding
    A. An evolutionary prototype is always on paper; a throwaway prototype is always coded
    B. Evolutionary prototyping is used only for databases
    C. An evolutionary prototype is refined step by step into the delivered system; a throwaway prototype is discarded after it clarifies requirements
    D. Throwaway prototypes need more testing than the final system

712. Norman's "gulf of evaluation" refers to the difficulty of: Understanding
    A. Working out which action to take to reach a goal
    B. Paying for an expensive usability evaluation
    C. Evaluating code quality during reviews
    D. Working out the system's state and whether the action achieved the goal

713. Which method has evaluators step through a task as a new user would, asking at each step whether the user will know what to do and will understand the system's response? Understanding
    A. Heuristic evaluation
    B. A/B testing
    C. Cognitive walkthrough
    D. Card sorting

714. Shneiderman's golden rules have changed across editions of his book. In the 5th and 6th editions, which rule replaced "Enable frequent users to use shortcuts"? Understanding
    A. Offer informative feedback
    B. Seek universal usability
    C. Strive for consistency
    D. Reduce short-term memory load

715. A file upload shows no indicator for 20 seconds, so users click Upload again. Which Nielsen heuristic is mainly violated? Application
    A. Visibility of system status
    B. Aesthetic and minimalist design
    C. Help and documentation
    D. Match between system and the real world

716. In one app, the Save button is green and bottom-right on some screens and grey and top-left on others. Which principle does this break? Application
    A. Error prevention
    B. Help and documentation
    C. Consistency and standards
    D. User control and freedom

717. A date field is replaced by a calendar picker that greys out past dates for a flight booking. Which heuristic does this mainly apply? Application
    A. Error prevention
    B. Visibility of system status
    C. Help and documentation
    D. Aesthetic and minimalist design

718. A form shows the error "ERR_422: constraint violation" when the phone number has letters in it. Which heuristic is violated? Application
    A. Flexibility and efficiency of use
    B. Help users recognise, diagnose and recover from errors
    C. Recognition rather than recall
    D. Visibility of system status

719. Early in a project, designers draw screens on paper and ask users to "tap" them while a team member swaps the sheets. What is this? Application

A. High-fidelity prototyping
B. Beta testing
C. Regression testing
D. Low-fidelity paper prototyping

720. An editor adds Ctrl+S, Ctrl+Z and customisable toolbars, while keeping menus for new users. Which Nielsen heuristic does this support? Application

A. Recognition rather than recall
B. Help and documentation
C. Match between system and the real world
D. Flexibility and efficiency of use

721. A checkout wizard has no Back button and no Cancel option once started. Which heuristic is violated? Application

A. User control and freedom
B. Aesthetic and minimalist design
C. Match between system and the real world
D. Help and documentation

722. Designers test a voice assistant before the speech engine exists: a hidden team member types the replies that users believe come from the system. What technique is this? Application

A. Wizard of Oz prototyping
B. Heuristic evaluation
C. Cognitive walkthrough
D. Evolutionary prototyping

723. Using Nielsen and Landauer's model, the share of usability problems found by n users is 1 − (1 − L)^n, with L = 0.31. About what share do 5 users find? Application

A. About 16%
B. About 69%
C. 100%
D. About 84%

724. With Hick's law T = b·log2(n + 1) and b = 0.2 s, how much longer does choosing from 15 equally likely options take than choosing from 7? Application

A. Twice as long
B. 1.6 s longer
C. 0.2 s longer
D. No change

725. A web form shows labels only as grey placeholder text inside the boxes, which disappears once the user starts typing. Which heuristic does this mainly violate? Application

A. Recognition rather than recall
B. Flexibility and efficiency of use
C. Help and documentation
D. Visibility of system status

726. Which grey text colour on a white background meets the WCAG AA contrast minimum of 4.5:1 for normal-size text? Application

A. #767676
B. #777777
C. #999999
D. #AAAAAA

727. Using Fitts's law in the Shannon form MT = a + b·log2(D/W + 1), with a = 0.1 s and b = 0.2 s/bit, what is the movement time to a button 16 px wide at a distance of 256 px? Application
    A. About 0.92 s
    B. 0.90 s
    C. 1.10 s
    D. 3.50 s

728. A Delete Account button sits right next to Save and is styled the same, and users sometimes delete accounts by accident. Which redesign best applies error prevention? Application
    A. Show a clearer error message after the account is deleted
    B. Separate and visually distinguish the destructive button and ask for an explicit confirmation
    C. Add a help page explaining what each button does
    D. Make both buttons smaller so users click more carefully

729. A team will run one evaluation before its next sprint. Budget is tiny, no users can be recruited this week, and they want to find as many general usability problems as possible. Which method is the best choice? Analysis
    A. A large summative usability test with 50 recruited users
    B. Heuristic evaluation by three to five evaluators
    C. A/B testing two designs on live traffic
    D. Waiting for app-store reviews after release

730. Requirements for a hospital booking app are unclear, and clinicians cannot say what they want until they see something. The final system must be built to strict reliability standards. Which prototyping strategy fits best? Analysis
    A. Evolutionary prototyping, shipping the first prototype code as the product
    B. Throwaway prototyping to clarify requirements, then build the real system properly
    C. Skip prototyping and write a full specification first
    D. Build only a high-fidelity visual mock-up and ship it

731. A heuristic evaluation finds: (P) the logo is 2 px off-centre; (Q) the payment button fails silently on some phones so orders are lost; (R) a help link uses jargon. Using Nielsen's severity scale, what is the correct priority order? Analysis
    A. P, then Q, then R
    B. R, then P, then Q
    C. Q, then R, then P
    D. All three are equal because each breaks a heuristic

732. An ATM's message says "Transaction complete" before the card is returned, and many users walk away without their card. Which design change best fixes the root cause? Analysis
    A. Make the "Transaction complete" message larger
    B. Return the card before dispensing cash, so the goal cannot be met while the card is still in the machine
    C. Add a printed notice on the ATM telling users to take their card
    D. Sound a beep after the user has walked away

733. Analytics show 70% of users abandon a ten-field sign-up form. Usability tests show they are unsure which fields are required and lose all their input when one error occurs. Which change addresses both findings best? Analysis
    A. Mark required fields clearly, validate inline and keep entered data after an error
    B. Change the brand colours of the form to be more attractive
    C. Add a 20-page user manual linked from the form
    D. Move all ten fields onto one very long line to save space

734. A product team has an untested idea for a new budgeting feature and two days to decide whether users understand the concept. What should they build first? Analysis
    A. A fully coded high-fidelity version connected to real bank data
    B. A production database schema for the feature
    C. A detailed visual style guide for the feature
    D. Low-fidelity sketches or wireframes to test with a few users

735. A popular screen shows every one of 40 settings at once. Novices feel overwhelmed, while power users want all of them quickly. Which design best balances Nielsen's minimalist design with flexibility and efficiency? Analysis
    A. Delete the 30 least-used settings for everyone
    B. Keep all 40 settings on one screen in a smaller font
    C. Show the most common settings by default and put advanced ones behind an "Advanced" section with search
    D. Hide every setting and require a command-line flag to change any

736. After an email is sent, the app shows "Message sent" with an Undo link for ten seconds instead of an "Are you sure?" dialog before every send. What is the main design advantage? Analysis
    A. It supports easy reversal without slowing every send with a confirmation users learn to ignore
    B. It prevents users from ever sending an email by mistake
    C. It removes the need for any feedback after sending
    D. It hides system status to reduce clutter

737. A government portal passes automated accessibility checks, yet blind users using screen readers cannot complete a payment because a custom drop-down has no keyboard focus or label. What does this show? Analysis
    A. The portal is fully accessible because it passed the checker
    B. Screen-reader users are outside the target audience
    C. Automated checks are not enough; testing with assistive technology and real users is needed
    D. Only colour contrast matters for accessibility

738. A team must choose between (X) a settings screen where changes apply immediately with visible undo, and (Y) one where nothing applies until the user presses Save at the bottom of a long page. Users often leave Y without saving. Which choice and reason are best? Analysis
    A. Y, because users should be forced to read the whole page
    B. Y, because removing feedback reduces visual clutter
    C. X, because it gives immediate feedback and easy reversal, so changes are not silently lost
    D. X, because it makes undo unnecessary

739. Nielsen's heuristic "Aesthetic and minimalist design" says that: Understanding
    A. Interfaces should use as few colours as possible
    B. Every screen must look identical to every other screen
    C. Decorative images should replace text labels
    D. Interfaces should not contain information that is irrelevant or rarely needed, because it competes with the relevant information

740. Which statement matches Nielsen's heuristic "Help and documentation"? Understanding
    A. A good system never needs documentation, so help should be removed
    B. Help should be easy to search, focused on the user's task and list concrete steps
    C. Documentation should be a complete technical manual for developers
    D. Help should only be available by contacting support

741. In Nielsen's five components of usability, what does memorability mean? Understanding

A. How quickly users complete tasks once they have learned the design
B. How easily users can re-establish proficiency after a period of not using the design
C. How easily users complete basic tasks the first time
D. How pleasant the design is to use

742. In Nielsen's five components of usability, what does efficiency mean? Understanding

A. How few errors users make
B. How easily users return after a break
C. How quickly a new user learns basic tasks
D. How quickly users can perform tasks once they have learned the design

743. In Norman's terms, what is an affordance? Understanding

A. A label that tells the user what a button does
B. The delay between an action and the system's feedback
C. A relationship between an object's properties and a person's abilities that determines how the object could be used
D. A rule that forces actions to happen in a fixed order

744. What does Norman mean by natural mapping? Understanding

A. Arranging controls so their layout corresponds to the layout of the things they control
B. Using real-world photographs as icons
C. Translating the interface into the user's language
D. Linking each menu item to a keyboard shortcut

745. Norman's gulf of execution is the gap between: Understanding

A. What the user intends to do and the actions the system makes available
B. What the system displays and the user's understanding of its state
C. The system's response time and the user's patience
D. The designer's model and the programmer's code

746. Norman's seven stages of action have execution and evaluation sides. Which three stages are on the evaluation side? Understanding

A. Plan the action, specify the sequence, perform the sequence
B. Perceive the state of the world, interpret the perception, compare the outcome with the goal
C. Form the goal, plan the action, perceive the state
D. Specify the sequence, interpret the perception, form the goal

747. A USB-C plug fits only one way into a USB-C socket's shape, while a large drag handle is the only part of a panel that can be grabbed. In Norman's terms these are examples of: Understanding

A. Mappings, which link controls to their effects
B. Feedback, which reports the result of an action
C. Conceptual models, which explain how the system works
D. Constraints, which limit the possible actions

748. In a think-aloud usability test, participants: Understanding

A. Say what they are thinking, looking for and feeling while they carry out tasks
B. Read the interface text out loud to check its grammar
C. Explain the design to other users after the test
D. Answer a questionnaire about their age and background

749. What is the difference between formative and summative usability evaluation? Understanding

A. Formative evaluation uses experts only; summative uses users only
B. Formative evaluation is done after release; summative is done on paper sketches
C. They are two names for the same method
D. Formative evaluation finds problems to improve a design during development; summative evaluation measures how well a finished design performs

750. What is card sorting mainly used for in UI design? Understanding

A. Testing whether buttons are large enough to tap
B. Measuring the colour contrast of text
C. Discovering how users group and label content, to shape the information architecture and navigation
D. Estimating the time a user needs to press each key

751. What is A/B testing? Understanding

A. Asking two experts to review the same screen
B. Testing the design first on Android, then on iOS
C. Showing two versions of a design to different groups of live users and comparing a measured outcome
D. Grading usability problems from A (severe) to B (minor)

752. Shneiderman's rule "Offer informative feedback" says that feedback should be: Understanding

A. Matched to the action: modest for frequent, minor actions and more substantial for infrequent, major ones
B. The same full dialog box for every action
C. Given only when an error occurs
D. Delayed until the user finishes the whole task

753. Shneiderman's rule "Reduce short-term memory load" is often linked to which finding? Understanding

A. Fitts's law on pointing time
B. Hick's law on choice time
C. The Pareto principle that 80% of effects come from 20% of causes
D. Miller's finding that people can hold only about seven, plus or minus two, chunks in short-term memory

754. In older editions of Shneiderman's Eight Golden Rules, "Keep users in control" was worded as: Understanding

A. Strive for universal usability
B. Support internal locus of control
C. Design dialogs to yield closure
D. Offer simple error handling

755. Which Gestalt principle says that items placed close together are seen as a group? Understanding

A. Proximity
B. Similarity
C. Closure
D. Continuity

756. A toolbar shows all clickable links in blue and underlined, and plain text in black. Users treat the blue items as one group. Which Gestalt principle is at work? Understanding

A. Proximity
B. Similarity
C. Figure-ground
D. Closure

757. The four principles of WCAG are often summarised as POUR. What do they stand for? Understanding

A. Portable, open, usable, reliable
B. Perceivable, organised, usable, responsive
C. Predictable, operable, uniform, readable
D. Perceivable, operable, understandable, robust

758. WCAG 2 level AA needs 4.5:1 contrast for normal text. What is the AA minimum for large text (about 18 pt, or 14 pt bold)? Understanding

A. 4.5:1
B. 7:1
C. 3:1
D. 2:1

759. How does a user's mental model differ from the designer's conceptual model? Understanding

A. They are the same thing under two names
B. The mental model is the user's own understanding of how the system works; the conceptual model is the explanation the designer intends the system to convey
C. The mental model is a diagram drawn by the designer; the conceptual model is the database schema
D. The mental model applies only to experts; the conceptual model only to novices

760. What is a persona in UI design? Understanding

A. A real customer who signs the project's acceptance form
B. A fictional but research-based profile of a typical user, with goals and behaviours, used to guide design decisions
C. The visual style guide of the product
D. A chatbot character that greets users

761. Pressman describes the user interface design process as a spiral of four framework activities. Which set is correct? Understanding

A. Requirements, coding, testing, maintenance
B. Sketch, colour, animate, publish
C. Plan, risk analysis, engineering, customer evaluation
D. Interface analysis and modelling, interface design, interface construction, interface validation

762. Shneiderman's direct manipulation interfaces are characterised by: Understanding

A. Visible objects of interest, physical actions instead of complex syntax, and rapid, reversible actions with immediate visible results
B. Typed commands with strict syntax and no visual feedback
C. Voice input only, with no screen
D. Batch jobs submitted and reviewed the next day

763. With Hick's law $T = b \cdot \log_2(n + 1)$ and $b = 0.15$ s, what is the decision time for 31 equally likely options? Application

A. 4.65 s
B. 0.15 s
C. 0.6 s
D. 0.75 s

764. Using Fitts's law $MT = a + b \cdot \log_2(D/W + 1)$ with $a = 0.05$ s and $b = 0.1$ s/bit, what is the movement time to a target 20 px wide at a distance of 300 px? Application

A. 1.55 s
B. 0.45 s
C. 0.40 s
D. 0.35 s

765. Using Fitts's index of difficulty ID = log2(D/W + 1), a button 448 px away is widened from 32 px to 64 px. By about how much does the ID fall? Application

A. About 0.9 bits
B. Exactly 1 bit
C. About 2 bits
D. It does not change, because the distance is the same

766. Using Nielsen and Landauer's model 1 − (1 − L)^n with L = 0.31, about what share of usability problems do 3 test users find? Application

A. About 93%
B. About 31%
C. About 67%
D. About 84%

767. With Nielsen and Landauer's model 1 − 0.69^n, what is the smallest number of test users predicted to find at least 90% of the usability problems? Application

A. 5
B. 6
C. 7
D. 3

768. Grey text #777777 on a white background has a contrast ratio of about 4.48:1. Under WCAG 2 level AA, where may it be used? Application

A. For large text only, because normal text needs at least 4.5:1
B. For all text, because 4.48 rounds to 4.5
C. For no text at all, because AA needs 7:1
D. For normal text only, because large text needs a higher ratio

769. Using Keystroke-Level Model values H = 0.4 s (move hand to mouse), M = 1.35 s (mental preparation), P = 1.1 s (point) and K = 0.2 s (press a key or mouse button), estimate the time for H, M, P, K. Application

A. 3.05 s
B. 1.70 s
C. 2.65 s
D. 4.40 s

770. A sign-up form shows its password rules (8+ characters, one digit) only after the user submits an invalid password. Which change best applies error prevention? Application

A. Make the error message red and bold
B. Clear the whole form after an invalid password
C. Add a link to a help page about passwords
D. Show the rules beside the field before typing and check them live as the user types

771. One screen says "Delete", another says "Remove" and a third says "Purge" for the same action. Which Nielsen heuristic is violated? Application

A. Visibility of system status
B. Consistency and standards
C. Help and documentation
D. Flexibility and efficiency of use

772. A save button is labelled "Invoke persistence commit". Which Nielsen heuristic does this mainly violate? Application

A. Error prevention
B. Aesthetic and minimalist design
C. Match between system and the real world
D. User control and freedom

773. An app's only help is a 200-page PDF manual with no search or task index. Which Nielsen heuristic is poorly met? Application

A. Recognition rather than recall
B. Help and documentation
C. Visibility of system status
D. Consistency and standards

774. A command-line tool requires users to remember exact command names. The team adds a menu that lists available commands as the user types. Which heuristic does this mainly apply? Application

A. Aesthetic and minimalist design
B. Help and documentation
C. Recognition rather than recall
D. Consistency and standards

775. A cooker has four burners in a square, but the four knobs sit in one straight row, so users often turn on the wrong burner. Which design concept is poor here? Application

A. Closure
B. Affordance
C. Learnability
D. Mapping

776. A door has a flat metal plate on one side and a vertical bar handle on the other, and people rarely push or pull the wrong side. In Norman's terms, what are the plate and the handle? Application

A. Signifiers that show where and how to act
B. Constraints that make the wrong action physically impossible
C. Mappings between two controls
D. Feedback about the door's state

777. A university is redesigning its website menu and wants students to group 50 page titles into categories they name themselves. Which method is this? Application

A. Closed card sorting
B. Heuristic evaluation
C. Open card sorting
D. A/B testing

778. An online shop shows half its visitors a green Buy button and half an orange one, and compares purchase rates over two weeks. What evaluation method is this? Application

A. A/B testing
B. Cognitive walkthrough
C. Card sorting
D. Heuristic evaluation

779. A designer builds a clickable prototype in a design tool with final colours, fonts and images so the client can approve the look before coding. What kind of prototype is this? Application

A. Low-fidelity prototype
B. High-fidelity prototype
C. Wizard of Oz prototype
D. Paper prototype

780. A team builds only the login feature, but fully: the real screen, validation, back-end and database. Other features are not built. What kind of prototype is this? Application

A. Vertical prototype
B. Horizontal prototype
C. Low-fidelity prototype
D. Wizard of Oz prototype

781. After payment, a checkout ends with a page saying "Order #48213 confirmed" and an estimated delivery date. Which Shneiderman rule does this mainly apply? Application

A. Strive for universal usability
B. Reduce short-term memory load
C. Design dialogs to yield closure
D. Keep users in control

782. On a form, each label sits halfway between two text boxes, so users cannot tell which box a label describes. Which Gestalt principle should guide the fix? Application

A. Similarity: make all boxes the same colour
B. Closure: draw incomplete borders around the form
C. Proximity: place each label clearly closer to its own box
D. Figure-ground: add a dark background

783. This button is used to delete an item. Which change makes it accessible to screen-reader users? Application

```
<button>
  <img src="trash.svg">
</button>
```

A. Add a title attribute to the svg file itself
B. Increase the image size to 48 px
C. Add alt="Delete" to the img so the button has an accessible name
D. Wrap the button in a div with a border

784. A team wants to show that its redesigned checkout cuts mean task time compared with the old one, with results stakeholders can trust statistically. Which evaluation fits best? Analysis

A. A heuristic evaluation by three experts
B. A formative think-aloud test with five users
C. A cognitive walkthrough of the new design
D. A summative test with enough participants on both designs, timing tasks without think-aloud

785. A heuristic evaluation lists 25 minor issues on a booking app, but in user tests 4 of 5 users fail at one step the experts rated as minor. What should the team conclude? Analysis

A. Real users' failure is strong evidence; the step should be re-rated as severe and fixed first
B. The experts are right, so the user test should be ignored
C. The 25 minor issues together outrank the one failure
D. Both methods are invalid, so neither result should be used

786. To delete a code repository, a site makes users type the repository's name instead of clicking OK in a standard dialog. Why is this stronger error prevention for this action? Analysis

A. It is faster than clicking OK
B. It helps users recall the name for later
C. It follows the aesthetic and minimalist heuristic
D. Typing the name breaks habitual clicking and confirms the exact target of an irreversible action

787. On desktop systems, a menu bar fixed to the very top edge of the screen is faster to hit with a mouse than one placed a few pixels below it. Which explanation fits Fitts's law? Analysis
    A. The top of the screen is closer to every starting point
    B. Users read the top of the screen first, so they decide faster
    C. The screen edge stops the pointer, so the target acts as if it were very deep, giving a low index of difficulty
    D. Edge targets are always wider in pixels

788. A product team says its new banking app is "for everyone" and declines to define personas. Testing shows the app suits neither retired users nor small-business owners. What is the best analysis? Analysis
    A. The app needs more features so that every user finds something
    B. Retired users and business owners should be told to use another app
    C. Designing for an undefined 'everyone' gave no clear needs to design for; research-based personas for the main groups would focus decisions
    D. The problem is the colour scheme, not the target users

789. A company's app looks identical on iOS and Android, ignoring each platform's usual back navigation and control styles. Android users complain it feels wrong. Which analysis is best? Analysis
    A. The app is perfectly consistent, so the users are wrong
    B. The app should use a different colour on each platform
    C. Internal consistency was kept, but external consistency with platform conventions was lost; follow each platform's standards
    D. Consistency is not a usability principle

790. A team built a quick prototype with hacky code to test ideas. The client liked it, so management wants to ship it as the product. Following Sommerville and Pressman, what is the main risk? Analysis
    A. Prototypes are always too slow to show to users
    B. It was not built for quality, so shipping it brings poor structure and reliability that are costly to maintain
    C. The client will not like the final product
    D. Shipping a prototype breaks Nielsen's heuristic on help

791. In a Wizard of Oz test, a hidden person types replies for a planned voice assistant, and users rate it highly. What is the main threat to the results? Analysis
    A. The human wizard may understand users better than the real speech technology will, so results can be over-optimistic
    B. Users always know the system is fake, so the results are meaningless
    C. The test cannot reveal anything about dialogue design
    D. Wizard of Oz tests can only be run on paper

792. Image-only buttons on a site have no text alternative, and blind users with screen readers hear only "button, button, button". What is the core problem? Analysis
    A. The buttons are too small for screen readers
    B. Screen readers do not support buttons
    C. The page has too many colours
    D. The content is not perceivable to them; each button needs a text alternative such as alt text or an aria-label

793. Which error message best follows Nielsen's guidance for helping users recover from errors when a card payment fails because of a wrong expiry date? Analysis
    A. "Payment error 0x51. Contact your administrator."
    B. "Oops! Something went wrong."
    C. "Invalid input."
    D. "The expiry date doesn't match your card. Check the month and year printed on the card and try again."

794. A team tests five users and calls the app usable. Their users are pharmacists, warehouse staff and delivery drivers, who do very different tasks, and the five were all pharmacists. What is the main flaw? Analysis
    A. Five is too many; one user would have been enough
    B. The five-user rule applies per distinct user group, so warehouse staff and drivers were not tested at all
    C. The test should have used only experts
    D. Usability tests do not apply to business apps

795. During a think-aloud study, the team also records how many seconds each task takes and reports these as the app's expected task times. What is wrong? Analysis
    A. Thinking aloud changes how people work and usually slows them, so the times are not valid performance measures
    B. Think-aloud tests always make users faster than normal
    C. Time should never be measured in any usability study
    D. The study should have used card sorting instead

796. A new airport check-in kiosk will be used mostly by first-time users, and the critical task is printing a boarding pass. With no users available yet, which inspection method fits best? Analysis
    A. Heuristic evaluation of every screen
    B. Cognitive walkthrough of the boarding-pass task
    C. A/B testing on live traffic
    D. A summative timing study with 30 users

797. A report takes about 30 seconds to generate, and the screen shows nothing until it appears. Following Nielsen's response-time limits (0.1 s, 1 s, 10 s), what is the best change? Analysis
    A. Show nothing, because users expect reports to be slow
    B. Show a percent-done progress indicator and allow the user to cancel
    C. Show a busy cursor only
    D. Show the report instantly with placeholder data

798. A smart thermostat accepts a new target temperature, but its display keeps showing only the current room temperature. Users keep pressing the buttons, unsure whether the setting took. Which gulf is wide, and what fixes it? Analysis
    A. The gulf of execution; add more buttons
    B. The gulf of execution; hide the current temperature
    C. Neither; users should read the manual
    D. The gulf of evaluation; show the new target and heating status clearly after each change

Software Implementation & Coding MCQs for NSCT

799. What is the main purpose of a team coding standard? Understanding
    A. To make programs run faster on every platform
    B. To replace the need for testing
    C. To let each developer use a personal style
    D. To make code consistent and readable so it is easier to review and maintain

800. Under PEP 8, Python's official style guide, how should a class be named? Understanding

A. CapWords, for example StudentRecord
B. snake_case, for example student_record
C. UPPER_CASE, for example STUDENT_RECORD
D. camelCase, for example studentRecord

801. How many spaces per indentation level does PEP 8 recommend? Understanding

A. 2
B. 8
C. 4
D. 1 tab character

802. Martin Fowler defines refactoring as a change to the internal structure of software that: Understanding

A. Adds new features requested by the customer
B. Makes it easier to understand and cheaper to modify without changing its observable behaviour
C. Fixes bugs reported by users
D. Improves performance at any cost to readability

803. In pair programming, what are the two roles usually called? Understanding

A. Driver and navigator
B. Author and moderator
C. Tester and recorder
D. Master and scrum master

804. What is static analysis of code? Understanding

A. Examining source code for defects without running it
B. Running the program with test inputs and checking the outputs
C. Measuring the program's memory use while it runs
D. Asking users to try the program and report problems

805. In a Fagan inspection, who plans and leads the inspection meeting? Understanding

A. The moderator
B. The author
C. The reader
D. The recorder

806. How does a code walkthrough usually differ from a formal inspection? Understanding

A. A walkthrough executes the code; an inspection never looks at code
B. A walkthrough is done by customers; an inspection is done by testers only
C. There is no difference; the terms mean the same thing
D. A walkthrough is led by the author and is less formal; an inspection is moderator-led, uses checklists and records defect data

807. In Fowler and Beck's terms, what is a code smell? Understanding

A. A syntax error that stops the code compiling
B. A failing unit test
C. A surface sign in code that often points to a deeper design problem
D. A security vulnerability reported by a scanner

808. Which code smell is described as "one change forces many small edits across many different classes"? Understanding

A. Divergent change
B. Long method
C. Shotgun surgery
D. Lazy class

809. What does the DRY principle state? Understanding

A. Do not run your code until it has been reviewed
B. Data should always be stored in a relational database
C. Every piece of knowledge should have a single, authoritative representation in the system
D. Developers should avoid writing comments

810. What does the term technical debt describe? Understanding

A. Money a company owes to its software vendors
B. The extra future rework caused by taking shortcuts in design or code now
C. The number of open bug reports in a tracker
D. The cost of buying development tools

811. Why should a solid set of automated tests be in place before refactoring? Understanding

A. Because refactoring always introduces new features that need testing
B. To check quickly that the external behaviour has not changed after each step
C. Because tests make the code run faster
D. To replace the need for code reviews

812. In a Fagan inspection, what does the reader do? Understanding

A. Writes down every defect found and its classification
B. Paraphrases the code or document aloud during the meeting, section by section
C. Fixes the defects after the meeting
D. Decides whether the author should be promoted

813. Which code smell is it when one class is changed often, in different ways, for unrelated reasons (for example database changes and tax-rule changes)? Understanding

A. Shotgun surgery
B. Feature envy
C. Data clumps
D. Divergent change

814. Besides writing the code itself, which three implementation issues does Sommerville highlight? Understanding

A. Requirements elicitation, feasibility study and costing
B. Unit testing, system testing and acceptance testing
C. Reuse, configuration management and host-target development
D. Marketing, pricing and licensing

815. A student writes this loop to total three marks. What does it print? Application

```
marks = [70, 85, 90]
total = 0
for i in range(1, len(marks)):
    total += marks[i]
print(total)
```

A. 175
B. 245
C. 160
D. IndexError

816. Before: `fee = credits * 3500`. After: `FEE_PER_CREDIT = 3500` and `fee = credits * FEE_PER_CREDIT`. Which refactoring has been applied? Application

```
credits = 18
fee_before = credits * 3500
FEE_PER_CREDIT = 3500
fee_after = credits * FEE_PER_CREDIT
print(fee_before == fee_after)
```

A. Extract Method
B. Inline Variable
C. Replace Magic Number with Symbolic Constant
D. Introduce Parameter Object

817. Which function name follows PEP 8? Application

A. CalculateGPA
B. calculate_gpa
C. calculateGpa
D. CALCULATE_GPA

818. A reviewer flags this function. What does the last line print? Application

```
def add_item(item, basket=[]):
    basket.append(item)
    return basket

add_item("pen")
print(add_item("ink"))
```

A. ['ink']
B. ['pen', 'ink']
C. ['pen']
D. TypeError

819. What does this code print? Application

```
total = 0.1 + 0.2
print(total == 0.3, abs(total - 0.3) < 1e-9)
```

A. True True
B. True False
C. False True
D. False False

820. This function was refactored to use guard clauses. What does it print? Application

```
def shipping(order_total, is_member):
    if order_total <= 0:
        return "invalid"
    if is_member:
        return 0
    if order_total >= 5000:
        return 0
    return 250

print(shipping(6000, False), shipping(0, True), shipping(1200, False))
```

A. 250 invalid 250
B. 0 invalid 250
C. 0 0 250
D. invalid invalid 250

821. What happens when this code runs? Application

```
sum = 0
for n in [1, 2, 3]:
    sum += n
print(sum)
print(sum([4, 5]))
```

A. It prints 6 and then 9
B. It raises NameError before printing anything
C. It prints 6 twice
D. It prints 6, then raises TypeError: 'int' object is not callable

822. Five functions all take `(start_date, end_date)` together, and the same pair keeps travelling through the code. Which refactoring fits best? Application

A. Inline Method
B. Rename Variable
C. Introduce Parameter Object, such as a DateRange
D. Replace Constructor with Factory Method

823. Which code smell does Invoice.address_label show, and which refactoring fits? Application

```
class Customer:
    def __init__(self, street, city, postcode):
        self.street, self.city, self.postcode = street, city, postcode

class Invoice:
    def __init__(self, customer):
        self.customer = customer

    def address_label(self):
        c = self.customer
        return f"{c.street}, {c.city} {c.postcode}"

print(Invoice(Customer("12 Mall Rd", "Lahore", "54000")).address_label())
```

A. Feature envy; Move Method to Customer
B. Long method; Extract Method
C. Lazy class; Inline Class
D. Primitive obsession; Replace Data Value with Object

824. The same `if shape.kind == "circle" ... elif shape.kind == "square" ...` chain appears in area(), perimeter() and draw(). Adding a triangle means editing all three. Which refactoring fits best? Application

A. Replace Magic Number with Symbolic Constant
B. Rename Method
C. Inline Temp
D. Replace Conditional with Polymorphism

825. PEP 8 gives a rule for comparing a variable with None. Which line follows it? Application

A. if result == None:
B. if result is None:
C. if result = None:
D. if None in result:

826. A developer builds button callbacks in a loop. What does this print? Application

```
handlers = [lambda: i for i in range(3)]
print([h() for h in handlers])
```

A. [0, 1, 2]
B. [0, 0, 0]
C. NameError
D. [2, 2, 2]

827. The same five lines that validate a CNIC number are pasted into register(), update_profile() and apply_scholarship(). Which refactoring removes the duplication? Application

A. Inline Method
B. Rename Variable in each copy
C. Extract Method into one validate_cnic() function and call it from all three
D. Replace Parameter with Method Call

828. A developer validates user input with `assert age >= 0, "bad age"`. Why is this poor practice in Python? Application

```
print(__debug__)
```

A. assert always raises an error even when the condition is true
B. Assertions are removed when Python runs with the -O flag, so the check silently disappears
C. assert only works inside classes
D. assert converts age to a string first

829. A developer says this change is "just an Extract Method refactoring". What does the loop print, and what does it show? Analysis

```
def price_before(qty, unit):
    total = qty * unit
    if total > 1000:
        total = total * 0.9
    return round(total, 2)

def discount(total):
    if total >= 1000:
        return total * 0.9
    return total

def price_after(qty, unit):
    return round(discount(qty * unit), 2)

for q in (5, 10, 11):
    print(price_before(q, 100) == price_after(q, 100), end=" ")
```

A. True False True
B. True True True
C. False False False
D. True False False

830. A code reviewer looks at this function that averages ages from a form. What does it print? Analysis

```
def parse_age(text):
    try:
        return int(text)
    except:
        return 0

ages = ["21", "abc", "", "19"]
print(sum(parse_age(a) for a in ages) / len(ages))
```

A. 20.0
B. ValueError: invalid literal for int() with base 10: 'abc'
C. 13.333333333333334
D. 10.0

831. During a review you see this lookup. What does the final line print, and therefore what is the most important review comment? Analysis

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE users (name TEXT, pin TEXT)")
db.execute("INSERT INTO users VALUES ('ali', '1234'), ('sara', '9999')")

def find_user(name):
    q = f"SELECT name FROM users WHERE name = '{name}'"
    return db.execute(q).fetchall()

print(len(find_user("x' OR '1'='1")))
```

A. 0
B. 2
C. 1
D. sqlite3.OperationalError

832. Pull requests in a team are often 2,000+ lines, and reviewers approve them within minutes with "LGTM". Defects keep reaching production. Which change is most likely to improve review quality? Analysis

    A. Require five reviewers to approve every large pull request
    B. Ban comments in code so there is less to read
    C. Skip reviews and rely only on production monitoring
    D. Keep changes small and focused, and review them against a short checklist

833. Code reviews in a team are dominated by arguments over spacing, quote style and import order, and logic bugs are being missed. What is the best fix? Analysis

    A. Stop using a coding standard
    B. Ask the most senior developer to decide each style dispute
    C. Enforce the style standard automatically with a formatter and linter in CI, so humans focus on logic
    D. Hold a longer review meeting every week

834. You must add a feature to a tangled 3,000-line legacy module that has no tests. Which approach is safest? Analysis

    A. Write characterisation tests for its current behaviour, then refactor in small steps before adding the feature
    B. Rewrite the whole module from scratch over the weekend
    C. Add the feature directly and refactor later if time allows
    D. Refactor everything at once, then write tests afterwards

835. Which comment adds the most value to `timeout = 7`? Analysis

    A. # Gateway drops idle connections after 8 s, so stay just under it
    B. # set timeout to 7
    C. # timeout variable
    D. # TODO

836. A rounding bug in fee calculation was fixed in the admissions module, but students still see wrong fees in the hostel and transport modules. Each module has its own copy of the same calculation. What is the root cause, and the best fix? Analysis

    A. Insufficient logging; add log statements to all three modules
    B. Poor naming; rename the variables in each copy
    C. Slow hardware; move the modules to faster servers
    D. Duplicated code; extract the calculation into one shared function used by all three modules

837. A 3,000-line `StudentManager` class handles enrolment, fee calculation, email sending and PDF report generation, and nearly every change touches it. Which refactoring should come first? Analysis

    A. Extract Class to split out separate classes for each responsibility
    B. Rename the class to StudentSystem
    C. Inline its methods into the code that calls them
    D. Add more comments to each method

838. Before any profiling, a developer replaces a clear loop with a dense one-liner full of bit tricks, saying it "might be faster". The function runs once per day. What is the best review response? Analysis

    A. Accept it, because faster code is always better
    B. Ask for the whole module to be rewritten in assembly
    C. Accept it but remove all tests so it runs faster
    D. Keep the readable version; optimise only after profiling shows a real bottleneck

839. Under PEP 8, how should a module-level constant, such as the maximum credit hours a student may take, be named? Understanding

A. MAX_CREDIT_HOURS
B. maxCreditHours
C. MaxCreditHours
D. max_credit_hours

840. PEP 8 limits each line of code to a maximum of how many characters? Understanding

A. 80
B. 100
C. 120
D. 79

841. What does the YAGNI principle advise developers to do? Understanding

A. Design every possible future feature before writing any code
B. Add functionality only when it is actually needed, not because it might be needed later
C. Keep a single, authoritative copy of every piece of knowledge
D. Keep every function under a fixed number of lines

842. Defensive programming mainly means: Understanding

A. Hiding every error from the user so the program never stops
B. Writing code that checks its inputs and assumptions so that bad data or misuse is caught early
C. Putting a firewall in front of the application server
D. Avoiding exceptions entirely by always returning a default value

843. What does a linter such as Pylint or Flake8 do? Understanding

A. Analyses the source code without running it and reports likely bugs and style breaches
B. Runs the unit tests and reports how much code they covered
C. Compiles Python into machine code so it runs faster
D. Encrypts the source code before it is committed

844. In implementation practice, what does continuous integration (CI) mean? Understanding

A. Each developer works on a private branch for months and merges once before release
B. The program runs on a server continuously without ever being restarted
C. Developers merge their changes into a shared mainline often, and each merge triggers an automated build and test run
D. Integration testing is carried out once, just before delivery

845. In test-driven development (TDD), what is the order of the basic cycle? Understanding

A. Green (write the code), red (write a test), refactor
B. Red (write a failing test), green (write just enough code to pass), refactor
C. Refactor, then red, then green
D. Write all the code, then all the tests, then refactor

846. Why do inspection teams use a checklist of common defect types? Understanding

A. It replaces the need to read the code line by line
B. It focuses reviewers on the defects most likely to occur, so fewer are missed
C. It allows the author to skip the rework stage
D. It is needed so the code can be compiled

847. In a Fagan inspection, which role writes down every defect found during the inspection meeting?
Understanding

A. The recorder (scribe)
B. The moderator
C. The reader
D. The author

848. In Fagan's inspection process, which stage comes immediately after the inspection meeting?
Understanding

A. Rework
B. Follow-up
C. Preparation
D. Overview

849. What happens during the preparation stage of a Fagan inspection? Understanding

A. The author fixes the defects found in the meeting
B. The moderator checks that every defect was resolved
C. The whole team meets to log defects aloud
D. Each inspector studies the material alone, using the checklist, before the meeting

850. In Sommerville's description of implementation issues, what is host-target development?
Understanding

A. Hosting the source code on a web server for the target users
B. Testing the software only on the customers' own machines
C. Developing software on one computer (the host) and running it on a different one (the target)
D. Splitting the team into a host group and a target group

851. Which configuration management activity, as Sommerville describes it, lets developers report bugs and see who is working on each one and when it was fixed? Understanding

A. Version management
B. System integration
C. Problem tracking
D. Release management

852. PEP 8 asks for imports to be grouped, with a blank line between groups. In which order?
Understanding

A. Local application, then third-party, then standard library imports
B. One alphabetical list, whatever the source of each module
C. Third-party, then standard library, then local application imports
D. Standard library, then related third-party, then local application imports

853. A program stores phone numbers, CNIC numbers and money amounts as plain strings and floats, and the same validation code is repeated wherever they are used. Which code smell is this?
Understanding

A. Feature envy
B. Primitive obsession
C. Lazy class
D. Speculative generality

854. Code such as order.get_customer().get_address().get_city().get_name() shows which code smell?
Understanding

A. Middle man
B. Data clumps
C. Divergent change
D. Message chains

855. Why do Python coding standards forbid a bare except: clause? Understanding

A. It is a syntax error in Python 3
B. It only catches errors raised on the same line as the try
C. It makes the whole program run more slowly
D. It also catches exceptions such as KeyboardInterrupt and SystemExit, and it hides real programming errors

856. Gerald Weinberg's idea of 'egoless programming' encourages developers to: Understanding

A. Avoid code reviews so that no one's feelings are hurt
B. Work alone so that each person gets full credit for their code
C. Let the most senior developer approve all code without discussion
D. Treat code as a team product and welcome others finding defects in it

857. What does Python do at run time with the type hints in def area(r: int) -> int? Understanding

A. It raises TypeError if a float is passed as r
B. It converts the argument to int automatically
C. It enforces nothing: the hints are stored as annotations and checked only by separate tools such as mypy
D. It compiles the function to faster machine code

858. Fowler describes a 'rule of three' for deciding when to refactor. What does it say? Understanding

A. Refactor only once in every three iterations
B. A method should never have more than three parameters
C. The first time you do something, just do it; the second time, note the duplication; the third time, refactor
D. Each refactoring must be approved by three reviewers

859. Sommerville says software reuse is possible at several levels. Which list matches his levels? Understanding

A. Statement, function, file and folder
B. Requirements, design, testing and deployment
C. Source code, binary, byte code and cloud
D. Abstraction, object, component and system

860. A team adds abstract base classes, hook methods and unused parameters 'in case we need them later', and none of them is ever used. Which code smell has Fowler named for this? Understanding

A. Refused bequest
B. Speculative generality
C. Middle man
D. Divergent change

861. Why does Fowler list 'comments' among the code smells? Understanding

A. Comments slow down the interpreter at run time
B. Clean code must contain no comments at all
C. Heavy comments often explain unclear code, so the code should be refactored (for example with Extract Function) until the comment is unneeded
D. Comments stop static analysis tools from working

862. Which variable name best follows the meaningful-names guideline for the number of days a library book is overdue? Application

A. d
B. days_overdue
C. temp2
D. int_var_days_x

863. Which line follows PEP 8's whitespace rules? Application

A. result=compute(a,b)
B. result = compute(a, b)
C. result = compute(a, b)
D. result = compute (a, b)

864. What happens when this code runs? Application

```
name = "ali"
name[0] = "A"
print(name)
```

A. It raises TypeError: 'str' object does not support item assignment
B. It prints Ali
C. It prints ali
D. It raises NameError

865. Two subclasses, Lecturer and Professor, contain identical email_address() methods. Both inherit from Employee. Which refactoring removes the duplication? Application

A. Pull Up Method into Employee
B. Push Down Method into each subclass
C. Inline Class
D. Rename Method

866. A code inspection finds 18 defects in a 4,500-line module. What is the defect density per KLOC? Application

A. 0.4
B. 4.0
C. 25
D. 40

867. What does this code print? Application

```
print(list(range(2, 11, 3)))
```

A. [2, 5, 8, 11]
B. [3, 6, 9]
C. [2, 4, 6, 8, 10]
D. [2, 5, 8]

868. What does this code print? Application

```
print(7 // 2, -7 // 2, 7 % -2)
```

A. 3 -4 -1
B. 3 -3 1
C. 3 -3 -1
D. 3.5 -3.5 1

869. What does this code print? Application

```
def load(x):
    try:
        v = 10 // x
    except ZeroDivisionError:
        print("zero", end=" ")
        return -1
    else:
        print("ok", end=" ")
        return v
    finally:
        print("done", end=" ")

print(load(0))
```

A. zero -1 done
B. zero -1
C. zero done -1
D. zero ok done -1

870. What does this code print? Application

```
def check(tag):
    print(tag, end=" ")
    return tag == "b"

print(check("a") and check("b"), check("b") or check("c"))
```

A. a b b c False True
B. a b False True
C. a b c False True
D. False True a b

871. What happens when this code runs? Application

```
count = 0

def visit():
    count += 1
    return count

print(visit())
```

A. It prints 1
B. It prints 0
C. It raises SyntaxError
D. It raises UnboundLocalError

872. A coding standard says money must use Decimal. What does this code print? Application

```
from decimal import Decimal

a = 0.1 + 0.2 == 0.3
b = Decimal("0.1") + Decimal("0.2") == Decimal("0.3")
c = Decimal(0.1) == Decimal("0.1")
print(a, b, c)
```

A. False True True
B. True True True
C. False True False
D. False False False

873. Before: student.cgpa = 5.2 is set directly on a public attribute anywhere in the code. After: cgpa becomes a property whose setter rejects values outside 0.0 to 4.0. Which refactoring has been applied? Application

A. Extract Class
B. Inline Temp
C. Encapsulate Field (Encapsulate Variable)
D. Move Field

874. Before: if date < SUMMER_START or date > SUMMER_END: charge = qty * winter_rate + winter_fee. After: if not_summer(date): charge = winter_charge(qty). Which refactoring is this? Application

A. Replace Conditional with Polymorphism
B. Decompose Conditional
C. Replace Nested Conditional with Guard Clauses
D. Introduce Parameter Object

875. An inspection found 45 defects in a release. System testing later found 12 more, and users reported 3 after release. What percentage of all known defects did the inspection catch? Application

A. 79%
B. 25%
C. 80%
D. 75%

876. What does this code print? Application

```
try:
    value = int(None)
except ValueError:
    print("bad number")
except Exception:
    print("other")
else:
    print("ok")
```

A. other
B. bad number
C. ok
D. Nothing; it crashes with an uncaught TypeError

877. Many JavaScript coding standards require === instead of ==. What does this line print? Application

```
console.log("5" + 3, "5" - 3, 1 == "1", 1 === "1");
```

A. 8 2 true true
B. 53 2 true false
C. 53 2 false false
D. 8 8 true false

878. What does this code print? Application

```
grid = [[0] * 3] * 2
grid[0][1] = 5
print(grid)
```

A. [[0, 5, 0], [0, 0, 0]]
B. [[5, 5, 5], [0, 0, 0]]
C. [[0, 5, 0], [0, 5, 0]]
D. It raises IndexError

879. What does this code print? Application

```
squares = (n * n for n in range(4))
print(sum(squares), sum(squares))
```

A. 14 14
B. 30 0
C. It raises StopIteration
D. 14 0

880. The aim is to remove every mark below 50. What does this code actually print? Application

```
marks = [40, 55, 30, 20, 70]
for m in marks:
    if m < 50:
        marks.remove(m)
print(marks)
```

A. [55, 70]
B. [55, 30, 70]
C. [55, 20, 70]
D. It raises RuntimeError

881. A fee report rounds values with Python's built-in round(). What does this line print? Application

```
print(round(2.5), round(3.5), round(-0.5))
```

A. 2 4 0
B. 3 4 -1
C. 3 4 0
D. 2 3 -1

882. Before: base = qty * price is stored in a local variable and used three times in a long method. After: a method base_price() returns qty * price, and the three uses call it. Which refactoring is this? Application

A. Extract Variable
B. Inline Method
C. Split Temporary Variable
D. Replace Temp with Query

883. A reviewer notes four possible issues in this function. Which one should be fixed first? Analysis

```
def find_student(cur, roll_no):
    query = f"SELECT * FROM students WHERE roll_no = '{roll_no}'"
    cur.execute(query)
    return cur.fetchall()
```

A. SELECT * should list the needed columns explicitly
B. User input is pasted into the SQL string, so it is open to SQL injection; use a parameterised query
C. The function should be called findStudent
D. roll_no should be renamed roll_number

884. A reviewer examines this helper. What does it print, and what is the real problem? Analysis

```
def average(marks):
    try:
        return sum(marks) / len(marks)
    except:
        return 0

print(average([]), average([80, "90"]))
```

A. 0, followed by an uncaught TypeError
B. 0 85.0; the function is correct
C. It raises ZeroDivisionError for the empty list
D. 0 0; the bare except hides a TypeError from bad data as well as the empty-list case

885. A developer 'tidies' a condition to follow PEP 8's advice on None and calls it a refactoring. What does the loop print, and what does it show? Analysis

```
def label_old(discount):
    if discount:
        return "discounted"
    return "full price"

def label_new(discount):
    if discount is not None:
        return "discounted"
    return "full price"

for d in [0, None, 15]:
    print(label_old(d) == label_new(d), end=" ")
```

A. True True True; the change is a safe refactoring
B. False True True; the change alters behaviour for 0, so it is not a refactoring
C. False False True; it alters behaviour for both 0 and None
D. True False True; it alters behaviour only for None

886. A team is about to release the dose-calculation module of hospital infusion-pump software. Which review approach is most appropriate? Analysis

A. A formal Fagan inspection with a trained moderator, checklists and recorded defect data
B. A quick over-the-shoulder look by whichever teammate is free
C. No review, because the unit tests already pass
D. A walkthrough led by the author, mainly to explain the design

887. Two customers each add one item to their own cart. What does the code print, and what is the fix? Analysis

```
class Cart:
    items = []

    def add(self, item):
        self.items.append(item)

a = Cart()
b = Cart()
a.add("pen")
b.add("book")
print(len(a.items), len(b.items))
```

A. 1 1; each Cart object gets its own list
B. 1 2; b sees a's item because it was created later
C. 0 0; append on a class attribute is ignored
D. 2 2; items is a class attribute shared by all carts, so create it in __init__ as self.items = []

888. What does this code print, and what does a reviewer conclude? Analysis

```
def parse(s):
    try:
        return int(s)
    except Exception:
        return "error"
    except ValueError:
        return "not a number"

print(parse("7"), parse("seven"))
```

A. 7 not a number; Python picks the most specific handler
B. 7 error; the ValueError handler can never run, because except Exception catches it first
C. 7 error; both handlers run in turn
D. It raises SyntaxError, because handler order is checked when the code is compiled

889. This loop is meant to step a threshold from 0 to 1 in steps of 0.1, but it never stops. What is the cause and the best fix? Analysis

```
x = 0.0
while x != 1.0:
    x += 0.1
```

A. After ten steps x is 0.9999999999999999, not 1.0, so x != 1.0 stays true; loop over an integer counter, such as x = i / 10 for i in range(11)
B. 0.1 is stored exactly, so the defect must be in the while syntax
C. Python floats overflow just above 1.0, so x wraps around to 0
D. The loop needs a break statement straight after x += 0.1

890. You add a function to a legacy module in which every existing function uses camelCase, such as getStudentFee(). The team has no other style rule. What does PEP 8 itself advise? Analysis

    A. Name it get_student_fee, because PEP 8 always overrides local style
    B. Match the module's existing style, because PEP 8 says consistency within one module matters most
    C. Rename every function in the module to snake_case in the same commit as the feature
    D. Use CapWords so the name matches neither style

891. The comment and the code disagree. What is the best review action? Analysis

```
# Apply 10% discount for members
if member:
    total *= 0.85
```

    A. Trust the comment, since comments always state the intent
    B. Trust the code, since it is what runs, and leave the comment as it is
    C. Delete every comment in the module so they cannot go stale
    D. Check the intended rule in the requirement, fix whichever line is wrong, and keep comment and code in step

892. A reviewer flags Penguin. Which code smell does it show, and what is the usual remedy? Analysis

```
class Bird:
    def fly(self):
        return "flying"

class Penguin(Bird):
    def fly(self):
        raise NotImplementedError("penguins cannot fly")
```

    A. Feature envy; Move Method to Bird
    B. Lazy class; Inline Class into Bird
    C. Refused bequest; rework the hierarchy, for example by replacing inheritance with delegation
    D. Shotgun surgery; Move Field into Penguin

893. Review logs show that reviews covering more than 1,500 lines per hour find far fewer defects per line than reviews at a few hundred lines per hour. What should the team do? Analysis

    A. Limit the review rate and session length so that reviewers read carefully
    B. Speed up all reviews, since the fast ones are clearly more efficient
    C. Stop reviewing and rely on system testing alone
    D. Let authors review their own code to save time

894. A code review spots a function that is never called and a comment that contradicts the code. Why might testing alone have missed both? Analysis

    A. Testing always finds every defect that a review finds
    B. Reviews run the code with more inputs than tests do
    C. Testing runs code and checks outputs, so unused code and misleading comments cause no failure to observe
    D. Dead code always crashes the program at run time

895. Unit tests pass on a developer's 64-bit PC, but the same C code fails on a 16-bit embedded board once a counter passes 32,767. Which implementation issue does this illustrate, and what helps most? Analysis

A. Host-target development: int sizes differ, so use fixed-width types such as int32_t and test on the target or a simulator
B. Code reuse: the counter was copied from another project
C. Configuration management: the wrong file version was compiled
D. Refactoring: the function needs Extract Method

896. Doing TDD, a developer writes a new test for a feature that does not exist yet, and it passes on the very first run. What does this suggest? Analysis

A. The test is not checking the new behaviour (or the feature already exists), so the test must be examined before continuing
B. The feature has been implemented correctly
C. TDD has been followed correctly, so the developer can move to refactoring
D. The test runner is faster than expected

897. Module P changes in almost every sprint and is full of duplicated code. Module Q has not changed in two years and is equally messy. There is time for one refactoring. Which should it be? Analysis

A. Module Q, because messy code is equally risky wherever it is
B. Neither, because refactoring never repays its cost
C. Module P, because technical debt costs 'interest' every time code is changed
D. Rewrite both from scratch instead of refactoring

Software Testing MCQs for NSCT

898. In ISTQB terminology, a developer misreads the specification and therefore writes an incorrect condition. What is the developer's misreading called? Understanding

A. A defect (fault)
B. A failure
C. A test incident
D. An error (mistake)

899. Black-box test cases are derived mainly from which source? Understanding

A. The specification or requirements, without looking at the internal code
B. The source code statements and branches
C. The control-flow graph of each function
D. The compiler's optimisation report

900. Which of these is a white-box (structure-based) test technique? Understanding

A. Equivalence partitioning
B. Branch testing
C. Boundary value analysis
D. State transition testing

901. Which statement matches the ISTQB principle 'testing shows the presence of defects, not their absence'? Understanding

A. If all planned tests pass, the software is proven defect-free
B. Testing can reveal defects but cannot prove that no defects remain
C. Exhaustive testing is practical when tests are automated
D. Testing should only start after coding is finished

902. In top-down integration testing, what replaces a lower-level module that has not been written yet? Understanding

A. A driver
B. A stub
C. A test oracle
D. A mutant

903. Beta testing is best described as: Understanding

A. Testing by developers at their own site using debuggers
B. Unit testing of beta versions of each class
C. Load testing performed by an automated tool on the build server
D. Acceptance testing by real users in their own environment before general release

904. A defect is fixed, and the tester re-runs the exact test that failed to check that the fix works. ISTQB calls this: Understanding

A. Regression testing
B. Smoke testing
C. Confirmation testing
D. Exploratory testing

905. Which activity is static testing? Understanding

A. Running unit tests with pytest
B. A load test with 1,000 virtual users
C. A peer review of the source code before it is run
D. Beta testing by customers

906. Which statement about statement and branch coverage is always true? Understanding

A. 100% statement coverage guarantees 100% branch coverage
B. Statement and branch coverage are always equal
C. Branch coverage can never exceed statement coverage
D. 100% branch coverage guarantees 100% statement coverage

907. A regression suite has run unchanged for two years and now rarely finds new defects. Which ISTQB principle explains this? Understanding

A. Tests wear out (the pesticide paradox)
B. Defects cluster together
C. Testing is context dependent
D. Absence-of-defects fallacy

908. In testing, what is a test oracle? Understanding

A. A tool that generates random test inputs
B. The person who approves the test plan
C. A source used to determine the expected result of a test
D. A database that stores test execution logs

909. Under the ISTQB CTFL v4.0 test levels, an online shop is tested to check that its order service exchanges messages correctly with an external bank's payment gateway. Which level is this? Understanding

A. Component integration testing
B. Component testing
C. System integration testing
D. Acceptance testing

910. A team finds that 80% of last release's defects came from two of its twenty modules. Which testing principle does this illustrate? Understanding
    A. Defects cluster together
    B. Exhaustive testing is impossible
    C. Early testing saves time and money
    D. Tests wear out

911. In the V-model, integration tests are designed against which development artefact? Understanding
    A. The user requirements
    B. The architectural (high-level) design
    C. The detailed module design
    D. The source code

912. Checking that a login page answers within 2 seconds while 500 users log in at once is an example of: Understanding
    A. Functional testing
    B. Non-functional (performance) testing
    C. White-box testing
    D. Confirmation testing

913. In mutation testing, which statement about an equivalent mutant is correct? Understanding
    A. It is killed by every test in the suite
    B. It behaves identically to the original program for every input, so no test can kill it
    C. It is a mutant that causes a compilation error
    D. It is a duplicate of another mutant and is counted twice

914. An age field accepts whole numbers from 18 to 60 inclusive. Using equivalence partitioning, how many partitions should you identify, counting valid and invalid ones? Application
    A. 2
    B. 4
    C. 43
    D. 3

915. A field accepts integers from 18 to 60 inclusive. Which set gives 2-value boundary value analysis (BVA) coverage? Application
    A. 17, 18, 60, 61
    B. 18, 60
    C. 17, 18, 19, 59, 60, 61
    D. 0, 18, 60, 100

916. A field accepts integers from 1 to 100. How many test values does 3-value boundary value analysis require? Application
    A. 6
    B. 4
    C. 3
    D. 8

917. Income tax rules: income 0 to 50,000 pays 0%, 50,001 to 100,000 pays 5%, above 100,000 pays 10%, and negative income is invalid. What is the minimum number of tests to cover every equivalence partition once? Application
    A. 3
    B. 6
    C. 4
    D. 8

918. For the same tax rules (invalid below 0; bands 0 to 50,000, 50,001 to 100,000 and above 100,000), which set is the 2-value boundary set? Application

A. -1, 0, 50000, 50001, 100000, 100001
B. 0, 50000, 100000
C. -1, 0, 50000, 100000, 100001
D. 0, 25000, 50000, 75000, 100000, 150000

919. The function below is tested with one input, m = 70. What statement coverage does this achieve? Application

```
def grade(m):
    if m < 0 or m > 100:
        return "invalid"
    result = "fail"
    if m >= 50:
        result = "pass"
    return result
```

A. 100%
B. 50% (3 of 6 statements)
C. 66.7% (4 of 6 statements)
D. 83.3% (5 of 6 statements)

920. The same grade function is tested only with m = 70. Treating each if as one decision with two outcomes, what branch coverage does it achieve? Application

```
def grade(m):
    if m < 0 or m > 100:
        return "invalid"
    result = "fail"
    if m >= 50:
        result = "pass"
    return result
```

A. 50%
B. 83.3%
C. 25%
D. 75%

921. What is the minimum number of test inputs needed for 100% branch coverage of the grade function? Application

```
def grade(m):
    if m < 0 or m > 100:
        return "invalid"
    result = "fail"
    if m >= 50:
        result = "pass"
    return result
```

A. 3
B. 2
C. 4
D. 6

922. What is the cyclomatic complexity V(G) of this function? Application

```
def classify(n):
    count = 0
    while n > 0:
        if n % 2 == 0:
            count += 1
        n -= 1
    return count
```

A. 3
B. 2
C. 4
D. 6

923. A connected control-flow graph has 11 edges and 9 nodes. What is its cyclomatic complexity? Application

A. 4
B. 2
C. 3
D. 20

924. A decision table has three independent conditions, each true or false. How many rules (columns) does the full, unreduced table have? Application

A. 6
B. 3
C. 9
D. 8

925. An ATM model has these valid transitions: Idle to CardIn (insert); CardIn to Authenticated (valid PIN); CardIn to CardIn (1st or 2nd wrong PIN); CardIn to Blocked (3rd wrong PIN); CardIn to Idle (cancel); Authenticated to Idle (eject). How many transitions must be exercised for 100% valid-transition (0-switch) coverage? Application

A. 4
B. 5
C. 6
D. 3

926. A mutation run creates 50 mutants. 5 are found to be equivalent, and the test suite kills 36. What is the mutation score? Application

A. 72%
B. 82%
C. 90%
D. 80%

927. Module A calls modules B and C, and B calls D. Using top-down integration, you first test A on its own. How many stubs do you need for that first test? Application

A. 1
B. 2
C. 3
D. 0

928. The function has one existing test, shipping(12, True). Which single extra test raises branch coverage to 100%? Analysis

```
def shipping(weight, express):
    cost = 5
    if weight > 10:
        cost += 3
    if express:
        cost *= 2
    return cost
```

A. shipping(15, True)
B. shipping(5, False)
C. shipping(15, False)
D. shipping(8, True)

929. discount(1500) is the only test of this function. Which conclusion is correct? Analysis

```
def discount(total):
    rate = 0
    if total >= 1000:
        rate = 10
    return total - total * rate / 100
```

A. Statement and branch coverage are both 100%
B. Statement coverage is 100%, but the False outcome of the if is never exercised
C. Branch coverage is 100%, but one statement is not executed
D. Statement coverage is 50% because only one branch ran

930. Which single input achieves 100% branch coverage of this function? Analysis

```
def classify(n):
    count = 0
    while n > 0:
        if n % 2 == 0:
            count += 1
        n -= 1
    return count
```

A. n = 0
B. n = 1
C. n = 2
D. n = -1

931. A field accepts integers from 1 to 100. Which test set achieves full 2-value BVA coverage and also covers every equivalence partition, using the fewest values? Analysis

A. {1, 50, 100}
B. {0, 50, 101}
C. {-1, 1, 99, 101}
D. {0, 1, 100, 101}

932. Ayesha misreads the fee rule and types > instead of >= in the code. Weeks later a customer exactly on the threshold is charged the wrong fee. In ISTQB terms, what is the failure? Analysis

A. Ayesha's misreading of the rule
B. The > operator in the source code
C. The bug report the customer files
D. The wrong fee shown to the customer

933. In a payroll system, the lowest-level module that talks to the bank's file format is new, complex and considered the riskiest part. Which integration approach tests it earliest? Analysis

A. Top-down integration, using stubs
B. Big-bang integration of all modules at once
C. Bottom-up integration, using drivers
D. Acceptance testing by the payroll staff

934. A bug fix in the discount module passed its confirmation test, but after release the checkout total broke in a module nobody touched. Which testing activity was most likely skipped? Analysis

A. Confirmation testing
B. Regression testing
C. Static analysis of the discount module
D. Alpha testing of the discount screen

935. What is the cyclomatic complexity of fee(), and what does it tell you? Analysis

```
def fee(items, member):
    total = 0
    for p in items:
        if p > 100:
            total += p * 0.9
        elif p > 50:
            total += p * 0.95
        else:
            total += p
    if member:
        total -= 10
    return total
```

A. 4; at least 4 tests are needed for statement coverage
B. 3; the for loop does not count as a decision
C. 5; a basis set has 5 independent paths
D. 6; each assignment adds one path

936. Using the same fee() function (V(G) = 5), what is the minimum number of test cases for 100% branch coverage? Analysis

```
def fee(items, member):
    total = 0
    for p in items:
        if p > 100:
            total += p * 0.9
        elif p > 50:
            total += p * 0.95
        else:
            total += p
    if member:
        total -= 10
    return total
```

A. 5
B. 3
C. 4
D. 2

937. A lead says: 'All 4,000 automated tests passed, so this release has no defects.' What is wrong with this claim? Analysis

A. Nothing, because 4,000 tests amount to exhaustive testing
B. The claim is wrong only because 4,000 is too few; 40,000 would prove it
C. Passing tests show only that the tested cases work; they cannot prove that no defects remain
D. Automated tests cannot find defects, only manual tests can

938. Why does ISTQB say exhaustive testing is impossible for all but trivial software? Understanding

A. Testers are not allowed to see the source code
B. Automated tools cannot run more than a few thousand tests
C. The number of input and precondition combinations is far too large, so tests must be chosen by risk and priority
D. Every program contains at least one equivalent mutant

939. The ISTQB principle 'early testing saves time and money' means: Understanding

A. Testing should finish as early as possible so that release is not delayed
B. Only the first build of a system needs testing
C. Defects removed early, for example by reviewing requirements, cost less to fix than ones found later
D. Testers should be hired before the developers

940. Alpha testing is best described as: Understanding

A. Acceptance testing by real users at their own sites before general release
B. The first unit tests written for a new class
C. Acceptance testing by users or an independent test team at the developer's site
D. Testing done by an automated tool without human input

941. A quick set of tests runs on every new build to check that the main functions work before deeper testing begins. What is it called? Understanding

A. Regression testing
B. Exploratory testing
C. Smoke testing
D. Soak testing

942. Boehm's question 'Are we building the right product?' describes: Understanding

A. Verification
B. Debugging
C. Validation
D. Static analysis

943. In a defect report, how does severity differ from priority? Understanding

A. Severity is how urgently it should be fixed; priority is the defect's impact
B. Severity is the defect's impact on the system; priority is how urgently it should be fixed
C. They are two names for the same field
D. Severity is set by developers and priority by the test tool

944. In ISTQB terms, what is the test basis? Understanding

A. The body of knowledge, such as requirements or design, used to analyse and design tests
B. The minimum number of tests needed for statement coverage
C. The lowest test level in the V-model
D. The tool that runs the automated tests

945. In the V-model, acceptance tests are planned against which artefact? Understanding

A. The architectural design
B. The detailed module design
C. The source code
D. The user requirements

946. Which of these is debugging rather than testing? Understanding

A. Running a test suite and recording which tests fail
B. Designing boundary values for an input field
C. Finding the defect that causes a failure and fixing it
D. Reviewing a requirements document for ambiguities

947. A developer tests one class on its own, calling its methods directly with fake objects in place of its collaborators. Which test level is this? Understanding

A. Component (unit) testing
B. System testing
C. Acceptance testing
D. System integration testing

948. Which ISTQB principle says that a heart-monitor system and a mobile game should be tested differently? Understanding

A. Defects cluster together
B. Testing is context dependent
C. Tests wear out
D. Early testing saves time and money

949. A system passes every test and has no known defects, yet users reject it because it does not support the way they actually work. Which ISTQB principle does this illustrate? Understanding

A. Defects cluster together
B. Exhaustive testing is impossible
C. Tests wear out
D. The absence-of-defects fallacy

950. What is the main drawback of big-bang integration? Understanding

A. It needs both stubs and drivers for every module
B. It can only be used with object-oriented code
C. When a failure appears, it is hard to locate the faulty interface because all modules were combined at once
D. It tests the user interface before any lower modules exist

951. Sandwich (hybrid) integration testing combines: Understanding

A. Unit testing and acceptance testing in one phase
B. Black-box and white-box techniques in one test case
C. Big-bang integration of the top and bottom layers only
D. Top-down and bottom-up integration, meeting at a middle layer

952. A tester designs and runs tests at the same time, learning about the product as they go and adjusting the next test to what they just saw. What is this approach called? Understanding

A. Scripted regression testing
B. Static testing
C. Equivalence partitioning
D. Exploratory testing

953. A tester uses experience of common developer mistakes, such as division by zero, empty lists and missing files, to decide what to test. ISTQB calls this technique: Understanding

A. Equivalence partitioning
B. Error guessing
C. Boundary value analysis
D. Statement testing

954. How does stress testing differ from load testing? Understanding

A. Load testing goes beyond the limits; stress testing stays within expected load
B. Load testing checks behaviour at expected and peak loads; stress testing goes beyond those limits to see how the system fails and recovers
C. Stress testing is functional testing; load testing is structural
D. They differ only in which tool is used

955. In the test pyramid, which kind of test should be the most numerous? Understanding

A. End-to-end UI tests
B. Manual exploratory sessions
C. Unit (component) tests
D. User acceptance tests

956. Which arrangement gives the highest level of test independence? Understanding

A. The author tests their own code
B. A colleague from the same development team tests the code
C. Two developers test each other's code during pair programming
D. Testers from outside the organisation

957. How does static testing differ from dynamic testing in what it finds? Understanding

A. Static testing finds only spelling mistakes; dynamic testing finds all logic errors
B. Static testing finds defects directly in work products; dynamic testing finds failures, whose causing defects must then be located
C. Static testing needs the code to run; dynamic testing does not
D. They find exactly the same defects at the same cost

958. Exit criteria for a test level are best described as: Understanding

A. The list of testers allowed to leave the project
B. Conditions, such as coverage reached and no open critical defects, that must be met before that level is finished
C. The last test case in the test procedure
D. The conditions under which a test case is skipped

959. According to ISTQB CTFL, which events trigger maintenance testing of a system already in use? Understanding

A. Only planned feature enhancements
B. Only defects reported by users
C. Code reviews, walkthroughs and inspections
D. Modifications, migration to a new environment, and retirement of the system

960. How does a mock object differ from a simple stub in unit testing? Understanding

A. A stub records and verifies calls; a mock only returns canned answers
B. A mock replaces the module under test; a stub replaces the test
C. A mock also records and verifies the calls made to it; a stub only returns canned answers
D. A stub is used only in system testing; a mock only in unit testing

961. What does modified condition/decision coverage (MC/DC) require? Understanding

A. Each condition in a decision must be shown to affect the decision's outcome independently
B. Each decision must take both True and False outcomes, and nothing more
C. Every possible combination of condition values must be tested
D. Every path through the program must be executed

962. Module A calls B and C, and C calls D and E. With bottom-up integration, which modules are tested first? Application

A. A on its own, using stubs for B and C
B. C on its own, using stubs for D and E
C. A, B and C together, since they are at the top
D. The leaf modules B, D and E, using drivers

963. A function takes two independent 16-bit unsigned integers. How many input combinations would exhaustive testing need? Application

A. 131,072 (2 × 2^16)
B. 4,294,967,296 (2^32)
C. 65,536 (2^16)
D. 1,024 (32^2)

964. Voters must be at least 18 years old, counted in whole years, and the rule sets no upper limit. Which pair is the 2-value boundary set? Application

A. 18 and 19
B. 17 and 19
C. 17 and 18
D. 0 and 18

965. A vending machine's response depends on its current mode (idle, coin inserted, dispensing) and on the event that occurs. Which black-box technique fits best? Application

A. State transition testing
B. Equivalence partitioning
C. Statement testing
D. Pairwise testing

966. What is the minimum number of tests for 100% statement coverage of this function? Application

```
def sign(n):
    if n >= 0:
        s = "non-negative"
    else:
        s = "negative"
    return s
```

A. 1
B. 3
C. 2
D. 4

967. Rule: delivery is free if the order is at least Rs 2,000 or the customer is premium. A full decision table for these two conditions has 4 rules. How many of them give free delivery? Application

A. 1
B. 3
C. 2
D. 4

968. A price field accepts values from 0.01 to 999.99, entered with exactly two decimal places. Which set is the 2-value boundary set? Application

A. 0, 1, 999 and 1000
B. 0.00, 0.01, 999.99 and 1000.00
C. 0.01, 0.02, 999.98 and 999.99
D. 0.00, 0.01, 0.02, 999.98, 999.99 and 1000.00

969. A state table has 4 states and 3 possible events. Only 5 state-event pairs lead to valid transitions; all the others are invalid. How many invalid state-event pairs can negative tests target? Application

A. 12
B. 5
C. 9
D. 7

970. The only test is ticket(30, True). Counting the elif as a statement, what statement coverage does it achieve? Application

```
def ticket(age, student):
    price = 1000
    if age < 12:
        price = 500
    elif student:
        price = 700
    if age >= 60:
        price -= 200
    return price
```

A. 62.5% (5 of 8)
B. 87.5% (7 of 8)
C. 75% (6 of 8)
D. 50% (4 of 8)

971. Using the same ticket() function, the tests are ticket(5, False) and ticket(30, True). Treating each if and the elif as a decision with two outcomes, what branch coverage do they achieve? Application

```
def ticket(age, student):
    price = 1000
    if age < 12:
        price = 500
    elif student:
        price = 700
    if age >= 60:
        price -= 200
    return price
```

A. 66.7% (4 of 6)
B. 50% (3 of 6)
C. 83.3% (5 of 6)
D. 100% (6 of 6)

972. System testing found 160 defects. Another 40 were found in the first three months of live use. What was the system test's defect detection percentage (DDP)? Application

A. 25%
B. 80%
C. 75%
D. 20%

973. This function has three independent if statements in sequence, none with an else. How many distinct paths run through it? Application

```
def adjust(a, b, c):
    x = 0
    if a:
        x += 1
    if b:
        x += 2
    if c:
        x += 4
    return x
```

A. 8
B. 4
C. 6
D. 3

974. Pressman's simple-loop testing is applied to a loop that allows at most n = 10 passes. Which set of pass counts matches it? Application

A. Only 1 and 10
B. 0, 10 and 20
C. 1 to 10, every value
D. 0, 1, 2, a typical value such as 5, then 9, 10 and 11

975. A program's control-flow graph has two separate connected components (a main routine and one subroutine), with 20 edges and 18 nodes in total. Using V(G) = E - N + 2P, what is the cyclomatic complexity? Application

A. 4
B. 3
C. 8
D. 6

976. When this test class runs, what is the result? Application

```
import unittest

class TestTotals(unittest.TestCase):
    def test_sum(self):
        self.assertEqual(0.1 + 0.2, 0.3)

    def test_close(self):
        self.assertAlmostEqual(0.1 + 0.2, 0.3)
```

A. Both tests pass
B. test_sum fails and test_close passes
C. Both tests fail
D. test_sum passes and test_close fails

977. A form has two fields. Each field has one valid partition and two invalid partitions. Invalid partitions are tested one at a time, with every other field valid. What is the minimum number of tests that covers every partition? Application

A. 5
B. 3
C. 4
D. 9

978. Before testing, 20 defects are deliberately seeded into a program. Testing finds 15 of the seeded defects and 45 real ones. Using the defect-seeding estimate, about how many real defects remain undetected? Application

A. 15
B. 60
C. 5
D. 45

979. Three parameters each have two values: browser (Chrome, Firefox), OS (Windows, Linux) and role (admin, user). What is the minimum number of tests that covers every pair of values for every pair of parameters? Application

A. 8
B. 6
C. 4
D. 2

980. What is the minimum number of tests that achieves MC/DC for the decision if a and b:? Application

A. 2
B. 4
C. 3
D. 1

981. The tests are access(True, False) and access(False, True). What condition coverage and decision coverage do they achieve? Application

```
def access(is_admin, is_owner):
    if is_admin or is_owner:
        return "allow"
    return "deny"
```

A. Condition coverage 100%, decision coverage 50%
B. Condition coverage 100%, decision coverage 100%
C. Condition coverage 50%, decision coverage 100%
D. Condition coverage 50%, decision coverage 50%

982. A mutation tool changes >= to > in this function. Which test kills the mutant? Application

```
def eligible(cgpa):
    return cgpa >= 2.5
```

A. assert eligible(3.0) is True
B. assert eligible(2.0) is False
C. assert eligible(2.6) is True
D. assert eligible(2.5) is True

983. Which test set achieves 100% branch coverage of shipping_fee() with the fewest tests? Analysis

```
def shipping_fee(weight, city):
    fee = 200
    if weight > 5:
        fee += 100
    if city == "remote":
        fee += 150
    if weight > 20:
        fee = fee * 2
    return fee
```

A. (25, "remote") and (2, "local")
B. (25, "remote") and (10, "local")
C. (2, "remote"), (10, "local") and (3, "local")
D. (25, "remote") alone

984. This test runs every line of Cart.total() and the coverage report shows 100%. Why is it still a weak test? Analysis

```
def test_total():
    cart = Cart()
    cart.add(price=500, qty=3)
    cart.total()
```

A. It should use a price of 0 to reach 100% coverage
B. Tests must never create objects directly
C. 100% coverage proves total() is correct, so it is not weak
D. It has no assertion, so it passes whatever total() returns; coverage measures execution, not checking

985. Spec: A for 85 marks and above, B for 70 to 84, C below 70. Which single 2-value BVA test reveals the defect in this code? Analysis

```
def grade_band(marks):
    if marks > 85:
        return "A"
    if marks >= 70:
        return "B"
    return "C"
```

A. marks = 84
B. marks = 85
C. marks = 70
D. marks = 69

986. An insurance premium depends on age band, smoker status and previous claims, and the business rules give a different premium for particular combinations of these. Which technique best ensures every combination is considered? Analysis

A. Decision table testing
B. Boundary value analysis
C. State transition testing
D. Statement testing

987. A company wants feedback from 200 real customers using a new app on their own phones and networks before general launch. Which testing fits? Analysis

    A. Alpha testing
    B. Component testing
    C. Static testing
    D. Beta testing

988. Management wants an early working demo of a new system's menus and screens, while the lower-level modules are still being written. Which integration approach fits? Analysis

    A. Bottom-up integration, using drivers
    B. Big-bang integration once every module is complete
    C. No integration testing until acceptance
    D. Top-down integration, using stubs for the unfinished lower modules

989. A test passes when run on its own but fails randomly when the full suite runs. Investigation shows it reads a shared database row that another test changes. What is the best fix? Analysis

    A. Mark the test as 'skip' so the suite stays green
    B. Make each test set up and clean up its own data so that tests are independent of order
    C. Re-run the suite until it passes, then release
    D. Run the test only in production

990. A mutation tool changes balance -= 50 to balance += 50, and test_fee still passes. What does the surviving mutant reveal? Analysis

```
def apply_fee(balance):
    if balance < 1000:
        balance -= 50
    return balance

def test_fee():
    assert apply_fee(500) < 1000
```

    A. The assertion is too weak; checking the exact result, assert apply_fee(500) == 450, would kill the mutant
    B. The mutant is equivalent, so it can be ignored
    C. The code has a defect, because the mutant passed
    D. Branch coverage is below 100%, so more inputs are needed

991. The bank's name is misspelt on the home page of its mobile app; every function works correctly. How would most teams classify this defect? Analysis

    A. High severity, high priority
    B. High severity, low priority
    C. Low severity, low priority
    D. Low severity, high priority

992. A tester enters an invalid age (-5) and an invalid email ('abc') in the same test, and the form is rejected. Why is this a weak test? Analysis

    A. One rejection can mask the other: the email check may be broken but never reached, so invalid partitions should be tested one at a time
    B. Negative numbers cannot be used as test data
    C. It is a strong test, because two partitions are covered at once
    D. Email fields can only be tested with boundary values

993. Spec: score >= 80 earns 10%, and 5 or more years adds 5 percentage points. Tests bonus(90, 6) and bonus(50, 1) both pass. Which conclusion is correct? Analysis

```
def bonus(score, years):
    rate = 0
    if score >= 80:
        rate = 10
    if years >= 5:
        rate = 15
    return rate
```

A. Branch coverage is 100%, but the path score < 80 with years >= 5 never runs, and there the code returns 15 instead of 5
B. Branch coverage is 100%, so the function is correct
C. Branch coverage is only 50%, which is why a defect was missed
D. The defect appears in bonus(90, 1), which returns 15

994. A change touches only the currency-conversion function, which is called by checkout and refunds. The full regression suite takes 3 days, and there is 1 day. What is the best plan? Analysis

A. Run only the confirmation test for the change
B. Use impact analysis: test the changed function, then checkout and refunds, then add a risk-prioritised subset of the remaining tests
C. Run the first third of the regression suite in its usual order
D. Delay the release until the full 3-day suite can run, whatever the business impact

995. Which of these mutants of total() is an equivalent mutant? Analysis

```
def total(items):
    s = 0
    i = 0
    while i < len(items):
        s += items[i]
        i += 1
    return s
```

A. Change i < len(items) to i != len(items)
B. Change s = 0 to s = 1
C. Change i += 1 to i += 2
D. Change i < len(items) to i <= len(items)

996. A developer misread a requirement, then wrote both the code and its unit tests from that misreading. All the tests pass. Why did testing miss the defect? Analysis

A. Unit tests can never find defects
B. The tests share the code's wrong assumption; an independent tester working from the specification would be more likely to catch it
C. The tests should have used more random data
D. Passing tests prove that the requirement was misread correctly

997. A team's automated suite is 80% end-to-end UI tests with few unit tests. A full run takes 6 hours and often fails randomly. What is the best long-term direction? Analysis

A. Move most checks down to fast unit and service-level tests, keeping only a few key end-to-end journeys
B. Add more end-to-end UI tests so the random failures average out
C. Run the suite less often, once a month
D. Delete the unit tests, since UI tests already cover the code

Software Maintenance & Evolution MCQs for NSCT

998. A user reports that the app crashes when a form is submitted empty, and the team fixes it. Which type of maintenance is this? Understanding
    A. Adaptive
    B. Perfective
    C. Preventive
    D. Corrective

999. Changing a system so that it keeps working after the company moves to a new operating system version is: Understanding
    A. Corrective maintenance
    B. Perfective maintenance
    C. Adaptive maintenance
    D. Preventive maintenance

1000. Improving a working report so that it runs twice as fast, at users' request, is mainly which type of maintenance? Understanding
    A. Perfective
    B. Corrective
    C. Adaptive
    D. Preventive

1001. During a code review, a developer notices a date counter that will overflow in 2038 and fixes it before any user is affected. Under ISO/IEC 14764 this is: Understanding
    A. Corrective maintenance
    B. Preventive maintenance
    C. Adaptive maintenance
    D. Perfective maintenance

1002. Which description best fits a legacy system? Understanding
    A. Any system that has been released to customers
    B. A prototype that was thrown away after requirements were clear
    C. An older system, often built with obsolete technology, that is still critical to the business
    D. A new system written in a legacy-free language

1003. ISO/IEC 14764 groups the four maintenance types into two classes. Which grouping is correct? Understanding
    A. Correction: corrective and preventive; Enhancement: adaptive and perfective
    B. Correction: corrective and adaptive; Enhancement: perfective and preventive
    C. Correction: corrective only; Enhancement: adaptive, perfective and preventive
    D. Correction: preventive and perfective; Enhancement: corrective and adaptive

1004. Lehman's first law, continuing change, states that an E-type system: Understanding
    A. Must be continually adapted, or it becomes progressively less satisfactory
    B. Becomes simpler with every release
    C. Can be frozen once it meets its original specification
    D. Grows only when new developers join the team

1005. According to Lehman's law of increasing complexity, what happens as an evolving system is changed? Understanding
    A. Complexity stays constant because each change is local
    B. Complexity falls because developers learn the code
    C. Complexity rises only when the programming language changes
    D. Its structure tends to degrade and complexity rises, unless work is done to maintain or reduce it

1006. Lehman's laws of software evolution are formulated mainly for which kind of program in his S, P and E classification? Understanding
    A. S-type programs, which are fully defined by a formal specification
    B. P-type programs, which solve a precisely stated problem such as chess
    C. All three types equally, with no difference
    D. E-type programs, which are embedded in and interact with a real-world environment

1007. Which Lehman law says that, over a system's lifetime, the amount of change in each release stays roughly constant? Understanding
    A. Continuing growth
    B. Conservation of familiarity
    C. Self-regulation
    D. Declining quality

1008. In Chikofsky and Cross's taxonomy, what is reverse engineering? Understanding
    A. Rewriting a system in a new language to improve performance
    B. Moving from requirements to design to code
    C. Analysing a system to identify its components and relationships and create higher-level representations, without changing it
    D. Removing features that users no longer need

1009. Which statement best describes software re-engineering? Understanding
    A. Examining and altering an existing system to rebuild it in a new form, then implementing that new form
    B. Building a brand-new system from fresh requirements and discarding the old one
    C. Only drawing UML diagrams from old code
    D. Replacing hardware while leaving the software untouched

1010. Converting unstructured code full of GOTO jumps into structured code, at the same level of abstraction and with the same external behaviour, is called: Understanding
    A. Reverse engineering
    B. Forward engineering
    C. Data re-engineering
    D. Restructuring

1011. Lehman's law of declining quality states that: Understanding
    A. Quality declines only when the number of users falls
    B. The quality of an E-type system will appear to decline unless it is rigorously maintained and adapted to its changing environment
    C. Quality always improves with each release because more defects are fixed
    D. Quality is fixed at delivery and cannot change

1012. Lehman's eighth law describes evolution processes as: Understanding
    A. Simple linear processes controlled by the project manager
    B. Multi-level, multi-loop, multi-agent feedback systems
    C. Random processes with no measurable pattern
    D. Processes that stop once the system is fully documented

1013. Sommerville lists factors that make maintenance costly compared with development. Which one is among them? Understanding
    A. Maintenance work always uses newer languages than development
    B. Maintenance is done before requirements are known
    C. Maintained systems have fewer users than new systems
    D. Team stability: after delivery the original team often disbands, and new staff must first understand the system

1014. A banking app must be updated because Android 15 removes an API it depends on. Classify this change. Application
    A. Corrective
    B. Perfective
    C. Preventive
    D. Adaptive

1015. Users ask for an 'Export to PDF' button on a reporting screen that already works correctly. Adding it is: Application
    A. Corrective maintenance
    B. Perfective maintenance
    C. Adaptive maintenance
    D. Preventive maintenance

1016. A 120 KLOC system took 600 person-months (PM) to develop. Each year about 9 KLOC are added and 6 KLOC are modified. Using Boehm's annual change traffic (ACT), what is the estimated annual maintenance effort? Application
    A. 75 PM
    B. 45 PM
    C. 30 PM
    D. 90 PM

1017. A 200 KLOC product has 12 KLOC added and 8 KLOC changed in a year. What is its annual change traffic (ACT)? Application
    A. 0.06
    B. 0.10
    C. 0.04
    D. 0.20

1018. A system cost 2.0 million to develop and then 0.3 million a year to maintain for 10 years. What share of the lifetime cost went on maintenance? Application
    A. 15%
    B. 30%
    C. 60%
    D. 150%

1019. A team's 20 change requests last quarter were: 6 bug fixes after failures, 2 fixes of latent faults found in review, 4 changes for a new database version and 8 user-requested improvements. What percentage was enhancement, as ISO/IEC 14764 defines it? Application
    A. 60%
    B. 40%
    C. 50%
    D. 20%

1020. Module dependencies (X -> Y means X uses Y): UI -> Orders, UI -> Reports, Orders -> Pricing, Orders -> DB, Reports -> DB, Pricing -> Tax, Billing -> Pricing. If Tax changes, which modules are in its impact set? Application

A. Pricing, Orders, Billing and UI
B. Pricing only
C. Pricing, Orders and Billing
D. Pricing, Orders, Billing, UI and Reports

1021. Dependencies (X -> Y means X uses Y): Cart -> Auth, Checkout -> Cart, Checkout -> Payment, Payment -> Auth, Admin -> Auth, Email -> Checkout. Payment is changed. How many modules, including Payment itself, should be regression-tested for ripple effects? Application

A. 2
B. 5
C. 3
D. 6

1022. Sommerville assesses legacy systems on business value and system quality. What is the usual recommendation for a low-quality, high-business-value system? Application

A. Scrap it completely
B. Continue normal maintenance unchanged
C. Freeze it and stop all changes
D. Re-engineer it, or replace it if a suitable system is available

1023. Using Sommerville's value-quality assessment, what should normally happen to a legacy system with low quality and low business value? Application

A. Re-engineer it
B. Continue normal maintenance
C. Scrap it
D. Add new features to raise its value

1024. A maintainability study logs the days taken to implement five change requests: 4, 10, 6, 12 and 8. What is the mean time to implement a change? Application

A. 6 days
B. 8 days
C. 10 days
D. 40 days

1025. A team uses a tool to generate class and sequence diagrams from an undocumented Java system, without changing any code. This activity is: Application

A. Forward engineering
B. Refactoring
C. Reverse engineering
D. Data re-engineering

1026. An old system stores customer records in flat files with three different date formats. The team cleans the data, settles one format and migrates it into a relational database. This is: Application

A. Program modularisation
B. Source code translation
C. Reverse engineering only
D. Data re-engineering

1027. Pressman's software re-engineering process model is a cycle of six activities. Which activity normally starts it? Application

A. Forward engineering
B. Code restructuring
C. Inventory analysis
D. Data restructuring

1028. A maintenance log labels four changes. Which label is wrong under ISO/IEC 14764? Analysis

A. Fixing a crash on empty input reported by a customer: corrective
B. Porting a module to Python 3 because Python 2 is end-of-life: adaptive
C. Fixing an overflow found in code review before any user hit it: corrective
D. Adding dark mode requested by users: perfective

1029. The last three releases added 30, 35 and 32 new modules. Management now plans one release adding 90 modules. Which Lehman law warns against this, and why? Analysis

A. Continuing growth: functionality must grow, so a large release is expected
B. Self-regulation: the release will shrink itself back to normal size
C. Declining quality: quality falls only when no release is made
D. Conservation of familiarity: changes far above the usual increment tend to bring many faults

1030. A bank's core system is poor in code quality but runs every transaction, and the bank cannot afford a long outage. Which strategy fits best? Analysis

A. Re-engineer or replace it incrementally, piece by piece, while the old system keeps running
B. Scrap it immediately because its quality is low
C. Do a single big-bang rewrite and switch over in one weekend
D. Leave it unchanged, since high-value systems never need work

1031. Dependencies (X -> Y means X uses Y): Ledger -> Rates, Rates -> Ledger, Report -> Ledger, Audit -> Report, Export -> Rates, and Users has no dependencies. Rates changes. Which modules, apart from Rates, are in the impact set? Analysis

A. Ledger and Export only
B. Ledger, Report, Audit and Export
C. Ledger, Report, Audit, Export and Users
D. Report and Audit only

1032. Re-engineering a system costs 300 PM and cuts annual maintenance from 100 PM to 40 PM. Ignoring inflation and risk, after how many years does re-engineering pay for itself? Analysis

A. 5 years
B. 3 years
C. 7.5 years
D. 2 years

1033. During one sprint a developer renames unclear variables and extracts long methods into smaller ones, without changing behaviour or architecture. Which term fits best? Analysis

A. Re-engineering
B. Reverse engineering
C. Adaptive maintenance
D. Refactoring

1034. Release after release, a product's average cyclomatic complexity rises while its features barely change, and each change now takes longer. Which Lehman law is at work, and what is the remedy? Analysis

A. Continuing growth; add more features to keep users happy
B. Increasing complexity; plan effort for preventive restructuring and refactoring
C. Conservation of organisational stability; hire more developers
D. Self-regulation; the trend will reverse without action

1035. The original developers of a 15-year-old system have left and there is no design documentation. Where will most effort on a new change request probably go, and what helps most? Analysis

A. Typing the new code; faster keyboards and IDE shortcuts help most
B. Writing user manuals; technical writers help most
C. Understanding the existing code; reverse engineering and redocumentation help most
D. Compiling the system; a faster build server helps most

1036. At 2 a.m. a payment service fails, and an engineer applies an unscheduled hotfix to keep it running until a proper fix is released. How does ISO/IEC 14764 classify this? Analysis

A. Emergency maintenance, a temporary unscheduled form of corrective maintenance
B. Preventive maintenance, because it prevents further failures
C. Adaptive maintenance, because production is an environment
D. Perfective maintenance, because the service is improved

1037. Before implementing a change to the tax calculation, the team wants to estimate which modules, tests and documents will be affected. Which activity does this? Analysis

A. Inventory analysis
B. Change impact analysis
C. Forward engineering
D. Beta testing

1038. Which of Lehman's laws states that the functional content of an E-type system must keep increasing to maintain user satisfaction over its lifetime? Understanding

A. Continuing growth
B. Continuing change
C. Self-regulation
D. Conservation of familiarity

1039. In Lehman's SPE classification, a program whose correctness is judged entirely against a complete formal specification, such as a routine that sorts integers, is: Understanding

A. An E-type program
B. A P-type program
C. An S-type program
D. A legacy program

1040. Which program best fits Lehman's P-type category? Understanding

A. A function that computes the greatest common divisor exactly as specified
B. A hospital management system used daily by staff whose work it changes
C. A chess-playing program: the problem is precisely stated, but a practical solution can only approximate the best play
D. A compiler's lexical analyser defined by a regular-expression specification

1041. In Chikofsky and Cross's taxonomy, what is forward engineering? Understanding

A. Recovering designs from existing code without changing it
B. Moving from high-level abstractions and designs down to a physical implementation
C. Transforming code into a different form at the same level of abstraction
D. Adding comments and documents to an existing program

1042. Chikofsky and Cross describe redocumentation as: Understanding

A. Rewriting the code in a new language while keeping its documents
B. Replacing all old documents with a new requirements specification written from scratch
C. Moving documents from paper into a configuration management tool
D. Creating or revising a semantically equivalent representation at the same level of abstraction, a simple form of reverse engineering

1043. How does design recovery differ from basic reverse engineering? Understanding

A. It changes the code so that it matches the recovered design
B. It works only on databases, not on programs
C. It adds domain knowledge and external information to the code analysis to identify meaningful higher-level abstractions
D. It is done only by the original developers

1044. Fowler's code smell called shotgun surgery means: Understanding

A. One logical change forces many small edits spread across many classes
B. One class is changed for many unrelated reasons
C. A method uses another class's data more than its own
D. Several methods share the same long list of parameters

1045. During a legacy replacement, the old and new systems process the same live inputs side by side for several weeks, and their outputs are compared before the old one is switched off. This cut-over approach is called: Understanding

A. Parallel running
B. Big-bang replacement
C. Reverse engineering
D. Code restructuring

1046. What is wrapping, as a legacy modernisation technique? Understanding

A. Compressing the legacy source code to save disk space
B. Rewriting every legacy module in a new language
C. Placing a modern interface, such as a web service, around legacy code so new clients can use it without changing its internals
D. Deleting unused legacy features before migration

1047. Why is regression testing essential after every maintenance change? Understanding

A. To measure how fast the new feature runs
B. To test only the lines that were changed
C. To replace the need for an impact analysis
D. To check that the change has not broken features that worked before

1048. In Rajlich and Bennett's staged model of the software life cycle, which is cited by Sommerville, what characterises the servicing stage? Understanding

A. Major new features and architectural changes are added freely
B. The system is being specified and designed for the first time
C. Only small, tactical changes such as fixes and minor adaptations are made, because major change has become too costly
D. The system has been switched off and its data archived

1049. Which activity in the ISO/IEC 14764 maintenance process covers moving existing software to a new operational environment? Understanding

A. Retirement
B. Problem and modification analysis
C. Process implementation
D. Migration

1050. In maintenance, the ripple effect refers to: Understanding

A. The steady growth in user numbers after each release
B. The delay between a bug report and its fix
C. Secondary changes and faults that spread to other parts of the system as a result of one change
D. The time taken to deploy a patch to all servers

1051. Surveys from Lientz and Swanson onwards report which kind of work as taking the largest share of maintenance effort? Understanding

A. Corrective fixes of reported failures
B. Preventive fixes of latent faults
C. Emergency patches made outside the normal schedule
D. Enhancements: adaptive and perfective changes such as new or changed features

1052. What does Lehman's law of self-regulation say about E-type system evolution? Understanding

A. Developers regulate themselves, so no management is needed
B. The evolution process is self-regulating, so measures such as size and release intervals follow statistically regular trends
C. The system automatically fixes its own defects over time
D. Each release must be smaller than the previous one

1053. Lehman's law of conservation of organisational stability states that: Understanding

A. The organisation's structure never changes while the system is in use
B. The amount of change in each release stays roughly constant
C. The system's quality stays stable without maintenance
D. The average effective work rate on an evolving system stays roughly constant over its lifetime, independent of the resources devoted to it

1054. Parnas, in his paper on software aging, names two main causes. What are they? Understanding

A. Hardware wear and magnetic decay of stored programs
B. Compiler upgrades and operating system patches
C. Failing to change the software as needs change, and changes made by people who do not understand the original design
D. Too many unit tests and too much documentation

1055. Sommerville describes a legacy system as a socio-technical system made of several layers. Which item is one of those layers? Understanding

A. Business processes and the policies and rules they follow
B. The source code of the compiler's test suite
C. The marketing team's social media accounts
D. The developer's personal IDE settings

1056. What is the main risk of a big-bang (cold-turkey) replacement of a legacy system? Understanding

A. It takes many years because each component is replaced one at a time
B. The old and new systems must run side by side, doubling the work
C. The legacy code must first be wrapped in web services
D. Everything switches over at once, so any missed business rule or defect can stop the business with no fallback

1057. In program comprehension, how does a top-down strategy differ from a bottom-up strategy? Understanding

A. Top-down reads the file from the first line; bottom-up reads it from the last line
B. Top-down is used for new code; bottom-up only for legacy code
C. Top-down uses tools; bottom-up uses only documentation
D. Top-down starts from a hypothesis about the program's purpose and refines it; bottom-up reads statements and groups them into higher-level chunks

1058. How does traceability between requirements, design, code and tests help maintenance? Understanding

A. It makes the code run faster
B. It shows which design elements, code and tests are linked to a requirement, so the impact of a change can be found quickly
C. It removes the need for regression testing
D. It automatically fixes defects in linked modules

1059. Why is a good automated test suite a prerequisite for safe refactoring? Understanding

A. Refactoring must keep behaviour unchanged, and the tests show quickly whether it did
B. Refactoring adds new features that need new tests
C. Tests make the refactored code run faster
D. Without tests, refactoring tools will not start

1060. Which of these is a sub-characteristic of maintainability in ISO/IEC 25010? Understanding

A. Fault tolerance
B. Analysability
C. Learnability
D. Time behaviour

1061. A payment provider retires version 1 of its API, and the shop's checkout must be changed to call version 2. The shop's own features stay the same. Under ISO/IEC 14764, this change is: Application

A. Perfective maintenance
B. Corrective maintenance
C. Adaptive maintenance
D. Preventive maintenance

1062. A team splits a 900-line class into smaller cohesive classes purely to make future changes easier; no user asked for it and nothing has failed. Using SWEBOK's convention, how is this classified? Application

A. Perfective maintenance, because it improves maintainability
B. Corrective maintenance, because the class was badly written
C. Adaptive maintenance, because the code structure changed
D. Emergency maintenance, because it was unscheduled

1063. A 250 KLOC system has 20 KLOC added and 10 KLOC modified in a year. What is its annual change traffic (ACT)? Application

A. 0.08
B. 0.12
C. 0.04
D. 0.30

1064. A 160 KLOC system took 800 person-months (PM) to develop. Each year 8 KLOC are added and 4.8 KLOC are modified. Using Boehm's ACT method, what is the estimated annual maintenance effort? Application

A. 40 PM
B. 24 PM
C. 128 PM
D. 64 PM

1065. A 300 KLOC system took 1,200 PM to develop. Each year 18 KLOC are added and 12 KLOC modified. Using the ACT method, about how many full-time maintainers does the system need? Application

A. 10
B. 120
C. 6
D. 4

1066. A system cost 1.5 million to develop and 0.25 million a year to maintain over a 12-year life. What share of its lifetime cost was maintenance? Application

A. About 66.7%
B. About 16.7%
C. 50%
D. About 83.3%

1067. Dependencies (X -> Y means X uses Y): Checkout -> Currency, Invoice -> Currency, Report -> Invoice, Dashboard -> Report, Login -> Session, Cart -> Checkout. The Currency module's rounding rule changes. Which modules could be affected and should be considered for retesting? Application

A. Checkout and Invoice only
B. Checkout, Invoice, Cart, Report and Dashboard
C. Checkout, Invoice and Report
D. Every module, including Login and Session

1068. Dependencies (X -> Y means X uses Y): A -> B, B -> C, C -> D, E -> C, F -> E, G -> H, H -> D. Module C is changed. Counting C itself, how many modules should be retested? Application

A. 3
B. 8
C. 5
D. 4

1069. As the first step of re-engineering, a team runs a tool that automatically converts COBOL source into equivalent Java, keeping the program's behaviour. In Sommerville's re-engineering process this activity is: Application

A. Data re-engineering
B. Program modularisation
C. Reverse engineering
D. Source code translation

1070. A method in class Order reads six getters from Customer and uses none of Order's own fields. Which smell and refactoring fit best? Application

A. Shotgun surgery; split the method across many classes
B. Feature envy; move the method to Customer
C. Long parameter list; introduce a parameter object
D. Duplicated code; extract a superclass

1071. Which refactoring best addresses the smell in this function? Application

```
def create_invoice(name, street, city, postcode, country, phone, email, amount):
    ...
```

A. Rename Method, giving the function a clearer name
B. Introduce Parameter Object, grouping the address fields into one Address object
C. Inline Method, copying the body into every caller
D. Replace Magic Number with Symbolic Constant

1072. The first version of total() was changed into the second version shown. Which refactoring was applied, and does the output stay the same? Application

```
# before
def total(price):
    return round(price * 1.17, 2)

# after
GST_RATE = 0.17

def total(price):
    return round(price * (1 + GST_RATE), 2)

print(total(100))
```

A. Extract Method; the second prints 17.0
B. Replace Magic Number with Symbolic Constant; both print 117.0
C. Introduce Parameter Object; both print 117.0
D. Inline Variable; the second prints 100.17

1073. Using Sommerville's value-quality assessment, what is the usual recommendation for a legacy system with high system quality but low business value? Application

A. Re-engineer it urgently
B. Continue normal maintenance and invest in new features
C. Replace it with an off-the-shelf system, scrap it, or keep maintaining it while maintenance stays cheap
D. Wrap it and expose it as a web service to all customers

1074. A legacy payroll system is well structured, well documented and critical to the business. Under Sommerville's assessment, what should happen to it? Application

A. Continue in operation with normal maintenance
B. Scrap it immediately
C. Re-engineer it from scratch
D. Replace it with an off-the-shelf package

1075. The classic maintainability index is MI = 171 - 5.2 ln(V) - 0.23 G - 16.2 ln(LOC), where V is Halstead volume and G is cyclomatic complexity. For a module with V = 1,000, G = 10 and LOC = 200, what is MI to the nearest whole number? Application

A. 133
B. 47
C. 85
D. 68

1076. A team spends 2,400 hours a year on maintenance, and about half of that goes on understanding existing code. A reverse-engineering tool would cut comprehension time by 30%. How many hours a year would it save? Application

    A. 720 hours
    B. 360 hours
    C. 1,200 hours
    D. 1,680 hours

1077. Keeping a legacy system costs 120k a year. Replacing it costs 500k now, after which upkeep is 30k a year. The business will retire the function in 5 years. Ignoring discounting, which option is cheaper over those 5 years? Application

    A. Replace it, by 50k
    B. Replace it, by 450k
    C. They cost exactly the same
    D. Keep the legacy system, by 50k

1078. A 150 KLOC system took 600 PM to develop, and its annual maintenance effort is 90 PM. Assuming Boehm's ACT model holds, about how many KLOC are added or modified each year? Application

    A. 15 KLOC
    B. 90 KLOC
    C. 0.15 KLOC
    D. 22.5 KLOC

1079. A year's 40 change requests were: 10 fixes of reported failures, 3 emergency overnight patches, 5 fixes of latent faults found in code review, 12 changes for a new OS version and 10 user-requested improvements. What share fall into ISO/IEC 14764's correction class? Application

    A. 37.5%
    B. 32.5%
    C. 45%
    D. 55%

1080. A team builds a facade in front of an old monolith and moves features one at a time to new services, routing each feature's traffic to the new code, until the monolith can be switched off. Which approach is this? Application

    A. The strangler fig pattern of incremental replacement
    B. Big-bang replacement
    C. Reverse engineering
    D. Redocumentation

1081. A maintainer is cleaning up the function below. Which part is dead code that can be removed without changing the function's behaviour for any integer n? Application

```
def classify(n):
    if n > 10:
        return "big"
    elif n > 20:
        return "huge"
    return "small"
```

    A. The final return "small"
    B. The elif branch that returns "huge"
    C. The if branch that returns "big"
    D. No part can be removed

1082. A change log reads: 'Generated the design of the billing module from its code, then rebuilt the module in C# with a new layered structure but the same functions.' Which term describes the whole activity most accurately? Analysis

    A. Reverse engineering
    B. Restructuring
    C. Re-engineering
    D. Redocumentation

1083. A developer 'refactors' the original function into the shorter version below and says behaviour is unchanged. Is this a valid refactoring? Analysis

```
# original
def average(xs):
    if not xs:
        return 0
    total = 0
    for x in xs:
        total += x
    return total / len(xs)

# refactored
def average(xs):
    return sum(xs) / len(xs)
```

    A. Yes: both return 5.0 for [2, 4, 9], so behaviour is identical
    B. No: the new version returns an integer instead of a float
    C. No: average([]) used to return 0 but now raises ZeroDivisionError
    D. Yes: sum() is always faster, and speed is part of refactoring

1084. To 'stabilise' an E-type retail system, a manager freezes all changes for three years while competitors, tax rules and payment methods keep changing. What do Lehman's laws predict? Analysis

    A. Users will find it progressively less satisfactory, because continuing change and declining quality apply even when nobody touches the code
    B. Quality will stay constant, because no new defects can be introduced
    C. Complexity will fall, because no changes are being made
    D. Nothing, because Lehman's laws apply only to S-type programs

1085. Over four releases, the maintenance team grew from 5 to 15 people, yet the number of features delivered per release stayed almost flat. Which Lehman law does this pattern illustrate? Analysis

    A. Continuing growth
    B. Conservation of familiarity
    C. Continuing change
    D. Conservation of organisational stability

1086. A system will be retired in 5 years. Option A, partial restructuring, costs 60 PM and saves 25 PM a year. Option B, full re-engineering, costs 240 PM and saves 70 PM a year. Ignoring discounting, which gives the larger net saving? Analysis

    A. B: net 110 PM against A's 65 PM
    B. A: it pays back in 2.4 years, faster than B
    C. A: net 125 PM against B's 110 PM
    D. Neither: both lose money over 5 years

1087. Dependencies (X -> Y means X uses Y): Report -> Stats, Stats -> Parser, Export -> Parser, Alerts -> Report. After changing Parser, the team retested only Stats and Export, and a fault then appeared in Alerts. What went wrong? Analysis

A. Alerts must have had a separate, unrelated fault, since it does not use Parser
B. Stats and Export should not have been retested at all
C. The fault proves that Parser should never have been changed
D. The impact analysis stopped at direct dependants; Alerts depends on Parser indirectly through Report and Stats

1088. An internal security test finds an injection flaw in a login form. No attack or failure has occurred in production, and the team fixes it before release. Under ISO/IEC 14764, which classification is best? Analysis

A. Corrective maintenance: any security fix is corrective
B. Preventive maintenance: a latent fault is corrected before it causes a failure
C. Adaptive maintenance: the threat environment has changed
D. Perfective maintenance: security is a quality improvement

1089. A legacy order system has clean, well-documented code and still matches the business well, but it runs on a hardware platform whose vendor has stopped all support. What is the most sensible strategy? Analysis

A. Migrate it to a supported platform, adapting the code as needed, rather than rewriting it
B. Scrap it, since unsupported hardware makes it worthless
C. Rewrite all of its business logic from scratch in a new language
D. Change nothing, since the code quality is high

1090. A team has capacity for 40 story points a sprint, but technical debt drags 15% of it away. A one-off clean-up costing 30 points would remove two-thirds of that drag. After how many sprints does the clean-up pay for itself? Analysis

A. 5 sprints
B. 0.75 sprints
C. 7.5 sprints, so it has paid back by the end of the 8th sprint
D. It never pays back, because some drag remains

1091. Maintenance effort last year: corrective 20%, adaptive 25%, perfective 50%, preventive 5%. A manager says, 'Better testing before release is the best way to cut maintenance costs, because maintenance is mostly bug fixing.' What does the data show? Analysis

A. The claim is weak: 75% of effort is enhancement, so better requirements, design and maintainability matter more than catching bugs
B. The claim is right: corrective work is the largest category
C. The claim is right: corrective and preventive together are the majority
D. No conclusion is possible without the number of change requests

1092. A vendor's proposal says: 'We will reverse engineer your system into Java, so you get a modern codebase.' What is wrong with this wording? Analysis

A. Nothing: reverse engineering always produces code in a new language
B. Reverse engineering applies only to hardware, not software
C. Reverse engineering is illegal for software the client owns
D. Reverse engineering only analyses and recovers designs; producing a new Java codebase requires forward engineering, so the whole job is re-engineering

1093. The two functions began as copies of each other. A bug fix capping the discount at 50 was applied to only one. What does the program print, and what maintenance problem does it show? Analysis

```
def member_price(p):
    d = min(p * 0.10, 50)   # fixed: discount capped at 50
    return p - d

def staff_price(p):
    d = p * 0.10            # copied before the fix
    return p - d

print(member_price(1000), staff_price(1000))
```

A. 950 950: both functions share the same fix
B. 950 900.0: duplicated code let the fix reach only one copy
C. 900.0 900.0: the min() call has no effect
D. 1000 900.0: the cap removes the whole discount

1094. Two 100 KLOC systems have the same development effort and the same ACT. System P has current design documents and its original team; System Q has neither. The plain ACT model gives them equal maintenance estimates. Which conclusion is sound? Analysis

A. Q's real effort will probably be higher, because the basic ACT estimate ignores understandability and staff factors that Boehm adjusts with cost drivers
B. Their real effort will be equal, because ACT captures everything that matters
C. P's real effort will be higher, because documents must also be maintained
D. ACT cannot be used for systems over 50 KLOC

1095. Four legacy systems are rated on business value and quality: W (high, low), X (low, low), Y (high, high), Z (low, high). Which mapping follows Sommerville's assessment? Analysis

A. W: scrap; X: re-engineer; Y: normal maintenance; Z: re-engineer
B. W: normal maintenance; X: scrap; Y: re-engineer; Z: replace with COTS
C. W: re-engineer or replace; X: scrap; Y: normal maintenance; Z: replace with COTS, scrap or keep cheaply
D. W: re-engineer; X: normal maintenance; Y: scrap; Z: re-engineer

1096. A product has been in Rajlich and Bennett's servicing stage for years: the original architects have left and only patches are made. The business now asks for a major new capability that cuts across the architecture. What is the most realistic assessment? Analysis

A. It is easy, because a system in servicing is stable
B. It is impossible, because systems in servicing cannot be changed at all
C. It is purely an adaptive change, so its cost is fixed
D. It will be costly and risky, because servicing systems have degraded architecture and lost expertise; re-engineering or replacement may be needed first

Software Quality Assurance MCQs for NSCT

1097. Which activity is quality assurance (QA) rather than quality control (QC)? Understanding

A. Running test cases on a build and logging defects
B. Inspecting a finished module for defects
C. Defining coding standards and auditing whether projects follow them
D. Measuring the response time of a release candidate

1098. Boehm summarised verification and validation as two questions. Which pairing is correct? Understanding

A. Verification: are we building the product right? Validation: are we building the right product?
B. Verification: are we building the right product? Validation: are we building the product right?
C. Verification: is the code fast? Validation: is the code secure?
D. Verification: did users accept it? Validation: did it compile?

1099. What is ISO 9001? Understanding

A. A standard only for software code quality metrics
B. A programming-language coding standard
C. An international standard specifying requirements for a quality management system, usable by any organisation
D. A model with five maturity levels for software processes

1100. In the CMMI staged representation (v1.3), what is maturity level 2 called? Understanding

A. Repeatable
B. Managed
C. Defined
D. Initial

1101. In a Fagan inspection, who plans the inspection and leads the meeting? Understanding

A. The moderator
B. The author
C. The reader
D. The recorder

1102. Which of these is one of the seven quality management principles behind ISO 9001:2015? Understanding

A. Evidence-based decision making
B. Zero defects in every release
C. Maximum lines of code per developer
D. Mandatory waterfall development

1103. ISO 9001's process approach is built around which improvement cycle? Understanding

A. Plan-Do-Check-Act (PDCA)
B. Red-Green-Refactor
C. Build-Measure-Learn
D. Analyse-Design-Code-Test

1104. At CMMI maturity level 3 (Defined), what characterises an organisation? Understanding

A. Processes are ad hoc and success depends on individual heroes
B. Projects tailor their processes from a set of organisation-wide standard processes
C. Processes are controlled with statistical and quantitative techniques
D. Each project plans its own process, with no organisational standard

1105. Which CMMI maturity level is marked by managing processes with statistical and other quantitative techniques? Understanding

A. Level 2, Managed
B. Level 3, Defined
C. Level 5, Optimizing
D. Level 4, Quantitatively Managed

1106. How does a walkthrough usually differ from an inspection? Understanding

A. A walkthrough executes the code; an inspection does not
B. A walkthrough always involves customers; an inspection never involves the author
C. A walkthrough is led by the author and is less formal; an inspection is led by a trained moderator and follows a defined process with checklists and metrics
D. There is no difference; the two terms are synonyms

1107. In the cost of quality model, the cost of holding code inspections and running tests falls under which category? Understanding

A. Prevention cost
B. Internal failure cost
C. Appraisal cost
D. External failure cost

1108. Which of these is a product quality characteristic in ISO/IEC 25010? Understanding

A. Profitability
B. Team velocity
C. Maintainability
D. Lines of code

1109. McCall's quality model groups factors under three perspectives. Under which perspective does maintainability fall? Understanding

A. Product revision
B. Product operation
C. Product transition
D. Product delivery

1110. How does CMMI's continuous representation differ from its staged representation? Understanding

A. Continuous has five maturity levels; staged has none
B. Continuous is used only for hardware; staged only for software
C. Continuous ignores process areas; staged ignores goals
D. Continuous rates capability levels for individual process areas; staged rates the whole organisation with maturity levels

1111. Who issues an ISO 9001 certificate to a software company? Understanding

A. ISO itself, through its head office in Geneva
B. The company's own QA manager
C. An accredited third-party certification body, after an audit
D. The CMMI Institute

1112. Which IEEE standard is dedicated to software quality assurance processes, including the SQA plan? Understanding

A. IEEE 829
B. IEEE 1028
C. IEEE 730
D. IEEE 830

1113. A module of 15 KLOC has 45 confirmed defects. What is its defect density? Application

A. 3 defects per KLOC
B. 0.33 defects per KLOC
C. 45 defects per KLOC
D. 30 defects per KLOC

1114. Before release, a team finds 90 defects. Users report 10 more after release. What is the defect removal efficiency (DRE)? Application

A. 0.10
B. 0.90
C. 0.89
D. 9.0

1115. A service has a mean time to failure (MTTF) of 98 hours and a mean time to repair (MTTR) of 2 hours. Using Pressman's formula, what is its availability? Application

A. 96%
B. 2%
C. 98%
D. 100%

1116. A server's MTTF is 450 hours and its MTTR is 50 hours. Using Pressman's definitions, what is its mean time between failures (MTBF)? Application

A. 450 hours
B. 400 hours
C. 9 hours
D. 500 hours

1117. A system averages 480 hours of uptime between failures, and each repair takes 20 hours on average. What is its availability? Application

A. 95.8%
B. 98%
C. 96%
D. 4%

1118. A service promises 99.9% availability. About how much downtime per year (8,760 hours) does that allow? Application

A. 0.876 hours
B. 8.76 hours
C. 87.6 hours
D. 1 hour

1119. A component has a constant failure rate with MTTF = 1,000 hours. What is the probability that it runs 100 hours without failure? (R(t) = e^(-t/MTTF)) Application

A. About 0.905
B. 0.900
C. 0.100
D. About 0.990

1120. An inspection of 2,000 delivered units, each with 5 defect opportunities, finds 18 defects. What is the defects per million opportunities (DPMO) figure? Application

A. 9,000
B. 3,600
C. 360
D. 1,800

1121. Annual quality costs: prevention 20k, appraisal 30k, internal failure 25k, external failure 45k. What percentage of the total cost of quality is failure cost? Application

A. 37.5%
B. 41.7%
C. 45%
D. 58.3%

1122. Three modules: A has 12 defects in 4 KLOC, B has 15 defects in 6 KLOC and C has 9 defects in 2 KLOC. Which has the highest defect density? Application

A. C (4.5 per KLOC)
B. B (most defects)
C. A (3.0 per KLOC)
D. B (2.5 per KLOC)

1123. Four inspectors each prepare for an inspection of a 1,200-line module at 200 lines per hour. How much preparation effort is that in total? Application

A. 24 person-hours
B. 6 person-hours
C. 1.5 person-hours
D. 4.8 person-hours

1124. An organisation sets statistical control limits for review defect density and investigates any project that falls outside them. Which CMMI maturity level does this practice most indicate? Application

A. Level 2, Managed
B. Level 4, Quantitatively Managed
C. Level 3, Defined
D. Level 1, Initial

1125. Every project in a firm has a plan, tracks requirements changes and controls configurations, but each project defines its own process and there is no company-wide standard. Which CMMI maturity level fits best? Application

A. Level 1, Initial
B. Level 3, Defined
C. Level 4, Quantitatively Managed
D. Level 2, Managed

1126. Defects found per stage: requirements review 20, design review 30, code review 40, testing 50, and after release 10. What is the defect removal efficiency of the testing stage, measured as defects it found against those still present when it began? Application

A. 93.3%
B. 33.3%
C. 50%
D. 83.3%

1127. Team X reports 30 defects in 10 KLOC; Team Y reports 10 defects in 2 KLOC. A manager says Team Y's code is better because it has fewer defects. Which conclusion is best supported? Analysis

A. Team Y is better because 10 defects is fewer than 30
B. Team Y's defect density (5 per KLOC) is higher than Team X's (3 per KLOC), so the claim does not hold on this data
C. Both teams are equal because each found defects
D. Team X is worse because it wrote more code

1128. Which activity in a release cycle is quality control rather than quality assurance? Analysis

A. Writing the project's quality plan
B. Training developers in the coding standard
C. Auditing whether teams follow the review process
D. Executing test cases on build 4.2 and logging the defects found

1129. A vendor says: 'We are appraised at CMMI maturity level 5, so your project will be defect-free.' What is the most accurate response? Analysis

A. The claim is correct, because level 5 means zero defects
B. Level 5 indicates mature, continuously improving processes, not a promise that any product is defect-free
C. The claim is meaningless, because CMMI says nothing about processes
D. Level 5 is the lowest level, so the vendor is weak

1130. System A: MTTF 1,000 h, MTTR 10 h. System B: MTTF 500 h, MTTR 2 h. Which has the higher availability? Analysis

A. A, because it fails half as often
B. They are equal, because both are above 99%
C. A, about 99.60% against B's 99.01%
D. B, about 99.60% against A's 99.01%

1131. A team building a safety-critical insulin-dose module wants the review type with the highest defect detection, defined roles, checklists and recorded metrics. Which should it choose? Analysis

A. A formal (Fagan-style) inspection
B. An informal pair walkthrough led by the author
C. An ad hoc desk check by the author
D. A management review of the schedule

1132. Before an SQA programme, yearly costs were prevention 10, appraisal 40, internal failure 60 and external failure 90 (thousands). After it, they were 30, 40, 40 and 40. What happened? Analysis

A. Cost of quality rose, because prevention tripled
B. Total cost of quality fell by 25%, because the extra prevention cut failure costs
C. Nothing changed, because appraisal stayed at 40
D. Total cost fell by 50%, because external failure fell from 90 to 40

1133. A small software firm sells to banks, hospitals and manufacturers, and wants a quality credential that all of them recognise. Which statement about ISO 9001 and CMMI is accurate? Analysis

A. Both are issued by ISO and are the same thing
B. ISO 9001 gives third-party certification of a generic QMS; CMMI gives an appraised maturity rating specific to development processes
C. CMMI certification is issued by ISO, and ISO 9001 applies only to software
D. ISO 9001 rates organisations from level 1 to 5

1134. Release 1: 180 defects found in-house, 20 by users. Release 2: 160 found in-house, 40 by users. A manager says quality improved because the team found fewer defects in-house. What does DRE show? Analysis

A. DRE rose from 0.80 to 0.90, so quality improved
B. DRE fell from 0.90 to 0.80, so more defects escaped to users and removal effectiveness got worse
C. DRE stayed at 0.90 in both releases
D. DRE cannot be computed without lines of code

1135. A pharmacy app passes every specification-based test, but in beta, pharmacists say its workflow does not match how they actually dispense medicine. What kind of problem is this? Analysis

A. A verification problem: the code does not match its specification
B. A configuration management problem
C. A validation problem: the product was built right but is not the right product
D. A performance problem

1136. Process A has 50 defects in 10,000 opportunities. Process B has 12 defects in 4,000 opportunities. Which is better on a DPMO basis? Analysis

A. A, because 10,000 opportunities is a larger sample
B. A, with 3,000 DPMO against B's 5,000
C. They are equal, because both have under 1% defects
D. B, with 3,000 DPMO against A's 5,000

1137. Philip Crosby defined quality as: Understanding

A. Fitness for use
B. Delight beyond expectations
C. The absence of any documentation errors
D. Conformance to requirements

1138. Joseph Juran's well-known definition of quality is: Understanding

A. Fitness for use
B. Conformance to requirements
C. Zero defects
D. Meeting the schedule and budget

1139. Garvin described five views of quality. Which view defines quality as conformance to the specification and doing things right first time? Understanding

A. The transcendental view
B. The user view
C. The manufacturing view
D. The value-based view

1140. In ISO/IEC 25010, fault tolerance is a sub-characteristic of which product quality characteristic? Understanding

A. Security
B. Reliability
C. Maintainability
D. Performance efficiency

1141. In ISO/IEC 25010, learnability is a sub-characteristic of: Understanding

A. Portability
B. Compatibility
C. Functional suitability
D. Usability

1142. ISO/IEC 25010 has a product quality model and a separate quality in use model. Which characteristic belongs to the quality in use model? Understanding

A. Portability
B. Compatibility
C. Satisfaction
D. Maintainability

1143. In McCall's quality model, which factor falls under product transition? Understanding

A. Correctness
B. Flexibility
C. Reusability
D. Usability

1144. In McCall's quality model, which factor belongs to product operation? Understanding

A. Testability
B. Portability
C. Integrity
D. Flexibility

1145. In the cost of quality model, which item is a prevention cost? Understanding

A. Training developers in secure coding practices
B. Running the regression test suite
C. Fixing a defect found in system testing
D. Issuing a patch after customers report a crash

1146. Which cost is an external failure cost? Understanding

A. Paying for code inspections
B. Writing coding standards
C. Reworking a module after a failed unit test
D. Handling customer complaints and support calls about a released defect

1147. Which CMMI maturity level focuses on continuous process improvement through causal analysis and innovation? Understanding

A. Level 5, Optimizing
B. Level 4, Quantitatively Managed
C. Level 3, Defined
D. Level 2, Managed

1148. What characterises an organisation at CMMI maturity level 1, Initial? Understanding

A. Every project tailors an organisation-wide standard process
B. Processes are controlled with statistical techniques
C. Projects are planned and tracked, but processes vary between projects
D. Processes are ad hoc and success depends on individual effort and heroics

1149. In CMMI v1.3, which SCAMPI appraisal class is the only one that can produce an official maturity-level rating? Understanding

A. Class B
B. Class A
C. Class C
D. Class D

1150. In a Fagan inspection, which stage comes immediately after the inspection meeting? Understanding

A. Rework
B. Overview
C. Individual preparation
D. Planning

1151. In a Fagan inspection meeting, what does the reader do? Understanding

A. Records each defect in the inspection log
B. Paraphrases the work product piece by piece, so the inspectors examine it together
C. Plans the inspection and chooses the team
D. Fixes the defects during the meeting

1152. In a formal inspection, which role writes down each defect found, with its location and type? Understanding

A. The recorder (scribe)
B. The author
C. The reader
D. The customer

1153. Which guideline is part of Pressman's advice for formal technical reviews? Understanding

A. Review the product, not the producer
B. Solve every problem fully during the meeting
C. Invite as many people as possible
D. Skip preparation so that views stay fresh

1154. At the end of a formal technical review, which set of decisions does Pressman say the reviewers may reach? Understanding

A. Pass or fail only
B. Release, delay or cancel the project
C. Rate the work product from level 1 to level 5
D. Accept without modification, reject because of severe errors, or accept provisionally once minor errors are corrected

1155. Six Sigma's goal is a process with no more than about how many defects per million opportunities? Understanding

A. 34
B. 3.4
C. 233
D. 6,210

1156. What is the correct order of the Six Sigma DMAIC cycle for improving an existing process? Understanding

A. Design, Model, Implement, Check, Close
B. Define, Manage, Inspect, Approve, Certify
C. Define, Measure, Analyse, Improve, Control
D. Detect, Measure, Isolate, Improve, Correct

1157. Which standard gives guidance on applying ISO 9001 to computer software? Understanding

A. ISO/IEC 25010
B. IEEE 1028
C. ISO/IEC 14764
D. ISO/IEC/IEEE 90003

1158. What is ISO/IEC 15504, known as SPICE? Understanding

A. A product quality model with eight characteristics
B. A standard for software reviews and audits
C. A process assessment framework that rates process capability levels, since succeeded by the ISO/IEC 330xx series
D. A certification of a quality management system

1159. What is the main purpose of an SQA audit? Understanding

A. To run the system test suite before release
B. To fix the defects found in code reviews
C. To estimate the project's cost and schedule
D. To check independently whether the project's processes and work products conform to agreed plans, standards and procedures

1160. In statistical SQA, the Pareto principle suggests that: Understanding

A. Every defect cause deserves equal attention
B. A small number of causes (the vital few) produce most of the defects, so effort should go on those first
C. 80% of testing should be done after release
D. Defects are spread evenly across all modules

1161. A 24 KLOC component has 36 confirmed defects. What is its defect density? Application

A. 1.5 defects per KLOC
B. 0.67 defects per KLOC
C. 36 defects per KLOC
D. 15 defects per KLOC

1162. A service promises 99.99% availability. About how much downtime does that allow per year of 8,760 hours? Application

A. About 8.76 hours
B. About 5.3 minutes
C. About 52.6 minutes
D. About 87.6 hours

1163. During system testing, a team spends three days fixing a defect and re-running tests before release. In the cost of quality model, this effort is: Application

A. An external failure cost
B. An internal failure cost
C. A prevention cost
D. An appraisal cost

1164. Design reviews found 24 design errors. Later activities found 6 more errors that had been introduced during design. Using Pressman's phase DRE, what is the DRE of the design activity? Application

A. 0.80
B. 0.25
C. 0.75
D. 4.0

1165. A request passes through three components in series with reliabilities 0.95, 0.90 and 0.98. What is the system's reliability? Application

A. About 0.943
B. 0.90
C. About 0.999
D. About 0.838

1166. Two identical servers, each with reliability 0.90 over a period, run in parallel, and the service works if at least one works. What is the service's reliability? Application

A. 0.81
B. 0.99
C. 0.90
D. 0.95

1167. A component has a constant failure rate and MTTF = 400 hours. Using R(t) = e^(-t/MTTF), what is the probability it runs 200 hours without failure? Application

A. 0.50
B. About 0.39
C. About 0.61
D. About 0.82

1168. A protection system is called on 2,000 times in testing and fails to respond correctly 4 times. What is its probability of failure on demand (POFOD)? Application

A. 0.02
B. 0.004
C. 500
D. 0.002

1169. A system logged 6 failures in 1,500 hours of operation. What is its rate of occurrence of failure (ROCOF)? Application

A. 250 failures per 1,000 hours
B. 4 failures per 1,000 hours
C. 6 failures per 1,000 hours
D. 0.4 failures per 1,000 hours

1170. Two inspectors review the same document independently. A finds 24 defects, B finds 18, and 12 are found by both. Using the capture-recapture estimate N = (A x B) / common, about how many defects remain undetected? Application

A. About 6
B. About 36
C. About 30
D. About 18

1171. Relative cost to fix a requirements defect: 1 unit if caught in a requirements review, 100 units if caught after release. Ten requirements defects slip through to release. How many extra cost units does that cause compared with catching them in review? Application

A. 1,000
B. 99
C. 10
D. 990

1172. A 1,800-line module will be inspected at 150 lines per hour, and the team limits each inspection meeting to 2 hours. How many meetings are needed? Application

A. 6
B. 12
C. 3
D. 9

1173. Four inspectors each spend 3 hours preparing and 2 hours in the inspection meeting, and they find 25 defects. What is the inspection effort per defect? Application

A. 0.2 person-hours
B. 1.25 person-hours
C. 5 person-hours
D. 0.8 person-hours

1174. A process produced 58 defects in 250,000 opportunities. Using the usual table with a 1.5-sigma shift (6,210 DPMO = 4 sigma, 233 = 5 sigma, 3.4 = 6 sigma), about what sigma level is it at? Application

A. About 5 sigma (232 DPMO)
B. About 4 sigma (5,800 DPMO)
C. About 6 sigma (3.4 DPMO)
D. About 3 sigma (23,200 DPMO)

1175. Annual quality costs: prevention 15k, appraisal 35k, internal failure 30k, external failure 70k. What share of the total cost of quality is the cost of conformance (prevention plus appraisal)? Application

A. About 23.3%
B. About 33.3%
C. 50%
D. About 66.7%

1176. A component has a constant failure rate with MTTF = 250 hours. How many failures are expected in 1,000 hours of operation? Application

A. 0.25
B. 250
C. 4
D. 40

1177. An organisation keeps a library of standard processes, tailoring guidelines and an organisation-wide training programme. Projects tailor the standard processes, but no process is controlled with statistical limits yet. Which CMMI maturity level fits best? Application

A. Level 2, Managed
B. Level 4, Quantitatively Managed
C. Level 3, Defined
D. Level 5, Optimizing

1178. A team found 200 defects before release in a 45 KLOC product, and its historical DRE is 0.95. If that DRE holds, about how many defects should users be expected to find? Application

A. About 10
B. About 2
C. About 11
D. About 190

1179. A service has MTTF = 995 hours. Using Pressman's availability = MTTF / (MTTF + MTTR), what is the longest mean time to repair that still meets a 99.5% availability target? Application

A. 4.975 hours
B. 5 hours
C. 0.5 hours
D. 50 hours

1180. A defect-cause table shows: incomplete specification 45, misunderstood customer needs 30, interface errors 8, data representation errors 6, standards violations 5, design logic errors 4, other 2. What is the smallest set of causes that explains at least 75% of defects? Application

A. Incomplete specification alone
B. The top three causes
C. Every cause above 5 defects
D. Incomplete specification and misunderstood customer needs

1181. An SQA engineer finds that a team skipped the code reviews required by the project's SQA plan for two sprints. Following Pressman's description of SQA group tasks, what should happen? Application

A. Quietly run the reviews themselves and say nothing
B. Record the noncompliance, report it to the responsible manager, and track it until it is resolved
C. Immediately cancel the release without telling the team
D. Change the SQA plan so that reviews become optional

1182. After a year with very few customer-reported failures, a manager proposes cutting the training and standards budget, since 'quality is clearly fine'. What does the cost of quality model suggest? Analysis

A. Prevention spending is wasted whenever failures are low
B. The low failure costs may be the result of prevention spending, so cutting it is likely to raise failure costs later
C. Only appraisal spending affects failure costs
D. Cost of quality says nothing about the link between prevention and failure

1183. Team A's inspection found 40 defects in a module later estimated to contain 50. Team B's found 30 in a module estimated to contain 32. Which inspection was more effective, and why? Analysis

A. A: it found 10 more defects
B. A: its module had more defects, so its review was harder
C. B: its yield was about 94% against A's 80%
D. They are equal, because both found most defects

1184. System X: MTTF 100 h, MTTR 0.1 h. System Y: MTTF 2,000 h, MTTR 10 h. A procedure where any failure during a 1-hour session is dangerous needs one of them. Which should be chosen? Analysis

A. Y, because its 1-hour reliability (about 0.9995) beats X's (about 0.990), even though X has higher availability
B. X, because its availability (about 99.9%) is higher than Y's (about 99.5%)
C. X, because it repairs 100 times faster
D. Either, because their availabilities are almost the same

1185. Design A puts two components, each with reliability 0.99, in series. Design B uses one 0.97-reliable component duplicated in parallel. Which design is more reliable? Analysis

A. A: its components are individually more reliable
B. A: about 0.9999 against B's 0.97
C. They are equal, at about 0.98
D. B: about 0.9991 against A's 0.9801

1186. A CMMI level 1 company wants to jump straight to statistical process control (level 4) by buying a metrics dashboard. Why is this unlikely to work? Analysis

A. Level 4 does not involve any measurement
B. CMMI forbids buying tools
C. Statistics can be used only at level 5
D. Statistical control needs stable, defined processes first; measuring ad hoc processes gives data too variable to control

1187. A company rewards teams for low defect density. Soon, the teams stop logging defects found in their own code reviews. What is the main problem? Analysis

A. The metric is being gamed: reviews still find defects, but hiding them corrupts the data and removes the feedback that improves the process
B. Nothing: fewer logged defects proves quality has improved
C. Defect density should be measured per developer instead
D. Reviews should be dropped, since they produce defect data

1188. The same feature was built in Python (10 KLOC) and in C (30 KLOC), each with 30 defects and each measured at 200 function points. A manager says the C version has better quality because its defect density is lower. What is the best analysis? Analysis

A. The manager is right: 1 defect per KLOC is better than 3
B. Python is better, since it has fewer lines
C. Per KLOC, C looks better (1 against 3), but per function point both are 0.15, so the LOC measure just penalises the more concise language
D. No comparison is possible, because the languages differ

1189. Two inspectors independently review a design. A finds 10 defects, B finds 12, and only 2 are common to both. Using capture-recapture, what should the moderator conclude? Analysis

A. Only 2 defects remain, because only 2 were common
B. The estimated total is 60 and only 20 have been found, so many defects probably remain and a re-inspection is justified
C. The estimated total is 22, so the inspection is complete
D. Low overlap shows the inspectors made mistakes, so their results should be discarded

1190. Review 1 covered 3,000 lines in one hour and found 2 defects. Review 2 covered 600 lines at 300 lines an hour and found 12. Both modules are of similar complexity. What is the most likely explanation? Analysis

A. Review 1 went far too fast to examine the code properly; its 0.7 defects per KLOC against 20 per KLOC reflects review speed, not better code
B. Review 1's module is much better, because fewer defects were found
C. Review 2's team invented defects to look busy
D. Review speed has no effect on defects found

1191. Errors found per activity, with the phase that introduced them: requirements errors 20 found in requirements review and 30 found later; design errors 34 found in design review and 6 later; code errors 40 found in code review and 10 later. Which review most needs improvement? Analysis

A. The code review: it let 10 errors through
B. The design review: it found the fewest errors
C. The requirements review: its DRE is only 0.40, against 0.85 for design and 0.80 for code
D. None: all three found more than 20 errors

1192. Quality costs (thousands) before an SQA programme: prevention 20, appraisal 50, internal failure 80, external failure 150. After: prevention 50, appraisal 60, internal 50, external 60. Which statement is correct? Analysis

A. Total cost rose, because prevention and appraisal spending increased
B. Total cost fell from 300 to 220, and failure costs fell from about 77% to 50% of the total
C. Total cost fell from 300 to 220, but the failure share rose
D. Failure costs fell by 50%, so total cost fell by 50%

1193. Product A: 1,000 units, 10 defect opportunities each, 50 defects. Product B: 400 units, 50 opportunities each, 60 defects. Which statement is correct? Analysis

A. A is better on every measure
B. B is better on every measure
C. They are equal, because both have about 5% defective units
D. B has the lower DPMO (3,000 against 5,000), even though it has more defects per unit (0.15 against 0.05)

1194. A project's V&V log labels four activities. Which label is wrong? Analysis

A. Inspecting the design against the requirements specification: verification
B. Running unit tests against the module specification: verification
C. Acceptance testing with real clerks to see if the system suits their daily work: verification
D. A beta trial where customers judge whether the product meets their needs: validation

1195. A firm's control chart for review defect density has a mean of 20 defects per KLOC, an upper control limit of 30 and a lower limit of 10. A project's review reports 8. What is the best response? Analysis

A. Investigate: below the lower limit can mean an ineffective review as easily as unusually clean code
B. Celebrate: below the lower limit proves the code is excellent
C. Ignore it: only points above the upper limit need investigation
D. Remove the project's data so the chart stays clean

1196. A developer wrote a 40-line internal script to rename log files; it is low risk and needs to ship today. The team lead asks for a full Fagan inspection with five roles. What is the best advice? Analysis

A. A full inspection is always mandatory, whatever the risk
B. Use a lighter review, such as a desk check or a quick walkthrough; match review formality to risk and cost
C. Skip all review, since scripts cannot have defects
D. Delay the script until a CMMI appraisal is done

Software Metrics & Measurement MCQs for NSCT

1197. Which of these is a direct measure of a software product? Understanding

A. Usability
B. Maintainability
C. Reliability
D. Lines of code (LOC)

1198. In IFPUG function point analysis, what is the possible range of the value adjustment factor (VAF)? Understanding

A. 1 to 70
B. 0 to 1
C. 0.5 to 1.5
D. 0.65 to 1.35

1199. Which of these is NOT one of the five function types counted in function point analysis? Understanding

A. External inquiries
B. Number of classes
C. External inputs
D. Internal logical files

1200. What does McCabe's cyclomatic complexity V(G) of a function tell you? Understanding

A. The depth of the function in the call tree
B. The number of linearly independent paths through its control flow graph
C. The number of lines in the function
D. The number of variables the function uses

1201. McCabe's original guideline suggests that a module should be reviewed or split when its cyclomatic complexity exceeds which value? Understanding

A. 2
B. 10
C. 100
D. 50

1202. Defects are classified as Low, Medium, High or Critical. On which measurement scale does this classification sit? Understanding

A. Interval
B. Ratio
C. Ordinal
D. Nominal

1203. In the Goal-Question-Metric (GQM) approach, what is defined first? Understanding

A. The metric to collect
B. The goal the organisation wants to achieve
C. The statistical test to apply
D. The tool that will collect data

1204. Defect density is usually expressed as: Understanding

A. Defects per developer
B. Defects per thousand lines of code (KLOC) or per function point
C. Defects per test case
D. Defects per month

1205. In the Chidamber and Kemerer (CK) metrics suite, what does DIT measure? Understanding

A. The number of methods in a class
B. The number of direct subclasses of a class
C. The number of classes a class is coupled to
D. The depth of a class in the inheritance tree

1206. In function point analysis, how does an internal logical file (ILF) differ from an external interface file (EIF)? Understanding

A. An ILF is maintained (updated) inside the application boundary; an EIF is only referenced and is maintained by another application
B. An ILF is stored on disk; an EIF is held in memory
C. An ILF holds user data; an EIF holds configuration data
D. An ILF is counted only for batch systems; an EIF only for online systems

1207. A screen displays a customer's record exactly as stored, with no calculation. Under IFPUG rules, how is this elementary process classified? Understanding

A. External inquiry (EQ)
B. External output (EO)
C. Internal logical file (ILF)
D. External input (EI)

1208. Why are function points often preferred over LOC for early estimation? Understanding

A. They measure code quality as well as size
B. They are always smaller numbers and so easier to work with
C. They can be counted from the requirements before any code exists, and they do not depend on the programming language
D. They are counted automatically by the compiler

1209. Which of these is a product metric rather than a process or project metric? Understanding

A. Effort spent by the team in each sprint
B. Number of change requests approved per month
C. Average time taken to fix a reported defect
D. Cyclomatic complexity of each module

1210. In Halstead's software science, what is the program vocabulary n? Understanding

A. The number of distinct operators only
B. The total count of operators plus operands
C. The number of lines multiplied by the number of operators
D. The number of distinct operators plus the number of distinct operands

1211. A class has a high LCOM (lack of cohesion in methods) value. What does this usually indicate? Understanding

A. Its methods share few instance variables, so the class may be doing several unrelated jobs
B. The class has too many subclasses
C. The class is highly cohesive and well designed
D. The class is tightly coupled to many other classes

1212. Sommerville lists POFOD (probability of failure on demand) as a reliability metric. For which kind of system is it most suitable? Understanding

A. A batch report that runs once a year
B. A protection system that is called on rarely, such as an emergency shutdown system
C. A system whose main concern is how long repairs take
D. A web server handling thousands of requests per second

1213. Testing a 12,000-line module finds 36 defects. What is its defect density? Application

A. 0.3 defects per KLOC
B. 3 defects per KLOC
C. 36 defects per KLOC
D. 33.3 defects per KLOC

1214. A team of 8 developers delivers 24,000 LOC in 6 months. What is their productivity in LOC per person-month? Application

A. 1,000
B. 4,000
C. 3,000
D. 500

1215. A connected control flow graph has 11 edges and 9 nodes. What is its cyclomatic complexity? Application

A. 4
B. 2
C. 3
D. 20

1216. A system has 5 external inputs, 4 external outputs, 3 external inquiries, 2 internal logical files and 1 external interface file, all of average complexity. Using IFPUG average weights (EI 4, EO 5, EQ 4, ILF 10, EIF 7), what is the unadjusted function point count? Application

A. 72
B. 15
C. 79
D. 89

1217. A system's unadjusted function point count is 79 and its 14 general system characteristics sum to 42. What is the adjusted function point count? Application

A. 84.53
B. 51.35
C. 121.00
D. 79.42

1218. What is the cyclomatic complexity of this function? Application

```
def grade(marks, late):
    if marks >= 80:
        g = "A"
    elif marks >= 60:
        g = "B"
    else:
        g = "C"
    if late:
        g = g + "-"
    return g
```

A. 4
B. 3
C. 2
D. 5

1219. Before release, reviews and testing found 90 defects. Users reported 10 more in the first months after release. What is the defect removal efficiency (DRE)? Application

A. 88.9%
B. 10%
C. 90%
D. 11.1%

1220. A server has a mean time to failure (MTTF) of 98 hours and a mean time to repair (MTTR) of 2 hours. What is its availability? Application

A. 98%
B. 49%
C. 96%
D. 2%

1221. A program has 10 distinct operators, 6 distinct operands, 30 operator occurrences and 18 operand occurrences. What is its Halstead volume V? Application

A. 64
B. 192
C. 768
D. 48

1222. A project is sized at 450 function points. Past projects achieved 9 FP per person-month and each person-month costs Rs 200,000. What is the estimated cost? Application

A. Rs 90,000,000
B. Rs 4,050
C. Rs 10,000,000
D. Rs 1,000,000

1223. A control flow graph has four predicate nodes, and each one has exactly two outgoing edges. What is its cyclomatic complexity? Application

A. 8
B. 6
C. 5
D. 4

1224. Using IFPUG weights (simple/average/complex): EI 3/4/6, EO 4/5/7, EQ 3/4/6, ILF 7/10/15, EIF 5/7/10. A system has 4 simple and 2 complex EIs, 3 average EOs, 2 simple EQs, 1 complex ILF and 2 simple EIFs. The 14 characteristics sum to 35. What is the adjusted FP? Application

A. 73.5
B. 70
C. 65
D. 56

1225. Using the Henry-Kafura information flow metric, complexity = length × (fan-in × fan-out)^2. A 20-line procedure has a fan-in of 3 and a fan-out of 2. What is its complexity? Application

A. 14,400
B. 120
C. 500
D. 720

1226. Counting each simple condition in a compound condition as a separate decision (as Pressman does), what is the cyclomatic complexity of this function? Application

```
def count_even_pos(xs):
    c = 0
    for x in xs:
        if x > 0 and x % 2 == 0:
            c += 1
    while c > 10:
        c -= 10
    return c
```

A. 3
B. 6
C. 5
D. 4

1227. Team A wrote 6,000 lines of assembly in 10 person-months. Team B built the same features in 1,500 lines of Python in 5 person-months. A manager says Team A was twice as productive. What is the best assessment? Analysis

A. The claim is correct because 600 LOC/PM is twice 300 LOC/PM
B. Both teams were equally productive because they built the same features
C. The claim is misleading: LOC/PM rewards verbose languages, and Team B delivered the same function with half the effort
D. Team B was less productive because Python code is shorter

1228. Module A has 30 defects in 10 KLOC. Module B has 12 defects in 2 KLOC. Where should extra reviews be focused first, and why? Analysis

A. Module A, because it has more defects in total
B. Module A, because larger modules are always riskier
C. Module B, because its defect density (6 per KLOC) is twice Module A's (3 per KLOC)
D. Both equally, because each module has defects

1229. A company starts ranking developers by LOC written per day and links it to bonuses. Which outcome is most likely? Analysis

    A. Developers write more verbose code and avoid reuse, so LOC rises while quality and maintainability may fall
    B. Defect density falls because developers write more carefully
    C. Nothing changes because developers ignore metrics
    D. Productivity rises because LOC is an objective measure of value

1230. A function has cyclomatic complexity 12. A tester shows that 7 test cases already achieve 100% branch coverage and says basis path testing needs no more. What is wrong with this reasoning? Analysis

    A. Basis path testing needs V(G) − 1 = 11 cases
    B. V(G) is an upper bound on branch coverage tests, so 7 cases is already too many
    C. Branch coverage can be reached with fewer cases than V(G); basis path testing needs one case per independent path, so up to 12 are needed
    D. Nothing: 100% branch coverage always means every independent path is tested

1231. A function has 14 binary decisions. You extract a block containing 6 of those decisions into a new helper function and call it once. What are the cyclomatic complexities afterwards? Analysis

    A. Original 9, helper 7
    B. Original 15, helper 7
    C. Original 9, helper 6
    D. Original 8, helper 6

1232. A system has 200 unadjusted FP. Its general system characteristics were first rated at a total of 30, then re-rated at 50 after the client added performance and security demands. By how many function points does the adjusted size grow? Analysis

    A. 20 FP
    B. 40 FP
    C. 0 FP, because UFP is unchanged
    D. 70 FP

1233. During a project, design and code reviews found 120 defects and testing found 60. Users found 20 after release. A manager reports DRE as 120 / (120 + 60) = 66.7%. What is the correct overall DRE, and what did the manager do wrong? Analysis

    A. 66.7%; the manager is correct
    B. 85.7%; DRE should use reviews and post-release defects only
    C. 75%; DRE = testing defects / (testing + post-release defects)
    D. 90%; the manager treated testing defects as escaped defects, but DRE compares all pre-release defects (180) with post-release ones (20)

1234. Project X (Java): 12,000 LOC, 200 FP, cost 6 million rupees. Project Y (C): 30,000 LOC, 200 FP, cost 9 million rupees. Which statement is correct? Analysis

    A. Y looks cheaper per LOC (300 vs 500) but costs more per FP (45,000 vs 30,000), so X delivered the same function more cheaply
    B. Y is cheaper because its cost per LOC is lower
    C. X and Y are equally cost-effective because both delivered 200 FP
    D. X is more expensive because it cost more per line

1235. What is the cyclomatic complexity of this function? Note that return and break statements do not add decisions. Analysis

```
def find(grid, target):
    for row in grid:
        for v in row:
            if v == target:
                return True
            elif v < 0:
                break
    return False
```

A. 6
B. 4
C. 7
D. 5

1236. System A has MTTF 500 h and MTTR 5 h. System B has MTTF 200 h and MTTR 1 h. A hospital needs the highest availability. Which should it choose? Analysis

A. A, because its MTBF is higher
B. Either, because both have availability above 99%
C. A, because it fails less often
D. B, about 99.50% available against A's 99.01%, because its fast repairs outweigh its more frequent failures

1237. In Pressman's terminology, what is an indicator? Understanding

A. The scale (nominal, ordinal, interval or ratio) on which a measure sits
B. A metric, or combination of metrics, that gives insight into a process, project or product so it can be improved
C. A single raw count, such as the number of errors found in one review
D. The tool that collects measurements automatically from the source code

1238. A defect log records each defect's type as interface, logic, data or documentation. On which measurement scale is defect type? Understanding

A. Ratio
B. Nominal
C. Interval
D. Ordinal

1239. Which of these measures is on an interval scale rather than a ratio scale? Understanding

A. The size of a module in LOC
B. The effort, in person-hours, spent on testing
C. The number of defects found in a module
D. The calendar date on which each defect was reported

1240. Which statement is meaningful only for measures on a ratio scale? Understanding

A. Module A, at 4 KLOC, is twice the size of Module B at 2 KLOC
B. Defect D3 was reported 10 days after defect D4
C. Defect D1 is in the logic category and D2 in the interface category
D. A Critical defect is more severe than a High defect

1241. Which of these is an indirect measure of software? Understanding

A. Lines of code
B. Memory size
C. Maintainability
D. Execution speed

1242. Which of these is a process metric rather than a product metric? Understanding

A. Cyclomatic complexity of the login module
B. Number of classes in the current design
C. Lines of code in release 2.0
D. Defect removal efficiency averaged across all of the organisation's projects

1243. In the CK metrics suite, what does WMC (weighted methods per class) measure? Understanding

A. The number of classes that inherit directly from the class
B. The depth of the class in the inheritance tree
C. The sum of the complexities of a class's methods, which equals the method count if every method has weight 1
D. The number of other classes the class is coupled to

1244. In the CK metrics suite, what does NOC (number of children) count? Understanding

A. The methods the class overrides from its parent
B. The objects created from the class at run time
C. The immediate subclasses of a class
D. All descendants of a class at every level below it

1245. A class has a high CBO (coupling between object classes) value. What is the most likely consequence? Understanding

A. It depends on many other classes, so changes elsewhere are more likely to affect it and it is harder to reuse or test alone
B. It sits deep in the inheritance tree, so it inherits many methods
C. Its methods share many instance variables, so it is highly cohesive
D. It has many subclasses, so it is heavily reused

1246. In the CK metrics suite, what does RFC (response for a class) count? Understanding

A. The class's own methods plus the methods of other classes that they call directly
B. The messages the class receives at run time during testing
C. Only the public methods the class exposes to callers
D. The number of classes that call the class

1247. In Halstead's software science, what is the program length N? Understanding

A. The total occurrences of operators plus the total occurrences of operands (N1 + N2)
B. The number of physical lines in the program
C. The distinct operators plus the distinct operands (n1 + n2)
D. N1 + N2 multiplied by log2 of the vocabulary

1248. In structured design, what is the fan-in of a module? Understanding

A. The number of modules it calls
B. The number of decisions inside it
C. The number of parameters it accepts
D. The number of modules that call it

1249. How do Capers Jones's feature points extend function points? Understanding

A. They drop the value adjustment factor so only the unadjusted count is used
B. They replace the five function types with a count of classes
C. They add a count of algorithms, which suits compute-heavy systems such as real-time or scientific software
D. They count lines of code per feature instead of functions

1250. In COCOMO II's application composition model, object points (application points) are counted from what? Understanding

A. The classes, methods and attributes in the class diagram
B. The external inputs, outputs, inquiries and files
C. The lines of code in each compiled object file
D. The screens, reports and 3GL components the application needs

1251. What does ROCOF (rate of occurrence of failures) express? Understanding

A. The probability that a single demand on the system fails
B. The average time needed to repair a failure
C. The percentage of time the system is available for use
D. How often failures occur over operating time, such as 2 failures per 1,000 hours

1252. How is MTBF related to MTTF and MTTR? Understanding

A. MTBF = MTTF + MTTR
B. MTBF = MTTF × MTTR
C. MTBF = MTTF ÷ (MTTF + MTTR)
D. MTBF = MTTF − MTTR

1253. Cyclomatic complexity can be computed from source code without running the program. What kind of metric is it? Understanding

A. A project metric
B. A dynamic product metric
C. A static product metric
D. A process metric

1254. A team uses the GQM approach. Its goal is to improve the reliability of the billing module. Which item belongs at the Question level? Understanding

A. Buy a static analysis tool for the billing team
B. Is the failure rate of billing falling from one release to the next?
C. Improve billing reliability before the next release
D. Number of failures per 1,000 billing transactions

1255. A screen shows each region's monthly sales with a calculated total and each region's percentage share. Under IFPUG rules, how is this elementary process classified? Understanding

A. External input (EI)
B. External inquiry (EQ)
C. External interface file (EIF)
D. External output (EO)

1256. Which of these is a function-oriented metric rather than a size-oriented one? Understanding

A. Defects per function point
B. Cost per line of code
C. Defects per KLOC
D. Pages of documentation per KLOC

1257. Sommerville separates control metrics from predictor metrics. Which of these is a predictor metric? Understanding

A. The time taken to complete each process stage
B. The number of change requests processed per month
C. The cyclomatic complexity of a module, used to estimate how hard it will be to maintain
D. The average effort needed to repair reported defects

1258. Pressman suggests mean time to change (MTTC) as a measure of which quality attribute? Understanding

A. Portability
B. Maintainability
C. Usability
D. Reliability

1259. Which of these is NOT a valid way to compute McCabe's cyclomatic complexity for one connected flow graph? Understanding

A. Nodes − edges + 2
B. Edges − nodes + 2
C. Number of predicate nodes + 1
D. Number of regions of the planar flow graph

1260. Under IFPUG rules, the complexity (simple, average or complex) of an internal logical file is set from which counts? Understanding

A. The number of transactions that read or update it
B. Its data element types (DETs) and file types referenced (FTRs)
C. Its size in records and bytes
D. Its data element types (DETs) and record element types (RETs)

1261. A release of 120 function points had 18 defects reported against it. What is its defect density per FP? Application

A. 1.5
B. 6.67
C. 0.15
D. 0.015

1262. A 600 FP system was built by 4 developers over 5 months. What is the productivity in FP per person-month? Application

A. 120
B. 30
C. 150
D. 66.7

1263. A project of 18,000 LOC cost Rs 5.4 million. What is its cost per line of code? Application

A. Rs 30
B. Rs 3,000
C. Rs 300
D. Rs 333

1264. A program's flow graph has two connected components (the main routine and one subroutine), with 20 edges and 17 nodes in total. Using V(G) = E − N + 2P, what is the cyclomatic complexity? Application

A. 4
B. 7
C. 5
D. 3

1265. What is the cyclomatic complexity of this function? Else branches and return statements add no decisions. Application

```
def grade(marks):
    if marks >= 85:
        g = "A"
    elif marks >= 70:
        g = "B"
    elif marks >= 50:
        g = "C"
    else:
        g = "F"
    attempts = 0
    while attempts < 2:
        attempts += 1
    return g
```

A. 6
B. 5
C. 3
D. 4

1266. What is the cyclomatic complexity of this C function? Count a switch with k case labels plus a default as k decisions. Application

```
int fee(int type, int late) {
    int f;
    switch (type) {
        case 1: f = 100; break;
        case 2: f = 150; break;
        case 3: f = 200; break;
        default: f = 0;
    }
    if (late) f += 50;
    return f;
}
```

A. 3
B. 6
C. 5
D. 4

1267. A program has 10 distinct operators, 12 distinct operands, 40 operator occurrences and 25 operand occurrences. What is its Halstead length N? Application

A. 65
B. 1,000
C. 22
D. 87

1268. Halstead difficulty is D = (n1 ÷ 2) × (N2 ÷ n2). A program has n1 = 12 distinct operators, n2 = 8 distinct operands and N2 = 40 operand occurrences. What is D? Application

A. 30
B. 60
C. 5
D. 6

1269. A program has n1 = 8, n2 = 8, N1 = 40 and N2 = 24. Using V = N × log2(n), D = (n1 ÷ 2) × (N2 ÷ n2) and effort E = D × V, what is Halstead effort E? Application

A. 12
B. 3,072
C. 256
D. 6,144

1270. A system has 120 unadjusted function points. All 14 general system characteristics are rated 0 (no influence). What is the adjusted FP count? Application

A. 0
B. 78
C. 162
D. 120

1271. Using IFPUG weights (simple/average/complex): EO 4/5/7, EQ 3/4/6, ILF 7/10/15. A system has 3 complex EOs, 2 average ILFs and 4 simple EQs. What is the unadjusted FP count? Application

A. 57
B. 51
C. 63
D. 53

1272. A system's unadjusted function point count is 100. What range can its adjusted FP count fall in? Application

A. 0 to 100
B. 70 to 130
C. 65 to 135
D. 100 to 170

1273. A service has an MTTF of 495 hours. The contract requires at least 99% availability. What is the longest average repair time (MTTR) that still meets it? Application

A. 4.95 hours
B. 10 hours
C. 5 hours
D. 0.5 hours

1274. A system ran for 1,200 operating hours and failed 6 times. What is its ROCOF? Application

A. 200 failures per 1,000 hours
B. 5 failures per 1,000 hours
C. 0.5 failures per 1,000 hours
D. 6 failures per 1,000 hours

1275. An emergency shutdown system was triggered 1,500 times in testing and failed to respond 3 times. What is its POFOD? Application

A. 0.003
B. 500
C. 0.02
D. 0.002

1276. Design reviews found 40 design defects. Coding and testing later found another 10 defects that had been introduced in design. What was the defect removal efficiency of the design review? Application

A. 25%
B. 75%
C. 20%
D. 80%

1277. Pressman's software maturity index is SMI = [MT − (Fa + Fc + Fd)] ÷ MT. Release 6 has MT = 500 modules; 10 were added, 25 changed and 5 deleted since release 5. What is SMI? Application

A. 0.95
B. 0.92
C. 0.08
D. 0.93

1278. Using the Henry-Kafura metric, complexity = length × (fan-in × fan-out)^2. A 50-line procedure has a fan-in of 2 and a fan-out of 4. What is its complexity? Application

A. 3,200
B. 400
C. 1,800
D. 64

1279. A system has 200 unadjusted FP and an adjusted count of 250 FP. What must the 14 general system characteristics sum to? Application

A. 125
B. 60
C. 25
D. 50

1280. A system is estimated at 36 KLOC. The organisation's past productivity is 600 LOC per person-month. If 6 developers work on it, roughly how many months will it take? Application

A. 6
B. 60
C. 10
D. 100

1281. Using the original CK definition, LCOM = P − Q if P > Q, else 0, where P counts method pairs sharing no instance variable and Q counts pairs that share one. Method M1 uses {a, b}, M2 uses {b, c} and M3 uses {d}. What is LCOM? Application

A. 1
B. 3
C. 2
D. 0

1282. Module A has 200 LOC and cyclomatic complexity 25. Module B has 600 LOC and cyclomatic complexity 8. With time to refactor only one for testability, which should you choose? Analysis

A. A, because V(G) of 25 is far above McCabe's 10 and needs 25 basis paths to test
B. Either, because both modules have the same defect density
C. Neither, because together they are under 1 KLOC
D. B, because it is three times longer, so it must be harder to test

1283. After a team doubled its testing effort and added code reviews, defect density found before release rose from 2 to 4 per KLOC, while defects reported by customers fell. What is the best interpretation? Analysis

A. Customers have stopped reporting defects, so the field data should be ignored
B. Detection became more effective, so DRE improved; the higher in-house density is not evidence of worse code
C. Code quality halved, because twice as many defects were injected
D. The metric is broken, because defect density can never rise when effort increases

1284. A team installs a tool that reports 40 metrics, collects all of them for a year, then meets to decide what the numbers mean. Nobody uses the data. What does GQM say went wrong? Analysis

A. They collected too few metrics; GQM recommends measuring everything a tool can produce
B. A year is too short; metrics only become useful after five years
C. They started with metrics; GQM starts from goals, derives questions, and collects only the metrics that answer them
D. They should have used ordinal scales instead of ratio scales

1285. Defect severity is coded 1 = Low, 2 = Medium, 3 = High, 4 = Critical. A manager reports 'average severity 2.6 this release, down from 2.8'. What is the best assessment? Analysis

A. Severity is nominal, so even ranking Critical above Low is invalid
B. The report is valid, because the codes are numbers and any numbers can be averaged
C. Severity is a ratio scale, so the manager should also say severity fell by 7%
D. Severity is ordinal, so the gaps between codes are not equal and a mean is not meaningful; report the median or the count in each level

1286. A class has NOC = 15. What is the most sensible conclusion? Analysis

A. A change to it can affect all 15 subclasses, so it needs careful testing, and such a high NOC may signal misused subclassing
B. It sits 15 levels deep in the hierarchy, so it inherits a lot of behaviour
C. It has 15 methods, so it needs exactly 15 test cases
D. High NOC always means excellent reuse, so the class needs no special attention

1287. A manager starts paying a bonus to the team with the fewest defects logged each month. What is the most likely result? Analysis

A. Testers log more defects to look busy, so counts rise
B. Defects get reported less, merged or argued away, so the count falls while real quality may not change
C. Nothing changes, because metrics do not affect behaviour
D. Developers write better code, so true defects fall and the metric stays accurate

1288. Project J (Java): 15 KLOC, 150 FP, 30 defects. Project C (COBOL): 40 KLOC, 150 FP, 60 defects. Both deliver the same functionality. Which product has the better defect quality? Analysis

A. C, because it has more code and so more tested lines
B. C, because 1.5 defects per KLOC is lower than J's 2
C. They are equal, because both deliver 150 FP
D. J: 0.2 defects per FP against C's 0.4, even though C looks better per KLOC (1.5 against 2)

1289. A telephone exchange must be usable around the clock. Short outages that are fixed within seconds are tolerable; long downtime is not. Following Sommerville, which reliability metric fits best? Analysis

A. POFOD, because each call is a demand that may fail
B. ROCOF, because the number of failures matters more than how long each lasts
C. Defect density, because it measures how reliable the code is
D. Availability (AVAIL), because what matters is the share of time the service can be used, including repair time

1290. The first version of fee() below is replaced by the second. How does cyclomatic complexity change, and what does that mean for testing? Analysis

```
def fee(code):
    if code == "A":
        return 100
    elif code == "B":
        return 150
    elif code == "C":
        return 200
    elif code == "D":
        return 250
    elif code == "E":
        return 300
    elif code == "F":
        return 350
    elif code == "G":
        return 400
    else:
        return 0

# --- replaced by ---

FEES = {"A": 100, "B": 150, "C": 200, "D": 250,
        "E": 300, "F": 350, "G": 400}

def fee(code):
    return FEES.get(code, 0)
```

A. It falls from 8 to 1; the branching now lives in a data table, so the fee table still needs checking even though the control flow has one path
B. It stays at 8, because the dictionary still has seven entries
C. It falls to 0, so the function no longer needs any test
D. It falls from 8 to 7, because only the else branch was removed

1291. Counting a registration form, an analyst records one external input for each of its 10 fields. What is wrong with this count? Analysis

A. The form should be counted as an internal logical file, not as inputs
B. Nothing; each field is a separate EI under IFPUG rules
C. The form is one elementary process, so it is one EI; the 10 fields are data element types that set its complexity
D. The fields should be counted as 10 external inquiries instead

1292. A team reports 50,000 LOC delivered in 10 person-months, but 30,000 of those lines were copied unchanged from an internal library. How should productivity be reported? Analysis

A. 5,000 LOC per PM, because all the lines are in the product
B. Productivity cannot be reported at all when any code is reused
C. 3,000 LOC per PM, counting only the reused code
D. Separate new and reused code, for example 2,000 new LOC per PM, and track reuse on its own

1293. A server has MTTF 200 h and MTTR 4 h. Option X doubles MTTF to 400 h. Option Y halves MTTR to 2 h. Which gives higher availability? Analysis

A. They give exactly the same availability, about 99.01%
B. X gives about 99.5% and Y about 99.0%
C. X, because preventing failures always beats repairing faster
D. Y, because repair time dominates the availability formula

1294. A vendor states MTBF = 100 hours and MTTR = 4 hours. A student computes availability as 100 ÷ (100 + 4) ≈ 96.15%. What is the correct availability? Analysis

A. 100%, because MTBF already includes repair time
B. 4%, because MTTR ÷ MTBF gives the availability
C. 96%, because MTTF = MTBF − MTTR = 96 h, so availability = 96 ÷ 100
D. 96.15%; the student's working is correct

1295. A manager proposes measuring each developer's productivity by the number of Git commits they push per week. What is the main weakness? Analysis

A. Commit counts are too hard for developers to influence
B. Commits measure activity, not delivered function or quality, and are easy to inflate by splitting work into tiny commits
C. Commits are a ratio-scale measure, so they cannot be compared
D. Commit counts cannot be collected automatically

1296. Project P counted physical lines including comments and blank lines; Project Q counted logical statements only. P reports 3 defects per KLOC and Q reports 5. What should you conclude? Analysis

A. P has better quality, because 3 is lower than 5
B. The difference proves comment-heavy code has fewer defects
C. Q has better quality, because logical statements are more accurate
D. The figures cannot be compared until both projects use the same LOC counting rule

Software Configuration Management MCQs for NSCT

1297. In software configuration management (SCM), what is a software configuration item (SCI)? Understanding

A. Only the final executable delivered to the customer
B. Any work product placed under configuration control, such as a requirement document, source file or test plan
C. A team member responsible for version control
D. A setting in the application's configuration file

1298. What is a baseline in SCM? Understanding

A. The oldest commit in a repository
B. The minimum hardware needed to run the software
C. A formally reviewed and agreed version of one or more configuration items that can be changed only through formal change control
D. The first draft of any document

1299. What is the main role of a change control board (CCB)? Understanding

A. To run the automated build every night
B. To evaluate change requests and decide whether to approve, reject or defer them
C. To write the code for every approved change
D. To assign version numbers to source files

1300. In Git, what does git add do? Understanding

A. Copies the current content of files into the staging area (index) for the next commit
B. Uploads files to the remote repository
C. Creates a new branch
D. Creates a new commit immediately

1301. Which Git command creates a local copy of an existing remote repository, including its full history? Understanding

A. git init
B. git fetch
C. git branch
D. git clone

1302. How does a distributed version control system such as Git differ from a centralised one such as Subversion (SVN)? Understanding

A. Git stores only the latest version of each file
B. Git needs a central server for every commit
C. Every clone holds the full repository history, so you can commit, branch and view history offline
D. SVN supports branching but Git does not

1303. Under semantic versioning (MAJOR.MINOR.PATCH), when must the MAJOR number be increased? Understanding

A. When a bug is fixed
B. When documentation is updated
C. Every calendar year
D. When a change breaks backward compatibility of the public API

1304. By default, what does git pull do? Understanding

A. Uploads your commits to the remote
B. Deletes local commits not on the remote
C. git fetch followed by git merge of the upstream branch into the current branch
D. Only downloads new commits without changing your branch

1305. What is a .gitignore file used for? Understanding

A. To delete files from the repository history
B. To stop Git tracking files that are already committed
C. To list untracked files and patterns that Git should not offer to add, such as build output or local secrets
D. To hide commits from git log

1306. Which of these is NOT one of the standard SCM activities described by IEEE 828 and Pressman? Understanding

A. Unit testing
B. Configuration auditing
C. Configuration status accounting
D. Configuration identification

1307. What does a physical configuration audit check? Understanding

A. That the server room is physically secure
B. That the software meets its functional requirements in testing
C. That the developers followed coding standards
D. That the delivered items and their versions match the technical documentation for that baseline

1308. In Sommerville's terms, what is a variant (as opposed to a release)? Understanding

A. One of several versions of a system that exist side by side for different platforms or customers
B. A version with a fixed bug
C. A temporary branch deleted after merging
D. A version distributed to customers

1309. Which concurrency model does Git use for ordinary text files? Understanding

A. Lock-modify-unlock: one developer locks a file before editing
B. Copy-modify-merge: developers edit independently and merge changes afterwards
C. Timestamp ordering: the latest save overwrites all others
D. Exclusive check-out: only the project manager may edit files

1310. How does an annotated Git tag differ from a lightweight tag? Understanding

A. A lightweight tag can be pushed but an annotated tag cannot
B. An annotated tag is a full tag object storing tagger, date and message; a lightweight tag is just a name pointing at a commit
C. An annotated tag points to a branch, a lightweight tag to a file
D. There is no difference except the command name

1311. A bad commit has already been pushed to a shared main branch. Why is git revert usually preferred over git reset to undo it? Understanding

A. git reset cannot move a branch backwards
B. git revert adds a new commit that undoes the change, so shared history is not rewritten
C. git revert works offline and git reset does not
D. git revert deletes the commit from every clone automatically

1312. What is the main idea of trunk-based development? Understanding

A. Only the release manager may commit to the main branch
B. Each feature lives on its own branch for months before merging
C. Developers integrate small changes into one shared main branch at least daily, using short-lived branches if any
D. Every developer keeps a private fork that is never merged

1313. A library at version 2.4.1 adds a new function without changing any existing behaviour. Under semantic versioning, what is the next version? Application

A. 2.5.0
B. 2.5.1
C. 2.4.2
D. 3.0.0

1314. A library at version 2.4.1 renames a public method, so existing callers break. Under semantic versioning, what is the next version? Application

A. 2.4.2
B. 2.5.0
C. 3.0.0
D. 3.4.1

1315. You edit a tracked file and run git add on it. What does plain git diff (with no options) now show? Application

A. An error, because staged changes cannot be diffed
B. Nothing, because git diff compares the working tree with the index, and they now match
C. The full change, because git diff compares with the last commit
D. Only the names of changed files

1316. Starting from one commit A on main, you run: git switch -c feature, commit B, git switch main, git merge feature. No other commits exist. What happens? Application

A. Nothing, because feature must be pushed first
B. A three-way merge creates a merge commit, giving 3 commits
C. A merge conflict, because both branches changed
D. A fast-forward: main moves to B, history has 2 commits and no merge commit

1317. After commit A on main, you branch feature and commit B there, then go back to main and commit C (touching a different file), and run git merge feature. How many commits does main's history contain? Application

A. 5, including two merge commits
B. 4, including one merge commit
C. 3, with no merge commit
D. 2, because B replaces C

1318. You run git reset --soft HEAD~1 right after committing a change to f. What state are you in? Application

A. The last commit is undone, and its changes are unstaged in the working tree
B. Nothing changes, because --soft only previews the reset
C. The last commit is undone, and its changes are deleted
D. The last commit is undone, and its changes to f are still staged

1319. You run git reset --hard HEAD~1 right after committing a change to f, with no other changes. What happens to that change? Application

A. It is moved to the stash
B. It stays in the working tree but is unstaged
C. It is removed from the branch, the index and the working tree
D. It stays staged

1320. History is A then B, where B added a line to f. You run git revert HEAD. What is the result? Application

A. Git opens a merge conflict
B. B is deleted, and history has 1 commit
C. History has 2 commits, and f still contains B's line
D. A new commit C undoes B, history has 3 commits, and f matches its state at A

1321. You modify the tracked file f and create a new file new.txt, then run git commit -am "update". What is committed? Application

A. Only the change to f; new.txt stays untracked
B. Neither, because -a needs git add first
C. Both f and new.txt
D. Only new.txt

1322. You run git rm --cached config.local on a tracked file. What happens? Application

A. Git stages the file's removal from the repository, but the file stays on disk as untracked
B. Nothing, because --cached only previews the command
C. The file is removed from history in all past commits
D. The file is deleted from disk and from Git

1323. Git tags v1.9.0 and v1.10.0 exist. Under semantic versioning precedence, which is newer? Application

A. v1.10.0, because each part is compared as a number, and 10 > 9
B. It cannot be decided without dates
C. They are equal
D. v1.9.0, because 9 comes after 1 when compared as text

1324. Branch ft has commits B and C after A. On main (still at A) you run git merge --squash ft. What does git log on main show immediately afterwards? Application

A. Only commit A; the combined changes from B and C are staged but not committed
B. A, B, C and a merge commit
C. A and one new squash commit
D. A, B and C

1325. A merge stops with a conflict. f contains the block below. You were on main and merging ft. Which line came from main? Application

```
<<<<<<< HEAD
c
=======
b
>>>>>>> ft
```

A. Neither; Git invents placeholder lines
B. b
C. Both lines came from ft
D. c

1326. git push origin main fails with "! [rejected] main -> main (fetch first)". What is the correct fix? Application

A. Delete your local repository and clone again, losing your commit
B. Run git commit again so the push succeeds
C. Fetch and integrate the remote commits (git pull or fetch plus rebase), resolve any conflicts, then push again
D. Run git push --force to overwrite the remote

1327. A developer committed secret.env last week. Today they add secret.env to .gitignore, then edit the file. git status still lists it as modified. Why, and what should they do? Analysis

A. Git always shows ignored files as modified; it is harmless
B. .gitignore needs a restart of Git; rerun git status later
C. The pattern is wrong; .gitignore only matches folders
D. .gitignore does not affect tracked files; run git rm --cached secret.env, commit, and rotate the secret, because it is still in history

1328. A developer pushed a broken commit to the shared main branch, then ran git reset --hard HEAD~1 and git push --force. Two teammates had already pulled. What is the best assessment? Analysis

A. It made no difference, because teammates' clones update automatically
B. It was the wrong tool: rewriting shared history breaks teammates' clones; git revert would have undone the change with a new commit
C. It was correct, as long as the developer tells the team
D. It was correct, because reset --hard is the standard way to undo pushed commits

1329. The SRS was baselined last month. A client calls a developer directly and asks for a new report. The developer adds it that afternoon. What has gone wrong from an SCM viewpoint? Analysis

A. The change bypassed change control: it needed a change request, impact analysis and a CCB decision before the baseline was changed
B. Only the version number was not updated
C. The developer should have waited for the next sprint but needed no approval
D. Nothing, because the client is always right

1330. Branch ft (commit B) is rebased onto main (A then C). Which statement about the result is true? Analysis

A. Rebase deletes commit C from main
B. ft's history becomes A, C, B' in a straight line; B' has a new hash, so rebasing a branch others already use rewrites their history
C. Rebase is identical to merge, so nothing changes for teammates
D. ft keeps commit B with the same hash and adds a merge commit

1331. A fix commit H on the feature branch is cherry-picked onto main. Which statement is true? Analysis

A. main and feature now share the exact commit H
B. main gets a new commit with the same change but a different hash from H
C. Cherry-pick merges the whole feature branch into main
D. H is removed from the feature branch

1332. You run git checkout on an old commit hash, make a fix and commit it, then run git switch main. What happens to the new commit, and how should you have kept it? Analysis

A. Git refuses to let you commit in detached HEAD
B. It is automatically added to main
C. It is pushed to the remote automatically
D. It is left unreferenced (detached HEAD) and can be lost to garbage collection; create a branch with git switch -c fix before switching away

1333. A developer has a modified tracked file f and a new untracked file u.txt. They run git stash and switch branches. Why is u.txt still in the working tree? Analysis

A. Switching branches copies all files to the new branch
B. git stash always deletes untracked files
C. u.txt was committed automatically
D. Plain git stash saves only tracked changes; untracked files need git stash -u (--include-untracked)

1334. A bug appeared somewhere in 1,024 commits between a known good release and the current bad build. Using git bisect (binary search), at most how many builds must you test to find the first bad commit? Analysis

A. 32
B. 512
C. 10
D. 1,024

1335. The release notes for version 4.2 list library X at version 2.3, but the shipped build contains X 2.1. All functional tests passed. Which SCM activity should have caught this? Analysis

A. The daily stand-up meeting
B. A functional configuration audit, which checks behaviour against requirements
C. A physical configuration audit, which checks the as-built items and versions against the documentation
D. Unit testing of the library

1336. A game studio stores large binary art files (.psd and 3D models) in Git. Two artists often edit the same file and lose work, because Git cannot merge binary content. Which approach best fits? Analysis

A. Use lock-modify-unlock for those files, for example Git LFS file locking, so only one person edits a binary at a time
B. Use semantic versioning for the art files
C. Rebase more often so binary conflicts disappear
D. Keep copy-modify-merge and resolve conflicts by hand-merging the binary files

1337. Which SCM activity records and reports the status of each configuration item and change request, such as which changes are approved, pending or done? Understanding

A. Configuration identification
B. Configuration auditing
C. Configuration status accounting
D. Version control

1338. In Sommerville's terms, what is a release? Understanding

A. A branch created for experimental work
B. A version that exists alongside others for a different platform
C. A version of the system that is distributed to customers
D. Any instance of a configuration item that differs from other instances

1339. What does git init do in a folder? Understanding

A. Commits all the files in the folder
B. Downloads a copy of a repository from a server
C. Creates a new, empty Git repository by adding a .git directory there
D. Connects the folder to a GitHub account

1340. You are on main and run git branch feature. Which branch is checked out afterwards? Understanding

A. None; HEAD becomes detached
B. main
C. Both, until the next commit
D. feature

1341. In change control, what is the purpose of impact analysis? Understanding

A. To merge the change into the baseline
B. To record the change in the status accounting log
C. To test the change after it has been released
D. To estimate the cost, schedule, risk and affected items of a proposed change before the CCB decides

1342. Which SCM activity selects the items to be controlled and gives each a unique name and version? Understanding

A. Configuration auditing
B. Configuration identification
C. Configuration status accounting
D. Change control

1343. After you run git clone, what is origin? Understanding

A. A backup copy kept inside .git
B. The default local branch
C. The first commit in the history
D. The default name Git gives the remote repository you cloned from

1344. Why are compiled binaries and other build outputs usually kept out of a Git repository? Understanding

A. Git cannot store binary files at all
B. Build outputs can never be configuration items in any project
C. They change too rarely to be worth storing
D. They can be regenerated from versioned sources by a repeatable build, and storing them bloats the repository and causes needless conflicts

1345. Where does git commit record a change? Understanding

A. On the remote server, immediately
B. Only in the working tree
C. Only in the staging area
D. In the local repository; it reaches the remote only when you push

1346. Under semantic versioning, how does 2.0.0-beta compare with 2.0.0? Understanding

A. 2.0.0-beta has higher precedence, because it has an extra label
B. 2.0.0-beta has lower precedence, because a pre-release comes before its normal version
C. They have equal precedence, because the numbers match
D. 2.0.0-beta is not a valid SemVer version

1347. Under semantic versioning, what does a version such as 0.7.3 signal? Understanding

A. Initial development: the public API should not be considered stable, and anything may change
B. A patch release of version 7.3
C. The seventh release candidate of version 3
D. A deprecated version that should not be used

1348. Which four object types does Git store in its object database? Understanding

A. Blob, branch, commit and stash
B. Diff, patch, commit and merge
C. Blob, tree, commit and tag
D. File, folder, branch and remote

1349. You amend only the message of your last commit, leaving its files unchanged. What happens to the commit's hash? Understanding

A. It changes only when you push
B. It stays the same, because the files did not change
C. It changes, because the hash covers the commit's metadata, including the message, not only its file tree
D. Only the tree's hash changes

1350. In the Git Flow branching model, where does a hotfix branch start, and where is it merged? Understanding

A. It starts from the current release branch and is never merged
B. It starts from master (the production branch) and is merged back into both master and develop
C. It starts from develop and is merged only into develop
D. It starts from a feature branch and is merged into that feature

1351. What does git reflog show? Understanding

A. A local record of where HEAD and branch tips have pointed, which helps you find commits lost after a reset or rebase
B. The list of all tags in the repository
C. The commit history of the remote repository
D. A log of edits to Git's reference documentation

1352. In Sommerville's terms, what is a codeline? Understanding

A. A single line of source code under version control
B. The command used to build the system
C. The line that separates two versions in a merge conflict
D. A sequence of versions of source code, where later versions are derived from earlier ones

1353. What does git merge --no-ff do when a fast-forward would have been possible? Understanding

A. It creates a merge commit anyway, so the branch's work stays grouped in history
B. It refuses the merge
C. It squashes the branch into one commit without committing
D. It rebases the branch before merging

1354. main and feature have diverged: each has a commit the other lacks. What happens when you run git merge --ff-only feature on main? Understanding

A. Git rebases feature onto main automatically
B. Git creates a merge commit
C. Git refuses and aborts, because a fast-forward is impossible
D. Git fast-forwards main and drops main's own commit

1355. How does Git's storage model differ from the per-file delta model of older tools such as RCS? Understanding

A. Git keeps only the latest version and drops older ones from clones
B. Git stores a separate full copy of every file in every commit with no sharing
C. Git stores only the differences between consecutive versions of each file
D. Git records each commit as a snapshot of the whole tree, reusing unchanged files by hash; deltas appear only as compression inside packfiles

1356. In a three-way merge, which three versions does Git compare? Understanding

A. The working tree, the index and HEAD
B. The two branch tips and the remote's copy
C. The first commit and the two branch tips
D. The two branch tips and their merge base (most recent common ancestor)

1357. History is A then B, with B the latest commit on your branch. You run git reset HEAD~1 with no option. What state are you in? Application

A. The branch points at A, and B's changes are discarded
B. A new commit is added that undoes B
C. The branch points at A, and B's changes are staged
D. The branch points at A, and B's changes are in the working tree but not staged

1358. f contains v1 in the last commit. You change it to v2 and run git add f, then change it to v3 without staging. You run git restore f. What does f contain now? Application

A. v2
B. Nothing; f is deleted
C. v1
D. v3

1359. f's last commit has one line, 'one'. You append 'two' and run git add f, then append 'three' without staging. Which added line does git diff --staged show? Application

A. Both 'two' and 'three'
B. Neither line
C. Only 'two'
D. Only 'three'

1360. In a repository with earlier commits, you create a.txt and b.txt, run git add a.txt b.txt, then git restore --staged b.txt, then git commit -m c. What is committed? Application

A. Neither file, because restore clears the whole index
B. Only b.txt
C. Both files
D. Only a.txt; b.txt stays on disk as an untracked file

1361. Branch feature has a commit that is not merged into main. On main you run git branch -d feature. What happens? Application

A. The branch and its commit are deleted immediately
B. Git merges feature into main first, then deletes it
C. Git refuses with 'not fully merged'; git branch -D would force the deletion
D. Only the remote copy of the branch is deleted

1362. With tracked file f, you make change 1 and run git stash, then make change 2 and run git stash again, then run git stash pop. How many entries does git stash list show, and which change is in f? Application

A. Two entries; f has change 2
B. One entry; f has change 2
C. One entry; f has change 1
D. No entries; f has both changes

1363. main is at commit A. Branch feature adds commit B on top of A. On main you run git merge --no-ff feature. How many commits does git log on main show? Application

A. 4
B. 1
C. 2
D. 3

1364. You create the tag v1.0 locally, then run git push origin main. What does the remote now have? Application

A. The commits on main but not the tag; tags need git push origin v1.0 or git push --tags
B. Nothing, because a tagged commit cannot be pushed without the tag
C. Only the tag
D. The commits on main and the tag

1365. .gitignore contains the two lines *.log and !keep.log. The folder has untracked files a.log and keep.log. Apart from .gitignore itself, which file does git status list as untracked? Application

A. keep.log only
B. a.log only
C. Both files
D. Neither file

1366. A .gitignore in the repository root contains the single line /temp. Untracked files exist at temp/x and src/temp/y. Which file appears in git status? Application

A. temp/x only
B. src/temp/y only
C. Neither file
D. Both files

1367. main has A then C; feature has A then B, and B touches a different file. On feature you run git rebase main, then on main you run git merge feature. What does main's history look like? Application

A. A, C and a merge commit
B. A, C, B' in a straight line, because the merge is a fast-forward and adds no merge commit
C. A, B, C and a merge commit
D. A, B, C with B keeping its original hash

1368. A library is at 1.4.2. In order, the team releases a backward-compatible new feature, then a bug fix, then a breaking API change. What are the three new versions? Application

A. 1.5.2, 1.5.3, 2.0.0
B. 1.4.3, 1.4.4, 1.5.0
C. 1.5.0, 1.5.1, 2.0.0
D. 1.5.0, 1.6.0, 2.0.0

1369. Sort these versions from lowest to highest semantic versioning precedence: 2.0.0, 2.0.0-rc.1, 2.0.0-beta.11, 2.0.0-beta.2. Application

A. 2.0.0, 2.0.0-beta.2, 2.0.0-beta.11, 2.0.0-rc.1
B. 2.0.0-beta.2, 2.0.0-beta.11, 2.0.0-rc.1, 2.0.0
C. 2.0.0-beta.11, 2.0.0-beta.2, 2.0.0-rc.1, 2.0.0
D. 2.0.0-rc.1, 2.0.0-beta.2, 2.0.0-beta.11, 2.0.0

1370. On main you have uncommitted edits to f. Branch ft has a different committed version of f. You run git switch ft. What happens? Application

A. Git switches and silently discards your edits
B. Git refuses, saying your local changes would be overwritten; commit or stash them first
C. Git creates a new branch called ft
D. Git switches and merges your edits into ft's version automatically

1371. During a merge, f has a conflict. You edit f, remove the conflict markers and save. Which command marks f as resolved before you commit? Application

A. git add f
B. git reset f
C. git resolve f
D. git merge --abort

1372. main has commits A then D. Branch ft was made from A and has commits B then C. What does git log main..ft list? Application

A. C and B
B. D only
C. B, C and D
D. A, B and C

1373. History on your branch is A, B, C, D, with D the newest commit at HEAD. Which commit is HEAD~2? Application

A. B
B. C
C. A
D. D

1374. .gitignore lists *.tmp. Untracked items are u.txt, i.tmp and a folder d/ containing z. You run git clean -f. What is removed? Application

A. Only u.txt
B. u.txt, d/ and i.tmp
C. Nothing, because -n is required first
D. u.txt and d/

1375. You run git rm f on a tracked, unmodified file f. What happens? Application

A. f is removed from every commit in history
B. Its removal is staged, but f stays on disk
C. f is deleted from the working tree, and its removal is staged
D. f is deleted from disk, but nothing is staged

1376. You run git mv old.py new.py on a tracked file. What does git status --short report? Application

A. Nothing, because Git does not track renames
B. M old.py
C. D old.py and ?? new.py
D. R old.py -> new.py, a staged rename

1377. A teammate has pushed 2 new commits to origin/main. You run git fetch, then git status. What does Git report, and what is in your working files? Application

    A. Your branch is ahead of origin/main by 2 commits
    B. Your branch and origin/main have diverged
    C. Your branch is up to date; your files now contain the new commits
    D. Your branch is behind origin/main by 2 commits and can be fast-forwarded; your files are unchanged

1378. Merge commit M brought feature into main, but it was faulty, so you ran git revert -m 1 M. Feature was later fixed in the working copy, with no new commits yet, and you run git merge feature. What happens? Analysis

    A. Git undoes the revert commit automatically
    B. Git re-applies all of feature's changes, restoring the reverted work
    C. Git raises a conflict for every file feature touched
    D. Git reports 'Already up to date': feature's commits are already in main's history, so you must revert the revert or add new commits

1379. Five developers each work on a separate branch for three months and merge in the final week. Merges take days and break the build. What is the best fix for the next release? Analysis

    A. Integrate small changes into main at least daily, checked by an automated build and tests (CI or trunk-based development)
    B. Squash each branch into one commit before merging
    C. Lock main so nobody merges until release week
    D. Move to a centralised version control system

1380. Version 2.3.0, tagged v2.3.0, is in production. main already holds unfinished 2.4 features. A security bug must be fixed today. What is the best approach? Analysis

    A. Patch the production server directly and commit the change later
    B. Branch from v2.3.0, fix and release 2.3.1, then merge or cherry-pick the fix into main
    C. Fix it on main and release main as 2.3.1
    D. Revert all 2.4 commits on main, fix, then re-apply them

1381. You pushed commit A to origin/main, then ran git commit --amend to fix its message. git push is rejected as non-fast-forward. Why, and what should you do on a shared branch? Analysis

    A. Amend deleted your local branch
    B. The remote is down; retry later
    C. Amending replaced A with a new commit, so your main and origin/main have diverged; leave the pushed commit alone or coordinate before any force-push
    D. Amend keeps the same hash, so Git sees nothing new to push

1382. A release builds on one developer's laptop but fails on the build server. The project installs whatever version of each dependency is newest at build time. Which SCM gap explains this best? Analysis

    A. The team should create more feature branches
    B. The build server needs a faster processor
    C. Git corrupted the source files during the push
    D. Dependency versions are not under configuration control; pin them, for example in a lock file, so each build uses identified versions

1383. A designer edits a draft design document several times before its formal review, without raising change requests. After the review it is baselined. Was change control bypassed? Analysis
    A. Yes, because every saved draft is automatically a baseline
    B. No, because design documents are never configuration items
    C. No: before baselining an item can change informally; formal change control applies once it is baselined
    D. Yes, because every edit to any document needs CCB approval

1384. Two developers on separate branches edit two different functions that are far apart in the same file. What happens when the second branch is merged? Analysis
    A. Git always reports a conflict when both branches change the same file
    B. Git refuses to merge files edited on both branches
    C. Git merges automatically, because the changed regions do not overlap
    D. The second developer's whole file overwrites the first one's

1385. Branch X renames calc() to compute() and updates every existing caller. Branch Y, created at the same time, adds a new file that calls calc(). Both merge into main without conflicts. What happens next? Analysis
    A. The merged code calls a function that no longer exists; Git cannot see this semantic conflict, so a build and test run after merging must catch it
    B. Git reports a conflict in the new file
    C. Git renames the new call to compute() automatically
    D. The code works, because Git keeps both function names

1386. An API key was pushed to a public repository yesterday. A developer deletes it in a new commit and pushes. What is the correct assessment? Analysis
    A. Running git revert on the commit that added it removes it from history
    B. Deleting it in a new commit fully removes it
    C. The key is still in history and may already be copied; revoke and rotate it first, then rewrite history if needed
    D. Making the repository private fixes the problem

1387. A library at 3.4.0 marks a public function as deprecated, but it still works exactly as before, and a replacement function is added. Following the SemVer spec, what is the next version? Analysis
    A. 3.4.0-deprecated
    B. 3.5.0
    C. 3.4.1
    D. 4.0.0

1388. A CCB approved a change to a shared date-format function after looking only at the screen that requested it. After release, three unrelated reports print wrong dates. Which step was weak? Analysis
    A. Configuration status accounting
    B. Version numbering
    C. Physical configuration audit
    D. Impact analysis, which should have found every item that uses the function before approval

1389. A developer has two unpushed commits on main, and teammates have pushed meanwhile. Each git pull adds a 'Merge branch main of ...' commit, cluttering history. What fits best? Analysis
    A. Use git pull --rebase, or set pull.rebase, so these private local commits replay on top of origin/main
    B. Force-push first so the remote matches the local branch
    C. Run git reset --hard origin/main before each pull
    D. Delete the clone and clone again before each pull

1390. Before release 5.0, an audit confirms that every file and version matches the release documentation. A month later, customers find that an export feature required by the approved SRS was never built. Which audit should have caught this? Analysis

A. A status accounting report
B. A financial audit of the project
C. A functional configuration audit, which checks the product against its requirements
D. A physical configuration audit

1391. You ran git reset --hard HEAD~1 and lost commit B, which was never pushed. What is the best way to recover it? Analysis

A. It cannot be recovered, because reset --hard deletes commits permanently
B. Find B in git reflog and reset or branch to it, for example git reset --hard HEAD@{1}
C. Run git revert HEAD
D. Run git pull to fetch it from the remote

1392. A team committed 2 GB of video files and later deleted them in a new commit. Fresh clones are still huge. Why, and what fixes it? Analysis

A. Older commits still hold the files; rewrite history (for example with git filter-repo) and keep large media in Git LFS or outside the repository
B. Adding the files to .gitignore now removes them from the clones
C. Running git rm --cached on them again removes them from history
D. Git has not run garbage collection yet; clones will shrink by themselves

Software Risk Management MCQs for NSCT

1393. According to Pressman, which two characteristics define a software risk? Understanding

A. Probability and frequency
B. Uncertainty (it may or may not happen) and loss (it causes harm if it happens)
C. Cost and schedule
D. Severity and priority

1394. How is risk exposure (RE) usually calculated? Understanding

A. Number of risks × average cost
B. Probability of the risk × cost (impact) if it occurs
C. Probability + impact
D. Impact ÷ probability

1395. What is proactive risk management? Understanding

A. Ignoring risks with low probability
B. Buying insurance for every project
C. Identifying and assessing risks before work starts and planning how to handle them
D. Fixing problems only after they occur

1396. Sommerville groups risks as project, product and business risks. A risk that the chosen database cannot meet the required response time is a: Understanding

A. Business risk
B. Project risk
C. Personal risk
D. Product risk

1397. Which software process model is explicitly risk-driven, with risk analysis in every cycle? Understanding

A. Code-and-fix
B. Spiral model
C. Waterfall model
D. V-model

1398. What is a risk register? Understanding

A. A document listing each identified risk with its probability, impact, owner, response and status
B. A list of bugs found in testing
C. The project's budget spreadsheet
D. A record of team members' working hours

1399. A company signs a fixed-price contract with a subcontractor for a risky module, so cost overruns fall on the subcontractor. Which risk response is this? Understanding

A. Acceptance
B. Transfer
C. Avoidance
D. Monitoring

1400. A team drops a planned feature because it depends on hardware that may not be ready. Which risk response is this? Understanding

A. Mitigation
B. Transfer
C. Avoidance
D. Acceptance

1401. What is a contingency plan? Understanding

A. A plan for the next project
B. A list of all risks without any actions
C. A plan to stop a risk from ever occurring
D. A planned set of actions to take if a risk actually occurs

1402. In Pressman's approach, what does RMMM stand for? Understanding

A. Requirements Management and Maintenance Model
B. Risk Matrix and Mitigation Method
C. Risk Measurement, Modelling and Maintenance
D. Risk Mitigation, Monitoring and Management

1403. In Pressman's risk table, impact is rated 1 to 4. What does an impact value of 1 mean? Understanding

A. Critical
B. Marginal
C. Catastrophic
D. Negligible

1404. Pressman divides risks into known, predictable and unpredictable. "Staff turnover" in an organisation that loses developers every year is an example of a: Understanding

A. Known risk
B. Predictable risk
C. Unpredictable risk
D. Business risk only

1405. During risk monitoring, what is a risk trigger (or indicator)? Understanding

A. The person who owns the risk
B. A warning sign showing that a risk is becoming more likely or is about to occur
C. The cost of the risk if it occurs
D. The final report written after the project

1406. How does qualitative risk analysis differ from quantitative risk analysis? Understanding

A. Qualitative ranks risks with categories such as high/medium/low; quantitative gives numerical estimates such as RE in money or days
B. Qualitative is only for business risks; quantitative only for technical risks
C. Qualitative is done after the project ends; quantitative before
D. There is no difference

1407. In Boehm's well-known list of top ten software risk items, which appears first? Understanding

A. Continuing stream of requirements changes
B. Real-time performance shortfalls
C. Gold plating
D. Personnel shortfalls

1408. In PMI's terminology, what is a secondary risk? Understanding

A. A new risk that arises directly from carrying out a risk response
B. A risk owned by a secondary stakeholder
C. A risk with low probability
D. The risk left over after a response has been applied

1409. There is a 30% chance that a key third-party API will be withdrawn. If it is, rework will cost Rs 2,000,000. What is the risk exposure? Application

A. Rs 600,000
B. Rs 6,000,000
C. Rs 2,000,000
D. Rs 1,400,000

1410. On a 5 × 5 matrix where score = likelihood × impact, a risk is rated likelihood 4 and impact 3. What is its score? Application

A. 43
B. 1
C. 12
D. 7

1411. There is a 25% chance that a vendor delivers late, which would cost 40 person-days of rework. What is the risk exposure in person-days? Application

A. 30 person-days
B. 10 person-days
C. 40 person-days
D. 1.6 person-days

1412. A project plans to reuse 60 components. There is a 70% chance that only 70% of them will be reusable; the rest must be built at 100 LOC each and Rs 14 per LOC. What is the risk exposure? Application

A. Rs 17,640
B. Rs 25,200
C. Rs 7,560
D. Rs 58,800

1413. Three risks have (probability, cost): (0.2, Rs 500,000), (0.5, Rs 200,000) and (0.1, Rs 1,500,000). If the contingency reserve equals total risk exposure, how big is it? Application

A. Rs 800,000
B. Rs 2,200,000
C. Rs 350,000
D. Rs 100,000

1414. Rank by risk exposure. R1: P 0.6, cost 100,000. R2: P 0.1, cost 900,000. R3: P 0.3, cost 250,000. R4: P 0.8, cost 50,000. Which order is correct, highest first? Application

A. R1, R2, R3, R4
B. R2, R3, R1, R4
C. R2, R1, R3, R4
D. R4, R1, R3, R2

1415. A risk has P = 0.4 and cost Rs 1,000,000. A Rs 60,000 prototype would cut P to 0.1. What is the risk reduction leverage (RRL)? Application

A. 5
B. 0.2
C. 6.67
D. 3

1416. A risk has RE Rs 300,000. Option M1 costs Rs 50,000 and lowers RE to Rs 150,000. Option M2 costs Rs 150,000 and lowers RE to Rs 90,000. Which option has the higher risk reduction leverage? Application

A. M2, because it lowers RE more
B. M1, with RRL 3 against M2's 1.4
C. Both have RRL 1
D. M2, with RRL 3 against M1's 1.4

1417. Using Sommerville's categories, how would you classify the risk "the only developer who understands the payment module may leave"? Application

A. Technology risk
B. Product risk
C. Project risk
D. Business risk

1418. Using Sommerville's categories, how would you classify the risk "a competitor launches a similar app three months before ours"? Application

A. Product risk
B. Business risk
C. Project risk
D. Technical risk

1419. To reduce the impact of a key developer leaving, the team pairs two others with her and documents the module as they go. Which strategy is this? Application

A. Mitigation (minimisation)
B. Acceptance
C. Avoidance
D. Transfer

1420. Build option: 60% chance of costing 1.0 million, 40% chance of 1.8 million. Buy option: 90% chance of 1.2 million, 10% chance of 2.0 million. Which has the lower expected cost? Application

A. Build, because its best case (1.0 million) is cheaper
B. They are equal at 1.30 million
C. Build, with an expected 1.32 million against buy's 1.28 million
D. Buy, with an expected 1.28 million against build's 1.32 million

1421. Risk R has P = 0.5 and impact Rs 800,000. Mitigation costs Rs 150,000 and lowers P to 0.1. Comparing total expected cost (mitigation cost + RE), what should you do? Application

A. Mitigate: expected cost falls from Rs 800,000 to Rs 80,000
B. Do not mitigate: expected cost rises to Rs 950,000
C. Do not mitigate: Rs 150,000 is too much to spend
D. Mitigate: expected cost falls from Rs 400,000 to Rs 230,000

1422. An integration risk has three outcomes: no delay (P 0.5), a 1-month delay (P 0.3) or a 3-month delay (P 0.2). Each month of delay costs Rs 400,000. What is the risk exposure? Application

A. Rs 1,600,000
B. Rs 200,000
C. Rs 360,000
D. Rs 533,333

1423. A team never discusses risks at planning. When the cloud provider changes its pricing mid-project, they scramble to rewrite the deployment. What is the best diagnosis? Analysis

A. They were unlucky; pricing changes cannot be foreseen
B. Their mistake was choosing the cloud at all
C. They should have transferred all risks to the client
D. They practise reactive risk management; proactive identification would have listed vendor dependency and planned a response early

1424. Risk A: 1% chance of losing all customer data, costing Rs 5,000,000. Risk B: 50% chance of a minor UI delay, costing Rs 100,000. Both have RE Rs 50,000. Which statement is most sensible? Analysis

A. Treat them identically because their RE is equal
B. Focus only on B because it is more likely
C. Ignore A because its probability is only 1%
D. Equal RE does not mean equal treatment: A's catastrophic impact justifies backups and a contingency plan even though it is unlikely

1425. A project filled in a detailed risk register at kick-off and never opened it again. Six months later several listed risks have occurred without warning. Which risk management activity was missing? Analysis

A. Risk monitoring: regularly re-assessing risks and watching their triggers
B. Risk identification
C. Risk transfer
D. Risk analysis

1426. A risk has P = 0.2 and cost Rs 500,000. A mitigation costing Rs 80,000 would halve the probability. Is the mitigation justified on expected cost alone? Analysis

A. No: RRL = (100,000 − 50,000) / 80,000 = 0.625, below 1, so it costs more than it saves
B. No: RRL is negative
C. Yes: any reduction in probability is worth paying for
D. Yes: RRL = 80,000 / 50,000 = 1.6

1427. On a 1–5 likelihood × impact scale: A (L5, I2), B (L2, I5), C (L3, I4), D (L4, I2). Which statement is correct? Analysis

A. C ranks highest at 12; A and B tie at 10, so the tie must be broken by another rule such as impact severity
B. All four are equal because each has one rating of 4 or more
C. A ranks highest because it is the most likely
D. B ranks highest because it has the highest impact

1428. Risk X: 5% chance of a regulatory fine of Rs 10,000,000. Risk Y: 50% chance of Rs 800,000 rework. A manager wants to handle Y first because it is "ten times more likely". What does risk exposure say? Analysis

A. They are equal, so the order does not matter
B. Neither, because both are below Rs 1,000,000
C. X has the higher RE (Rs 500,000 against Rs 400,000), so on expected loss X should come first
D. Y first, because likelihood matters most

1429. A start-up must build a product with vague requirements using a technology no one on the team has used. Which process choice best addresses the risk? Analysis

A. Waterfall, freezing requirements first to reduce uncertainty
B. V-model, because it emphasises testing
C. Code-and-fix, because planning is wasted when requirements are vague
D. A spiral or iterative approach that starts with prototypes of the riskiest parts and re-assesses risks each cycle

1430. For the risk "high staff turnover", which indicator would be most useful to monitor? Analysis

A. The number of open bugs
B. The project's cloud hosting bill
C. Team morale and signs that staff are job hunting, such as frequent leave requests and lower engagement
D. The number of lines of code written per week

1431. To reduce schedule risk, a team outsources the reporting module to a vendor. Which new risk should now be added to the register? Analysis

A. Database performance risk
B. The risk that the internal team works too fast
C. No new risk, because outsourcing removes the risk entirely
D. Vendor risk: late or poor-quality delivery and dependence on the vendor (a secondary risk)

1432. Two independent risks could each delay the release, with probabilities 0.3 and 0.3. What is the probability that at least one occurs? Analysis

A. 0.6
B. 0.09
C. 0.51
D. 0.3

1433. What is the difference between a risk and an issue? Understanding

A. A risk affects the schedule; an issue affects the budget
B. A risk is an uncertain future event; an issue is a problem that has already happened
C. A risk is found by testers; an issue is found by managers
D. A risk has a high probability; an issue has a low probability

1434. In PMI terminology, what is residual risk? Understanding

A. A new risk created by carrying out a risk response
B. A risk that has been transferred to an insurer
C. A risk that was missed during identification
D. The risk that remains after a planned response has been carried out

1435. In Sommerville's risk management process, what is the correct order of the four stages? Understanding

A. Identification, analysis, planning, monitoring
B. Analysis, identification, monitoring, planning
C. Planning, identification, monitoring, analysis
D. Identification, planning, analysis, monitoring

1436. Which activity belongs to risk identification rather than risk analysis? Understanding

A. Rating each risk's probability as low, moderate or high
B. Calculating risk exposure for each listed risk
C. Sorting the risk table and drawing a cut-off line
D. Running a brainstorming session that uses a checklist of common project risks

1437. What is a risk owner? Understanding

A. The stakeholder who will pay if the risk occurs
B. The developer whose code caused the risk
C. The external auditor who signs off the risk register
D. The person responsible for monitoring a specific risk and carrying out its response

1438. A team notes that a font it uses might change licence terms. The cost would be tiny, so it decides to do nothing now and deal with it if it happens. Which response is this? Understanding

A. Avoidance
B. Transfer
C. Acceptance
D. Mitigation

1439. Sommerville groups risks by type. "The organisation is restructured so that different managers become responsible for the project" is which type? Understanding

A. People risk
B. Organisational risk
C. Requirements risk
D. Estimation risk

1440. In Pressman's classification, what do technical risks threaten? Understanding

A. The quality and timeliness of the software being built
B. The viability of the product in the market
C. The project budget and staffing plan only
D. The legal ownership of the source code

1441. In each loop of Boehm's spiral model, what follows determining the objectives, alternatives and constraints? Understanding

A. Writing the full requirements specification
B. Delivering the final release to the customer
C. Evaluating the alternatives and identifying and resolving risks, often by prototyping
D. Running acceptance tests on the integrated system

1442. Under PMI's PMBOK, an uncertain event that would benefit the project if it occurred is called: Understanding

A. A secondary risk
B. A residual risk
C. An opportunity (a positive risk)
D. A contingency

1443. Pressman lists several business risks. "Building an excellent product that no one really wants" is which one? Understanding

A. Strategic risk
B. Market risk
C. Sales risk
D. Budget risk

1444. Pressman's business risks include "losing the support of senior management because of a change in focus or people". What is it called? Understanding

A. Management risk
B. Market risk
C. Strategic risk
D. Project risk

1445. Pressman names four risk components. Which one is the degree of uncertainty that the software will be easy to correct, adapt and enhance? Understanding

A. Support risk
B. Performance risk
C. Cost risk
D. Schedule risk

1446. During risk refinement, Pressman suggests writing a risk in CTC format. What does CTC stand for? Understanding

A. Condition, transition, consequence
B. Cause, threat, cost
C. Category, trigger, contingency
D. Control, test, correct

1447. In Pressman's risk table, what is the purpose of the cut-off line? Understanding

A. It separates technical risks from business risks
B. Risks below the line are deleted from the register
C. It marks the date after which no new risks may be added
D. Risks above the line, after sorting by probability and impact, receive further management attention

1448. Pressman distinguishes generic risks from product-specific risks. What is a product-specific risk? Understanding

A. One that threatens every software project, such as staff turnover
B. One that affects only the end users after release
C. One that can be identified only by people who understand the technology, people and environment of this particular product
D. One that has already occurred on a previous project

1449. Sommerville uses five probability bands. A risk rated "high" has a probability of: Understanding

A. 25 to 50%
B. Over 75%
C. 50 to 75%
D. 10 to 25%

1450. Sommerville rates the effect of a risk on a four-level scale. Which list gives his levels from worst to least? Understanding

A. Catastrophic, serious, tolerable, insignificant
B. Catastrophic, critical, marginal, negligible
C. Very high, high, medium, low
D. Severe, major, moderate, minor

1451. In PMI practice, how does a contingency reserve differ from a management reserve? Understanding

A. Contingency is held by the client; management reserve is held by the vendor
B. Contingency covers schedule only; management reserve covers cost only
C. They are two names for the same budget line
D. Contingency covers identified risks (known-unknowns); management reserve covers unidentified risks (unknown-unknowns)

1452. What is risk-based testing? Understanding

A. Testing only the features listed in the risk register
B. Giving more test effort to the features whose failure is most likely or most damaging
C. Testing risk management tools before using them
D. Skipping tests for any feature rated low risk by developers

1453. In Failure Mode and Effects Analysis (FMEA), how is the risk priority number (RPN) calculated? Understanding

A. Probability × cost
B. Severity × occurrence × detection
C. Severity + occurrence + detection
D. Likelihood × impact ÷ detection

1454. Pressman adopts SEI's principles of risk management. Which of these is one of them? Understanding

A. Emphasise a continuous process
B. Freeze the risk list after the kick-off meeting
C. Let only the project manager discuss risks
D. Address risks only once they become issues

1455. Which statement about risk transfer is accurate? Understanding

A. It reduces the probability of the risk to zero
B. It shifts the financial consequence and the response to a third party, usually for a fee, but rarely removes the risk itself
C. It is free, because the other party absorbs the whole cost
D. It is the same as avoidance, because the project no longer faces the risk

1456. In quantitative risk analysis, what does a Monte Carlo simulation of a project schedule produce? Understanding

A. A single most likely completion date
B. A ranked list of risks by their probability
C. The critical path assuming every task takes its optimistic time
D. A probability distribution of completion dates, from many runs that sample each task's uncertain duration

1457. There is a 15% chance that a cloud migration fails and must be redone, costing Rs 800,000. What is the risk exposure? Application

A. Rs 800,000
B. Rs 120,000
C. Rs 680,000
D. Rs 53,333

1458. A team scores risks as likelihood × impact on 1–5 scales: 15 or more is red, 8 to 14 amber, below 8 green. A risk has likelihood 3 and impact 3. What colour is it? Application

A. Green, with a score of 6
B. Red, with a score of 15
C. Amber, with a score of 9
D. Amber, with a score of 6

1459. Using Sommerville's risk types, how would you classify "the code produced by our code generation tool is inefficient"? Application

A. Technology risk
B. Estimation risk
C. Tools risk
D. People risk

1460. Using Sommerville's risk types, how would you classify "we cannot recruit staff with the skills this project needs"? Application

A. Organisational risk
B. Requirements risk
C. People risk
D. Tools risk

1461. A government grant for local cloud hosting may become available. The team assigns someone to prepare the application early so that the project is certain to get it. Which opportunity response is this? Application

A. Share
B. Enhance
C. Exploit
D. Transfer

1462. There is a 40% chance that test devices arrive late, which would delay the project by 15 days. What is the risk exposure in days? Application

A. 15 days
B. 6 days
C. 9 days
D. 37.5 days

1463. For the risk "requirements will change heavily", which indicator would Sommerville suggest monitoring? Application

A. A rising number of requirement change requests and customer complaints
B. Falling staff morale
C. Late delivery of hardware
D. Team reluctance to use the CASE tools

1464. A project's identified risks have a total exposure of Rs 450,000. The project budget is Rs 9,000,000. What percentage of the budget is at risk? Application

A. 20%
B. 0.5%
C. 5%
D. 50%

1465. To assess schedule risk, a task is estimated with PERT: optimistic 4 weeks, most likely 6 weeks, pessimistic 14 weeks. What are the expected duration and standard deviation? Application

A. 6 weeks, with a standard deviation of about 1.67 weeks
B. 7 weeks, with a standard deviation of about 1.67 weeks
C. 8 weeks, with a standard deviation of 10 weeks
D. 7 weeks, with a standard deviation of 10 weeks

1466. In an FMEA, a failure mode is rated severity 8, occurrence 4 and detection 6. What is its risk priority number? Application

A. 18
B. 32
C. 48
D. 192

1467. FMEA ratings: failure mode A has severity 9, occurrence 2, detection 2. Failure mode B has severity 5, occurrence 6, detection 5. Which has the higher RPN? Application

A. A, with an RPN of 150 against B's 36
B. B, with an RPN of 150 against A's 36
C. A, because its severity is higher
D. They are equal, because both total 13 or more

1468. A risk has P = 0.5 and cost Rs 600,000. A Rs 40,000 spike would cut P to 0.2. What is the risk reduction leverage? Application

A. 4.5
B. 7.5
C. 3
D. 0.22

1469. Four risks have (probability, cost): (0.1, Rs 2,000,000), (0.25, Rs 400,000), (0.6, Rs 150,000) and (0.05, Rs 6,000,000). What contingency reserve equals their total exposure? Application

A. Rs 8,550,000
B. Rs 1,000,000
C. Rs 590,000
D. Rs 690,000

1470. Pressman's impact scale runs 1 (catastrophic) to 4 (negligible). The cut-off rule is: probability 40% or more and impact 1 or 2. R1: 60%, impact 3. R2: 50%, impact 2. R3: 30%, impact 1. R4: 70%, impact 1. Which risks fall above the line? Application

A. R1 and R4
B. R3 and R4
C. R2 and R4
D. R1, R2 and R4

1471. Sommerville's table classifies "specification delays: specifications of essential interfaces are not available on schedule". How is it classified? Application

A. As a business risk only
B. As a product risk only
C. As a project risk only
D. As both a project risk and a product risk

1472. Three independent risks could delay a release: (P 0.3, 10 days), (P 0.5, 4 days) and (P 0.1, 30 days). What schedule buffer equals their total expected delay? Application

A. 8 days
B. 44 days
C. 14.7 days
D. 3 days

1473. Two small software firms form a joint venture to bid for a large government contract that neither could win alone. Which opportunity response is this? Application

A. Share
B. Exploit
C. Accept
D. Mitigate

1474. A risk has P = 0.3 and loss Rs 2,000,000. Option X costs Rs 250,000 and halves P. Option Y costs Rs 450,000 and cuts the loss to Rs 500,000. Comparing total expected cost (response cost + remaining RE), which choice is best? Application

A. Option X, at Rs 550,000
B. Option Y, at Rs 600,000
C. Doing nothing, at Rs 600,000
D. Option Y, at Rs 450,000

1475. A risk has P = 0.4 and cost Rs 1,200,000. A Rs 140,000 redesign lowers P to 0.25 and the cost to Rs 800,000. What is the risk reduction leverage? Application

A. 1.29
B. 2.0
C. 3.43
D. 0.5

1476. An FMEA failure mode is rated severity 8, occurrence 5, detection 7. After adding input checks and monitoring, occurrence falls to 3 and detection to 2. What is the new RPN, and by roughly what percentage has it fallen? Application

A. 48, a fall of about 83%
B. 48, a fall of about 17%
C. 120, a fall of about 57%
D. 13, a fall of about 95%

1477. A risky integration is estimated with PERT: optimistic 20 days, most likely 26, pessimistic 44. Assuming a roughly normal total, what deadline gives about 84% confidence? Application

A. 32 days (mean 28, standard deviation 4)
B. 30 days (mean 26, standard deviation 4)
C. 36 days (mean 28, standard deviation 8)
D. 44 days (the pessimistic estimate)

1478. A risk register contains the single entry "The project might be late." What is the main weakness of this entry? Analysis

A. It should be deleted, because lateness is an issue, not a risk
B. It needs a higher probability rating to be taken seriously
C. It belongs in the business risk category instead
D. It is too vague to act on; it should name a specific cause and consequence, for example in CTC form

1479. Every risk in a team's register has "The team" as its owner. When several trigger conditions appear, nobody acts. What is the most likely cause? Analysis

A. The risks were rated with the wrong probability scale
B. The team used qualitative instead of quantitative analysis
C. No individual is accountable for each risk, so monitoring and responses fall between people
D. The register should have been kept by the client

1480. A company insures its servers against data loss and then stops making backups, reasoning that the insurer will pay. Which assessment is correct? Analysis

A. The decision is sound, because the risk has been fully transferred
B. Insurance transfers part of the financial loss only; lost data still harms customers and the schedule, so backups are still needed
C. The decision is sound, because insurance is a form of avoidance
D. Insurance is pointless; the company should only keep backups

1481. A team works in two-week sprints and demonstrates working software to the client every sprint. How does this affect the risk "we have misunderstood the requirements"? Analysis

A. It avoids the risk completely, so it can be removed from the register
B. It transfers the risk to the client, who now owns the requirements
C. It mitigates the risk by limiting how much work can be built on a misunderstanding before feedback catches it
D. It has no effect, because sprints only affect the schedule

1482. Over three months, a project's schedule performance index falls from 1.0 to 0.92 to 0.85, and the open defect backlog keeps growing. Which of Sommerville's risk types do these indicators point to most directly? Analysis

A. Tools risk
B. Organisational risk
C. Technology risk
D. Estimation risk

1483. A team "accepts" the risk that its only build server fails. It sets no money or time aside and writes no plan. When the server fails, two weeks are lost improvising. What went wrong? Analysis

A. Acceptance is never a valid response for technical risks
B. They should have classified the risk as a business risk
C. The risk should have been ignored, because hardware failures are unpredictable
D. They used passive acceptance; for a risk with this impact, active acceptance with a contingency plan or reserve was needed

1484. A risk has P = 0.6 and cost Rs 1,000,000. M1 costs Rs 50,000 and cuts P to 0.4. M2 costs Rs 300,000 and cuts P to 0.05. Which statement is correct? Analysis

A. M1 is better on both leverage and total expected cost
B. M2 is better on both leverage and total expected cost
C. Both options leave the same total expected cost
D. M1 has the higher leverage (RRL 4 against about 1.83), but M2 leaves the lower total expected cost (Rs 350,000 against Rs 450,000)

1485. Five independent risks each have P = 0.2 and cost Rs 500,000. A manager sets the contingency reserve at Rs 2,500,000. What is wrong with this reasoning? Analysis

A. Nothing; a reserve must always cover the worst case
B. The reserve is too small, because the risks could repeat
C. It assumes every risk occurs; the expected loss is Rs 500,000, and all five occurring has a probability of only 0.00032
D. The reserve should be Rs 100,000, the exposure of one risk

1486. A team estimates P(lead developer leaves) = 0.3 and P(payment module is late) = 0.4, and says the chance of both is 0.12. But if the lead leaves, the module is late 90% of the time. What is the real chance of both, and what was the mistake? Analysis

A. 0.27; the team wrongly treated two dependent risks as independent
B. 0.12; the calculation was correct
C. 0.36; they should have multiplied 0.4 by 0.9
D. 0.70; they should have added the probabilities

1487. R1: P 0.5, delay of 20 days. R2: P 0.2, extra cost Rs 900,000. Each day of delay costs Rs 40,000. Which risk has the higher exposure? Analysis

A. R2, because Rs 900,000 is larger than 20 days
B. R1, at Rs 400,000 against R2's Rs 180,000
C. They cannot be compared, because the units differ
D. They are equal, at Rs 180,000 each

1488. Developers keep adding animations and extra settings that no one asked for, and the schedule slips. Which of Boehm's top-ten risk items is this, and which technique fits? Analysis

A. Continuing stream of requirements changes; a change control board
B. Gold-plating; requirements scrubbing and cost-benefit analysis of each addition
C. Personnel shortfalls; hiring more developers
D. Straining computer science capabilities; technical analysis

1489. A risk has RE Rs 200,000. A mitigation costing Rs 120,000 would lower RE to Rs 120,000. Which conclusion is sound? Analysis

A. RRL is 1.67, so the mitigation clearly pays for itself
B. RRL is about 0.67, so it costs more than it saves on expected value; only non-monetary reasons such as safety or regulation could justify it
C. RRL is 0.6, so the mitigation should be done twice
D. RRL cannot be computed without the probability

1490. To cover missing mobile skills, a team outsources its app to a vendor. Afterwards: (i) there is still a smaller chance the app is of poor quality; (ii) the vendor might go bankrupt mid-contract. How should these be classified? Analysis

A. (i) is secondary risk and (ii) is residual risk
B. Both are residual risks
C. Both are secondary risks
D. (i) is residual risk and (ii) is secondary risk

1491. Total risk exposure in a register, reviewed every sprint, reads Rs 900,000, Rs 850,000, Rs 870,000, then Rs 950,000. What does this trend most likely indicate? Analysis

A. Mitigation is working, because the figure is still under Rs 1,000,000
B. Nothing, because exposure normally rises until release
C. The register is wrong, because exposure can never rise once mitigation starts
D. New or growing risks are outpacing mitigation, so the risks and responses need review and possibly escalation

1492. In week one of a new kind of project with no historical data, a manager insists every risk gets an exact probability such as 0.37 and a cost to the rupee. Which critique is most accurate? Analysis

A. Exact numbers are always required, because RE needs them
B. Risks should not be analysed until the project has historical data
C. This is false precision; a qualitative ranking suits early analysis, with quantitative estimates kept for the top risks as data becomes available
D. Qualitative analysis is unreliable and should never be used

Software Security Engineering MCQs for NSCT

1493. In the STRIDE threat model, which security property does a Tampering threat violate? Understanding

A. Confidentiality
B. Integrity
C. Availability
D. Authentication

1494. What does "shifting security left" mean in a secure SDLC? Understanding

A. Moving the security team to a different department
B. Running penetration tests after deployment only
C. Writing security code in the left-hand panel of the IDE
D. Building security into the early phases, such as requirements and design, instead of testing for it only before release

1495. What is static application security testing (SAST)? Understanding

A. Attacking the running application over HTTP
B. Analysing source code, bytecode or binaries for vulnerabilities without running the program
C. Checking the physical security of servers
D. Load testing the application

1496. What is dynamic application security testing (DAST)? Understanding

A. Reading the source code for insecure functions
B. Scanning dependencies for known vulnerabilities
C. Testing a running application from the outside by sending crafted requests, without access to its source code
D. Reviewing the design documents

1497. A reporting service only needs to read sales data, so it is given a read-only database account. Which principle is applied? Understanding

A. Least privilege
B. Open design
C. Defence in depth
D. Security through obscurity

1498. What is the most effective primary defence against SQL injection? Understanding

A. Hiding database error messages
B. Parameterised queries (prepared statements)
C. Encrypting the database files
D. Using a longer database password

1499. How should an application store user passwords? Understanding

A. As plain MD5 hashes
B. In plain text in a protected table
C. As salted hashes from a slow password-hashing function such as bcrypt, scrypt, Argon2 or PBKDF2
D. Encrypted with AES, so they can be decrypted when needed

1500. A system uses a firewall, input validation, parameterised queries, least-privilege database accounts and audit logs together. Which principle does this show? Understanding

A. Single point of control
B. Economy of mechanism
C. Security through obscurity
D. Defence in depth

1501. In the OWASP Top 10:2021, which category is ranked first? Understanding

A. Cross-Site Scripting
B. Broken Access Control
C. Security Misconfiguration
D. Injection

1502. Which control best counters a Repudiation threat in STRIDE? Understanding

A. Input validation
B. Encryption of data at rest
C. Secure, tamper-evident audit logs and digital signatures
D. Rate limiting

1503. In threat modelling, why are trust boundaries drawn on the data flow diagram (DFD)? Understanding

A. They show where data crosses between zones of different trust, which is where threats are most likely and checks are needed
B. They show which developer owns each module
C. They mark the parts of the system that need no security
D. They are used only for performance analysis

1504. Saltzer and Schroeder's "fail-safe defaults" principle means: Understanding

A. Access is denied unless it is explicitly granted
B. Default passwords are allowed if they are long
C. Errors are shown in full to help users
D. The system restarts automatically after a crash

1505. What does the principle of complete mediation require? Understanding

A. Every access to every resource is checked for authorisation, not just the first one
B. All traffic passes through one firewall
C. All code is reviewed by two people
D. Security decisions are made by a mediator committee

1506. What is a misuse case (or abuse case) in security requirements engineering? Understanding

A. A test case that failed
B. A bug report filed by a user
C. A scenario describing how an attacker could use the system to cause harm, used to derive security requirements
D. A use case that the customer rejected

1507. What is fuzz testing (fuzzing)? Understanding

A. Reviewing code by eye for security flaws
B. Checking that passwords meet complexity rules
C. Feeding large volumes of malformed or random input to a program to find crashes and unexpected behaviour
D. Testing the user interface for usability

1508. Which tool category finds known vulnerabilities in the third-party libraries an application depends on? Understanding

A. Software composition analysis (SCA)
B. Fuzzing
C. Code coverage analysis
D. DAST

1509. An attacker edits a hidden form field to change a product's price from 5,000 to 50 before submitting the order. Which STRIDE category is this? Application

A. Repudiation
B. Information disclosure
C. Spoofing
D. Tampering

1510. A customer claims they never placed an order, and the system has no logs to prove otherwise. Which STRIDE threat has succeeded? Application

A. Elevation of privilege
B. Tampering
C. Repudiation
D. Denial of service

1511. Three threats are scored as likelihood × impact on 1–5 scales. T1: L4, I2. T2: L2, I5. T3: L3, I3. Which should be handled first? Application

A. T1 (score 8)
B. T3 (score 9)
C. T1, because it is the most likely
D. T2 (score 10)

1512. What does this Python program print? Application

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE users (name TEXT, pw TEXT)")
db.executemany("INSERT INTO users VALUES (?, ?)",
               [("ali", "a1"), ("sara", "s2"), ("omar", "o3")])
name = "x' OR '1'='1"
q = f"SELECT name FROM users WHERE name = '{name}'"
print(len(db.execute(q).fetchall()))
```

A. 1
B. 0
C. An error
D. 3

1513. The query is now parameterised, with the same malicious input. What does the program print? Application

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE users (name TEXT, pw TEXT)")
db.executemany("INSERT INTO users VALUES (?, ?)",
               [("ali", "a1"), ("sara", "s2"), ("omar", "o3")])
name = "x' OR '1'='1"
q = "SELECT name FROM users WHERE name = ?"
print(len(db.execute(q, (name,)).fetchall()))
```

A. 0
B. An error
C. 1
D. 3

1514. A blog shows user comments in its HTML. What does this escaping step print? Application

```
import html
comment = '<script>alert("hi")</script>'
print(html.escape(comment))
```

    A. <script>alert("hi")</script>
    B. <script>alert("hi")</script>
    C. <script>alert("hi")</script>
    D. alert("hi")

1515. A logged-in user changes the URL from /invoice?id=1043 to /invoice?id=1044 and sees another customer's invoice. What is the vulnerability and its fix? Application
    A. SQL injection; use parameterised queries
    B. Weak passwords; enforce password complexity
    C. Broken access control (an insecure direct object reference); check on the server that the invoice belongs to the current user
    D. XSS; escape the output

1516. A banking site lets a logged-in user transfer money with a simple POST form. A malicious page can make the victim's browser submit it. Which defence fits best? Application
    A. A longer session timeout
    B. An unpredictable anti-CSRF token in each form, checked by the server, plus SameSite cookies
    C. Escaping the output of the form
    D. HTTPS only

1517. An attacker gets a victim to log in using a session ID the attacker already knows, then reuses that ID. Which fix addresses this session fixation? Application
    A. Store the session ID in a hidden form field
    B. Issue a new session ID immediately after successful login
    C. Disable HTTPS for the login page
    D. Use longer passwords

1518. A team rates a threat with DREAD: Damage 8, Reproducibility 6, Exploitability 7, Affected users 9, Discoverability 5. Using the average, what is its score? Application
    A. 8
    B. 7
    C. 5
    D. 35

1519. A production error page shows a full stack trace with the framework version and database table names. Which threat is this, and what is the fix? Application
    A. Information disclosure; show a generic error to users and log the details on the server
    B. Spoofing; add multi-factor authentication
    C. Denial of service; add more servers
    D. Tampering; add a checksum to the page

1520. A file-download feature builds the path like this. What path does it resolve to? Application

```
import posixpath
base = "/srv/app/files"
user_input = "../../../etc/passwd"
print(posixpath.normpath(posixpath.join(base, user_input)))
```

A. /etc/passwd
B. /srv/app/files/etc/passwd
C. /srv/etc/passwd
D. /srv/app/files/../../../etc/passwd (unchanged)

1521. Using a simple rating scheme, factors are scored 0–9 and averaged: below 3 is Low, 3 to below 6 Medium, 6 and above High. Likelihood factors are 7, 6, 8, 7 and impact factors are 4, 5, 3, 4. What are the likelihood and impact levels? Application

A. Likelihood High (7), impact Medium (4)
B. Likelihood High (7), impact High (4)
C. Likelihood Medium (7), impact Low (4)
D. Likelihood Medium (4), impact High (7)

1522. A login module stores hashlib.md5(password).hexdigest() for each user. Which change best fixes password storage? Application

A. Switch to a slow, salted password hash such as Argon2, bcrypt or PBKDF2, and rehash each password at the user's next login
B. Encrypt the MD5 hashes with AES
C. Switch to SHA-256 without a salt
D. Hash the password twice with MD5

1523. A team does no security work until a penetration test a week before launch. The test finds that the design sends account numbers to the browser and trusts them on return. What is the main process lesson? Analysis

A. Penetration tests should be skipped because they cause delays
B. Only the coding standard was at fault
C. The pen tester should have fixed the design
D. Security should start in requirements and design (threat modelling), because design flaws found at the end are the most expensive to fix

1524. A comment filter removes the text <script>. Attackers then post <img src=x onerror=alert(1)> and their code runs. What is the underlying mistake? Analysis

A. The filter should also remove the word alert
B. The page should use HTTPS
C. Relying on a deny-list instead of context-aware output encoding (plus allow-list validation)
D. Comments should be limited in length

1525. An app validates every form field with JavaScript in the browser and does no checks on the server. What is the main risk? Analysis

A. Nothing, because users cannot edit JavaScript
B. Validation in JavaScript is always more secure than on the server
C. Only performance suffers
D. Attackers can bypass the browser and send any request straight to the server, so all validation must also happen server-side

1526. On the same web app, SAST reports 40 findings and DAST reports 5. Which conclusion is sound? Analysis

A. DAST is better because it reports fewer false positives, so SAST can be dropped
B. They are complementary: SAST sees all code but can flag false positives, while DAST confirms runtime issues but misses paths it never reaches
C. The tools contradict each other, so both are unreliable
D. SAST is better because it finds more, so DAST can be dropped

1527. Under Microsoft's STRIDE-per-element approach, which threat categories apply to a data flow on a DFD? Analysis

A. All six STRIDE categories
B. Elevation of privilege only
C. Spoofing and Repudiation only
D. Tampering, Information disclosure and Denial of service

1528. In an admin panel, the Delete User button is hidden unless the user is an admin. The /api/users/delete endpoint does not check the caller's role. What is the flaw? Analysis

A. No flaw, because ordinary users cannot see the button
B. A weak password policy
C. Broken access control: hiding UI is not authorisation; the server must check the role on every request
D. Cross-site scripting in the admin panel

1529. A database dump shows that users ali and sara have the same unsalted SHA-256 password hash. What can an attacker infer, and what would have prevented it? Analysis

A. The database is corrupted
B. They share a password, and cracking one cracks both; a unique salt per user would make their hashes differ
C. Nothing, because SHA-256 cannot be reversed
D. Their accounts are linked by the same email

1530. A web app connects to its database as the all-powerful admin account. A SQL injection flaw is found. Apart from fixing the query, which change most limits damage from the next flaw? Analysis

A. Turn off database logging
B. Change the admin password
C. Move the database to a bigger server
D. Use a least-privilege account with only the permissions the app needs, such as no DROP and no access to other databases

1531. A security audit finds that a transitive dependency (a library used by a library the app uses) has a published critical vulnerability. Which secure SDLC control would catch this automatically in future? Analysis

A. A penetration test once a year
B. More code review of the app's own code
C. Static analysis of the app's source files only
D. Software composition analysis in the CI pipeline, failing builds on known critical vulnerabilities

1532. Threats are scored likelihood × impact (1–5 each). T1: L5, I4. T2: L2, I5. T3: L4, I4. T4: L3, I2. What is the correct order, highest first? Analysis

A. T3, T1, T2, T4
B. T1, T3, T2, T4
C. T1, T2, T3, T4
D. T2, T1, T3, T4

1533. In STRIDE, which threat category violates availability? Understanding

A. Denial of service
B. Information disclosure
C. Repudiation
D. Spoofing

1534. A normal user exploits a bug and gains administrator rights. Which STRIDE category is this? Understanding

A. Spoofing
B. Elevation of privilege
C. Tampering
D. Repudiation

1535. What is a system's attack surface? Understanding

A. The total set of entry points, such as inputs, APIs, ports and files, through which an attacker could interact with it
B. The number of vulnerabilities found by the last penetration test
C. The area of the network protected by the firewall
D. The list of users who have administrator access

1536. How does CWE differ from CVE? Understanding

A. CWE scores severity; CVE describes the fix
B. CWE covers hardware; CVE covers software
C. CWE catalogues general types of weakness; CVE identifies specific vulnerabilities in specific products
D. They are two names for the same database

1537. What is DevSecOps? Understanding

A. Building security practices and automated security checks into the DevOps pipeline, as a shared responsibility of the whole team
B. A separate security team that reviews code once a year
C. A firewall product designed for DevOps teams
D. Running penetration tests only in production

1538. Saltzer and Schroeder's "open design" principle says that: Understanding

A. Source code must always be published as open source
B. All users should be able to see the design documents
C. A system's security should not depend on keeping its design secret; it should rest on protected keys or passwords
D. Systems should be designed in open meetings with the customer

1539. In a white-box penetration test, the testers: Understanding

A. Have no information beyond the public URL
B. Have only a normal user account and nothing else
C. Test only the user interface, never the server
D. Have full knowledge of the system, such as source code and architecture

1540. In Microsoft's Security Development Lifecycle (SDL), in which phase is threat modelling mainly done? Understanding

A. Release
B. Design
C. Verification
D. Response

1541. What is allow-list (whitelist) input validation? Understanding

A. Rejecting input that contains known attack strings
B. Allowing all input from logged-in users
C. Escaping input just before it is displayed
D. Accepting only input that matches a defined pattern of known-good values and rejecting everything else

1542. Under CVSS v3.x, how is a vulnerability with a base score of 7.5 rated? Understanding

A. High
B. Medium
C. Critical
D. Low

1543. Saltzer and Schroeder's "economy of mechanism" principle means: Understanding

A. Keep the security design as small and simple as possible so it can be checked thoroughly
B. Use the cheapest security product that meets the requirement
C. Reuse one security mechanism for every user and resource
D. Spend the security budget only on the most likely threats

1544. A payment above Rs 1,000,000 needs approval from two different managers before release. Which principle does this apply? Understanding

A. Least privilege
B. Separation of privilege
C. Economy of mechanism
D. Fail-safe defaults

1545. What does Saltzer and Schroeder's "least common mechanism" principle advise? Understanding

A. Use the most common, widely tested security mechanisms
B. Give every user the least privilege needed
C. Use as few security mechanisms as possible overall
D. Minimise mechanisms shared by several users, because each shared one is a path for information to leak between them

1546. Which OWASP Top 10:2021 category, new in that edition, focuses on flaws in design rather than in implementation? Understanding

A. Insecure Design
B. Security Misconfiguration
C. Injection
D. Cryptographic Failures

1547. A developer commits an API key to a Git repository, then deletes it in the next commit. Why is that not enough? Understanding

A. Git encrypts deleted lines, so the key is safe
B. The second commit overwrites the first on every clone
C. The key only becomes exposed if the repository is public
D. The key remains in the repository's history, so it must be revoked and rotated

1548. What is OWASP SAMM (Software Assurance Maturity Model)? Understanding

A. A static analysis tool for Java code
B. A framework for assessing and improving an organisation's software security practices across business functions
C. A scoring system for individual vulnerabilities
D. A catalogue of web attack payloads

1549. What does a stack canary do? Understanding

A. It prevents code on the stack from executing
B. It randomises where the stack is placed in memory
C. It places a secret value before the return address and aborts the program if a buffer overflow has changed it
D. It checks array indexes at compile time

1550. Under STRIDE-per-element, which threat categories apply to an external entity, such as a user or third-party system, on a DFD? Understanding

A. Tampering, Information disclosure and Denial of service
B. All six STRIDE categories
C. Elevation of privilege only
D. Spoofing and Repudiation

1551. What is a time-of-check to time-of-use (TOCTOU) flaw? Understanding

A. A bug where the system clock is used as a password
B. A session that stays valid after its timeout
C. A race condition where the state checked by a security test changes before the program acts on it
D. A log entry recorded with the wrong timestamp

1552. A password-reset token is generated like this. What does the program print? Application

```
import secrets
token = secrets.token_hex(16)
print(len(token))
```

A. 16
B. 64
C. 128
D. 32

1553. Two users choose the same password, but each gets a different random salt. What does the program print? Application

```
import hashlib
pw = b"Lahore@2026"
h1 = hashlib.sha256(b"k9F2" + pw).hexdigest()
h2 = hashlib.sha256(b"Qm7x" + pw).hexdigest()
print(h1 == h2, len(h1))
```

A. True 64
B. False 32
C. True 32
D. False 64

1554. An attacker tries thousands of passwords per minute against one account through the login API. Which control addresses this most directly? Application

A. Encrypting the password column
B. Hiding the login button for unknown IP addresses
C. Logging each failed attempt and taking no other action
D. Rate limiting with growing delays or temporary lockout after repeated failures

1555. An app sends users' private messages over plain HTTP, and an attacker on the same café Wi-Fi reads them. Which STRIDE category and control fit? Application

A. Tampering; add input validation
B. Repudiation; add audit logs
C. Denial of service; add rate limiting
D. Information disclosure; use TLS (HTTPS) for all traffic

1556. A search link puts user text in a URL query string. What does this code print? Application

```
const q = "a&b=c d";
console.log(encodeURIComponent(q));
```

A. a%26b%3Dc%20d
B. a&b=c%20d
C. a%26b%3Dc+d
D. a&b=c d

1557. A code review finds that the login function writes the submitted username and password to the debug log on every attempt. What is the problem and the fix? Application

A. Repudiation; log even more fields
B. Information disclosure; never log secrets, and mask or omit sensitive fields
C. Denial of service; rotate logs faster
D. Spoofing; hash the usernames in the log

1558. An attacker loads a bank's transfer page in an invisible frame on a game site and tricks users into clicking its button. Which defence fits? Application

A. Escape all HTML output on the transfer page
B. Use parameterised queries for transfers
C. Send X-Frame-Options: DENY or a CSP frame-ancestors directive so other sites cannot frame the page
D. Store passwords with bcrypt

1559. A team wants the security of a nightly build to rest on automated checks rather than manual reviews. Which pair of checks fits best in continuous integration? Application

A. SAST on the changed code and dependency scanning (SCA) on every build
B. A full manual penetration test on every commit
C. A bug bounty programme run on each build
D. Annual security awareness training

1560. A sign-up form validates usernames with an allow-list. What does this code print? Application

```
import re
names = ["ali_99", "ab", "sara khan", "x'--", "Omar2026"]
ok = [n for n in names if re.fullmatch(r"[A-Za-z0-9_]{3,16}", n)]
print(len(ok))
```

A. 3
B. 1
C. 2
D. 4

1561. A page puts a user's name into an HTML attribute after escaping it like this. What does the program print? Application

```
import html
name = '" onmouseover="alert(1)'
safe = html.escape(name, quote=False)
print('<input value="' + safe + '">')
```

A. <input value="" onmouseover="alert(1)">
B. <input value="" onmouseover="alert(1)">
C. <input value="<onmouseover=alert(1)>">
D. <input value="">

1562. This C code computes how many bytes to allocate for an array of records. What does it print? Application

```
#include <stdio.h>
#include <stdint.h>

int main(void) {
    uint32_t count = 65536, size = 65536;
    uint32_t total = count * size;   /* bytes to allocate */
    printf("%lu\n", (unsigned long)total);
    return 0;
}
```

A. 4294967296
B. 65536
C. 0
D. 2147483648

1563. To stop timing attacks that guess a reset token character by character, how should the marked line be rewritten? Application

```
import hmac

def check_reset_token(token: str, expected: str) -> bool:
    if token == expected:      # line to replace
        return True
    return False
```

A. if token is expected:
B. if token.lower() == expected.lower():
C. if hmac.compare_digest(token, expected):
D. if hash(token) == hash(expected):

1564. A banking site wants browsers to stop sending its session cookie on requests started by other sites, as one layer against CSRF. Which cookie attribute does this? Application

A. SameSite=Strict (or Lax)
B. HttpOnly
C. Secure
D. Max-Age=0

1565. A profile update endpoint copies every JSON field in the request into the user object. An attacker sends {"name": "Ali", "role": "admin"} and becomes an admin. What is this and the fix? Application

A. Mass assignment leading to elevation of privilege; bind only an allow-list of editable fields
B. SQL injection; use parameterised queries
C. Cross-site scripting; escape the name field
D. Spoofing; add multi-factor authentication

1566. A "link preview" feature fetches any URL the user supplies. An attacker submits http://169.254.169.254/latest/meta-data/ and gets cloud credentials. What is this and the fix? Application

A. Cross-site request forgery (CSRF); add anti-CSRF tokens
B. Open redirect; validate the Referer header
C. Server-side request forgery (SSRF); allow-list destinations and block internal and link-local addresses
D. Clickjacking; add X-Frame-Options

1567. An API accepts JSON Web Tokens and trusts whatever algorithm the token header names. An attacker sends a token with "alg": "none" and no signature, claiming another user's ID. What is the fix? Application

A. Encrypt the token payload with Base64
B. Shorten the token's expiry time to one minute
C. Verify the signature with a fixed, expected algorithm and reject tokens with alg none
D. Store the token in localStorage instead of a cookie

1568. Which DFD element in a threat model should be checked for Tampering, Information disclosure and Denial of service, and for Repudiation if it stores logs? Application

A. An external entity
B. A data store
C. A trust boundary
D. A data flow

1569. A download feature checks paths like this. What does the program print, and what does it mean? Application

```
import posixpath
BASE = "/srv/app/files"

def allowed(user_path):
    full = posixpath.normpath(posixpath.join(BASE, user_path))
    return full.startswith(BASE)

print(allowed("../files_backup/keys.txt"))
```

A. False, so the traversal is blocked
B. An error, because the path contains ..
C. True, because the file really is inside /srv/app/files
D. True, so a file outside the allowed folder is accepted

1570. A numeric ID is placed into a query without quotes. What does this program print? Application

```
import sqlite3
db = sqlite3.connect(":memory:")
db.execute("CREATE TABLE users (id INTEGER, name TEXT, pw TEXT)")
db.executemany("INSERT INTO users VALUES (?, ?, ?)",
               [(1, "ali", "a1"), (2, "sara", "s2"), (3, "omar", "o3")])
uid = "0 UNION SELECT pw FROM users"
rows = db.execute(f"SELECT name FROM users WHERE id = {uid}").fetchall()
print(sorted(r[0] for r in rows))
```

A. ['a1', 'o3', 's2']
B. []
C. ['ali']
D. An error, because id is an INTEGER column

1571. An attack tree's root is an OR node with two children: an AND node needing both a phished password (P 0.5) and a bypassed second factor (P 0.4), and a leaf "steal session cookie" (P 0.1). Assuming independence, what is the probability the goal is reached? Application

    A. 0.30
    B. 1.00
    C. 0.02
    D. 0.28

1572. Passwords are 8 characters from 62 letters and digits. An attacker tries 10 billion guesses per second against a fast hash, or 10,000 per second against bcrypt. Roughly how long does exhausting the whole space take in each case? Application

    A. About 6 hours with the fast hash; about 690 years with bcrypt
    B. About 6 minutes with the fast hash; about 6 years with bcrypt
    C. About 6 days with the fast hash; about 6,900 years with bcrypt
    D. About the same in both cases, because the space is the same

1573. A team moves the whole app to HTTPS and declares it secure. Users can still see other customers' records by changing the ID in the URL. Which assessment is correct? Analysis

    A. The TLS certificate is misconfigured
    B. TLS protects data in transit but does nothing for authorisation; the server must check ownership of each record
    C. The fix is to encrypt the IDs in the URL with Base64
    D. This is expected, because HTTPS only protects passwords

1574. What is wrong with this permission check? Analysis

```
def can_delete(user, doc):
    try:
        return acl.check(user, "delete", doc)
    except Exception:
        log.warning("ACL service unavailable")
        return True
```

    A. It should log at error level instead of warning
    B. It fails open: if the ACL service is down, everyone may delete, which breaks fail-safe defaults
    C. It should catch only KeyError
    D. It wastes time calling the ACL service on every request

1575. An upload feature blocks files ending in .php. Attackers upload shell.phtml and shell.PHP, and both run on the server. What is the best redesign? Analysis

    A. Add .phtml and .PHP to the block list
    B. Check that the file name contains no spaces
    C. Let only logged-in users upload files
    D. Use an allow-list of safe types, rename files on the server, and store them outside the web root where they cannot execute

1576. A SAST tool flags a query built by string concatenation. The concatenated value comes from a fixed enum of three column names inside the code. How should the team handle the finding? Analysis

    A. Ignore all SAST findings about SQL in future
    B. Confirm the data flow; it is likely a false positive, but switching to an allow-listed mapping documents the safety
    C. Treat it as a confirmed critical vulnerability and halt the release
    D. Turn off the SAST rule globally

1577. A requirements document says: "The system shall be secure." Why is this a poor security requirement? Analysis

A. It is too strict and will slow development
B. Security requirements should not appear in requirements documents
C. It cannot be tested or verified; it should state specific, measurable behaviour such as locking an account for 15 minutes after 5 failed logins
D. It should say "highly secure" to show its priority

1578. After a breach, investigators find the attacker was in the system for 200 days. Logs existed but no one reviewed them and no alerts were set. Which OWASP Top 10:2021 category does this match? Analysis

A. Cryptographic Failures
B. Insecure Design
C. Security Logging and Monitoring Failures
D. Identification and Authentication Failures

1579. A payment API signs each request with HMAC. An attacker captures a valid signed "transfer Rs 50,000" request and sends it again; the transfer happens twice. What is missing? Analysis

A. A stronger hash algorithm for the HMAC
B. Replay protection: include a nonce or timestamp in the signed data and reject reused or stale requests
C. Encryption of the request body
D. Input validation of the amount field

1580. A login page says "No account with that email" for unknown users and "Wrong password" for known ones, and it responds faster for unknown emails. What risk does this create, and what is the fix? Analysis

A. Username enumeration (information disclosure); return one generic message and keep timing uniform
B. Denial of service; add a CAPTCHA to the page
C. Repudiation; log every login attempt
D. No risk, because the passwords are still protected

1581. A company's build uses an internal package called acme-utils. An attacker publishes acme-utils version 99.0 to the public registry, and the next build installs it. Which controls address this best? Analysis

A. Run SAST on the company's own code
B. Rename the package every month
C. Add a web application firewall in front of the app
D. Resolve internal names only from the private registry (scoped names) and pin versions with hash-checked lock files

1582. A site sets Content-Security-Policy: script-src 'self' 'unsafe-inline'. A stored XSS payload <script>steal()</script> still runs. Which assessment is correct? Analysis

A. The policy is fine; the browser must be outdated
B. CSP never affects inline scripts
C. The fix is to add 'unsafe-eval' as well
D. 'unsafe-inline' allows inline scripts, which removes CSP's main XSS protection; use nonces or hashes and fix the output encoding

1583. Two requests redeem the same coupon at the same moment and the user is credited twice. What is the flaw and the right fix? Analysis

```
def redeem(user, code):
    c = db.get_coupon(code)
    if c.used:
        return "already used"
    credit(user, c.amount)          # takes ~200 ms
    db.mark_used(code)
    return "ok"
```

A. Missing input validation; check the coupon format with a regex
B. A TOCTOU race; make the check and the update atomic, for example UPDATE ... SET used = 1 WHERE code = ? AND used = 0 and credit only if one row changed
C. Weak authentication; require the password again
D. Slow code; make credit() faster

1584. Which SDLC tool is best placed to find each finding? (i) The app ships a library version with a published critical CVE. (ii) The deployed server omits security headers. (iii) Source code concatenates user input into SQL. Analysis

A. (i) SAST, (ii) SCA, (iii) DAST
B. (i) DAST, (ii) SAST, (iii) SCA
C. (i) SCA, (ii) SAST, (iii) DAST
D. (i) SCA, (ii) DAST, (iii) SAST

1585. What is the main danger in this code, and what is the fix? Analysis

```
import pickle, base64

def load_prefs(request):
    raw = request.cookies["prefs"]
    return pickle.loads(base64.b64decode(raw))
```

A. pickle can run arbitrary code while loading, so an attacker-crafted cookie gives remote code execution; use JSON and sign the cookie
B. Base64 is weak encryption; use AES on the cookie
C. The cookie may be too large; compress it first
D. Preferences may leak; mark the cookie HttpOnly

1586. A team puts a web application firewall (WAF) in front of its app to block SQL injection and closes the ticket without changing the code. Which assessment is correct? Analysis

A. The WAF fully fixes SQL injection, so code changes are unnecessary
B. WAFs make SQL injection worse and should be removed
C. A WAF is a useful extra layer, but rule-based filters can be bypassed with encodings, so the root fix is still parameterised queries
D. The team should instead hide the database error messages

1587. Inside a company network, microservices call each other without authentication because "the network is trusted". One service is compromised. What is the main flaw? Analysis

A. The services should share one database account to simplify access
B. No complete mediation between services: the compromised service can call every other one freely; each call should be authenticated and authorised
C. The firewall rules are too strict
D. The services need longer timeouts

SDLC Models Compared

1588. In which process model does the team write the acceptance test plan during requirements analysis and the integration test plan during architectural design? Understanding

A. Waterfall
B. Spiral model
C. V-model
D. Scrum

1589. What drives the number and content of loops in Boehm's spiral model? Understanding

A. A fixed number of phases set at the start
B. Risk: each loop resolves the biggest remaining risks
C. Customer-chosen sprint lengths
D. The number of increments in the contract

1590. Pressman groups prototyping and the spiral model under which category of process models? Understanding

A. Evolutionary process models
B. Linear sequential models
C. Specialised process models
D. Agile process models

1591. Which statement best separates incremental from iterative development? Understanding

A. Incremental always uses Scrum; iterative always uses waterfall
B. Incremental delivers once at the end; iterative delivers every week
C. Incremental adds new pieces of functionality in each release; iterative reworks and refines the same system through repeated cycles
D. They are two names for exactly the same thing

1592. According to Sommerville, what most distinguishes plan-driven processes from agile ones? Understanding

A. Plan-driven processes never produce documentation
B. Plan-driven processes plan all activities in advance and measure progress against the plan; agile planning is incremental
C. Agile processes forbid any planning
D. Plan-driven processes cannot deliver software in increments

1593. All else being equal, which model puts working software in front of users latest? Understanding

A. Incremental development
B. Scrum
C. Evolutionary prototyping
D. Waterfall

1594. A small co-located team is building a consumer app for an uncertain market. The founder is available daily and wants real-user feedback every two weeks to decide what to build next. Which approach fits best? Application

A. Waterfall
B. V-model
C. Spiral with formal risk reviews
D. Scrum (agile, two-week sprints)

1595. A government agency funds a large system using unproven sensor technology. Management wants an explicit risk analysis and a go/no-go decision before each major investment. Which model fits best? Application

A. Spiral model
B. Waterfall
C. Scrum
D. Throwaway prototyping alone

1596. A client cannot describe what the dashboard screens should do. The team plans to mock them up quickly in a design tool, agree the requirements with users, then build the real system on a different technology stack. What is this approach? Application

A. Evolutionary prototyping
B. Throwaway prototyping
C. V-model
D. Incremental delivery

1597. Flight-control software must be certified, and auditors want each requirement traced to a design element and to a planned test at every level, with test plans written early. Which model fits best? Application

A. Scrum
B. Evolutionary prototyping
C. Code-and-fix
D. V-model

1598. A payroll system must implement tax rules fixed by law for the year. The team has built two near-identical systems before, and the contract is fixed-price with a fixed scope. Which model fits best? Application

A. Waterfall
B. Scrum
C. Spiral
D. Evolutionary prototyping

1599. A hospital needs patient registration live in four months, billing three months later and pharmacy after that. The core requirements are known, but each module will be adjusted based on use of the previous one. Which model fits best? Application

A. Waterfall
B. V-model
C. Throwaway prototyping
D. Incremental development

1600. A company will meet its HR needs mostly by buying a commercial package, configuring it and integrating it with existing systems. Which of Sommerville's general process models is this? Application

A. Integration and configuration (reuse-oriented)
B. Waterfall
C. Incremental development
D. Spiral

1601. Waterfall plan: requirements 2 months, design 2, implementation 5, testing 3. Incremental plan: 2 months of core analysis, then three 3-month increments, each released at its end. How many months earlier do users get their first working software under the incremental plan? Application

A. 5
B. 9
C. 7
D. 3

1602. A ministry wants a citizen portal. Its rules change monthly as policy evolves, but a manager insists on waterfall 'because government projects need documentation'. What is the strongest response? Analysis

A. Use an incremental or agile approach with the documentation the ministry needs, because monthly rule changes would force costly rework in a waterfall
B. Waterfall is right, since documentation is only possible in plan-driven models
C. Use the V-model, because it handles change better than the waterfall
D. Use code-and-fix, because the requirements will change anyway

1603. A V-model project passes every system test, but at acceptance testing users say the workflow is not what they need. What limitation does this expose, and what would have helped? Analysis

A. Unit testing was too weak; more unit tests would have caught it
B. The V-model has no acceptance testing; adding it would fix the problem
C. Users validate the product only near the end; early prototypes or incremental releases would have exposed the mismatch sooner
D. Integration testing was skipped; it would have caught the workflow mismatch

1604. Weights: tolerance of changing requirements 0.4, regulatory traceability 0.35, early delivery 0.25. Scores out of 5 (change, traceability, early): Waterfall 1, 4, 1; V-model 1, 5, 1; Scrum 5, 2, 5; Incremental 4, 4, 4. Which model scores highest? Analysis

A. Scrum (3.95)
B. V-model (2.40)
C. Incremental (4.00)
D. Scrum and Incremental tie

1605. A team calls itself agile, but it writes a 200-page specification, freezes it at sign-off, codes for nine months and delivers once at the end. Which model is it really following? Analysis

A. Scrum
B. Waterfall (plan-driven, single delivery)
C. Incremental development
D. Spiral

1606. A safety-critical rail-control system will be built by 150 developers across three countries, and the regulator requires a safety case and design documents before deployment. Following Sommerville's guidance, which approach is most suitable? Analysis

A. Pure Scrum with no upfront design, relying on face-to-face communication
B. A mainly plan-driven process with incremental delivery, adding agile practices where they fit
C. Code-and-fix with heavy testing at the end
D. Throwaway prototyping as the whole development process

1607. Twelve features are planned. Waterfall delivers all twelve at month 12. Incremental delivers four features at each of months 5, 8 and 11. At the end of month 9, how many features are in users' hands under each plan? Analysis

A. Incremental 12, Waterfall 0
B. Incremental 4, Waterfall 0
C. Incremental 8, Waterfall 9
D. Incremental 8, Waterfall 0

UML Diagrams

1608. In a UML class diagram, what visibility does the '#' symbol before an attribute indicate? Understanding

A. Private
B. Protected
C. Package
D. Public

1609. A class diagram shows a dashed line with a hollow triangular arrowhead from class Wallet to «interface» Payable. What does it mean? Understanding

A. Wallet inherits from the class Payable
B. Wallet realises (implements) Payable
C. Wallet depends on Payable temporarily
D. Wallet is composed of Payable objects

1610. In a UML 2.5 sequence diagram, what does a solid line with an open (stick) arrowhead represent? Understanding

A. A synchronous call
B. A reply message
C. An asynchronous message
D. A creation message

1611. In an activity diagram, how does a flow final node (a circle with an X) differ from an activity final node (a bull's-eye)? Understanding

A. A flow final node ends the whole activity; an activity final node ends one flow
B. Both end the whole activity; they differ only in appearance
C. A flow final node starts a new parallel flow
D. A flow final node ends only the flow reaching it; an activity final node ends the whole activity

1612. In a class diagram, an attribute is shown underlined: totalStudents : Integer. What does the underline mean? Understanding

A. It is abstract
B. It is derived from other attributes
C. It is static (class-scope), shared by all instances
D. It is private

1613. Which UML diagram shows physical nodes such as servers and the artifacts (for example, .jar files) deployed on them? Understanding

A. Component diagram
B. Package diagram
C. Activity diagram
D. Deployment diagram

1614. A class diagram shows Hospital 1 —— 2..6 Ward and Ward 1 —— 10..30 Bed. What is the maximum number of beds one hospital can have? Application

A. 36
B. 60
C. 30
D. 180

1615. How should the relationship between Invoice and InvoiceLine in this code be drawn in a UML class diagram? Application

```
class InvoiceLine:
    def __init__(self, item, qty):
        self.item, self.qty = item, qty

class Invoice:
    def __init__(self, rows):
        # lines are created here and never shared
        self.lines = [InvoiceLine(i, q) for i, q in rows]
```

A. Aggregation: hollow diamond at the Invoice end
B. Composition: filled diamond at the Invoice end
C. Generalisation: hollow triangle pointing to Invoice
D. Dependency: dashed arrow from Invoice to InvoiceLine

1616. Which UML relationship best represents how Report uses Printer in this code? Application

```
class Printer:
    def send(self, text):
        print(text)

class Report:
    def __init__(self, title):
        self.title = title
    def print_to(self, printer):     # printer is only a parameter
        printer.send(self.title)
```

A. Association (solid line)
B. Composition (filled diamond at Report)
C. Dependency (dashed arrow from Report to Printer)
D. Realisation (dashed line with hollow triangle)

1617. 'Checkout' always performs 'Validate Payment'. 'Apply Coupon' runs during checkout only when the customer has a coupon. Which pair of use case relationships is drawn correctly? Application

A. «include» arrow from Validate Payment to Checkout; «extend» arrow from Checkout to Apply Coupon
B. «extend» arrow from Checkout to Validate Payment; «include» arrow from Apply Coupon to Checkout
C. «include» arrow from Checkout to Validate Payment; «extend» arrow from Apply Coupon to Checkout
D. Generalisation from Validate Payment to Checkout; generalisation from Apply Coupon to Checkout

1618. For a cart with 3 items and a total of 6,200, how many messages (calls and replies) does this sequence diagram exchange? Application

```
sd Checkout
Customer -> Cart : checkout()
loop [for each item in cart]
    Cart -> Inventory : reserve(item)
    Inventory --> Cart : ok
end
opt [total > 5000]
    Cart -> FraudCheck : verify()
    FraudCheck --> Cart : ok
end
Cart -> Payment : charge(total)
Payment --> Cart : receipt
Cart --> Customer : confirmation
```

A. 10
B. 12
C. 9
D. 14

1619. Using the same diagram, a cart has 4 items and a total of 3,500. How many synchronous calls (solid arrows, not replies) are sent? Application

```
sd Checkout
Customer -> Cart : checkout()
loop [for each item in cart]
    Cart -> Inventory : reserve(item)
    Inventory --> Cart : ok
end
opt [total > 5000]
    Cart -> FraudCheck : verify()
    FraudCheck --> Cart : ok
end
Cart -> Payment : charge(total)
Payment --> Cart : receipt
Cart --> Customer : confirmation
```

A. 7
B. 12
C. 5
D. 6

1620. A paper starts in Draft. Events occur in this order: submit, review (score 42), edit, submit, review (score 71), edit. Which state is it in at the end? Application

```
states: Draft, Submitted, Approved, Rejected
Draft     --submit-->                  Submitted
Submitted --review [score >= 50]-->    Approved
Submitted --review [score < 50]-->     Rejected
Rejected  --edit-->                    Draft
Approved  --edit-->                    (no transition)
```

A. Draft
B. Rejected
C. Submitted
D. Approved

1621. How many distinct paths lead from start to end in this activity diagram (the 'request documents' loop can happen at most once)? Application

```
[start] -> Receive application
Receive application -> <decision>
  [complete]   -> Check eligibility
  [incomplete] -> Request documents -> Receive application ... (once only)
Check eligibility -> <decision>
  [eligible]     -> <decision>
       [scholarship requested] -> Assess scholarship -> Issue offer
       [else]                  -> Issue offer
  [not eligible] -> Send rejection
Issue offer -> [end]
Send rejection -> [end]
```

A. 6
B. 4
C. 3
D. 5

1622. Which class-diagram fragment breaks a UML rule for composition? Analysis

A. House ◆—— Room, with multiplicity 1 at the House end and 1..* at the Room end
B. Room ◆—— Wall, with multiplicity 2 at the Room (composite) end
C. Order ◆—— OrderLine, with multiplicity 1 at the Order end and 1..* at the OrderLine end
D. Car ◇—— Passenger, with multiplicity 0..1 at the Car end

1623. After a fork, branch 1 runs 'Send email' then 'Send SMS'; branch 2 runs 'Update roster'. A join follows. If one processor runs the actions one at a time, how many different execution orders are possible? Analysis

A. 3
B. 6
C. 2
D. 1

1624. A team must show how an Order's status changes (Placed, Paid, Shipped, Cancelled) in response to events over its whole life. Which UML diagram is most suitable? Analysis

A. State machine diagram
B. Sequence diagram
C. Activity diagram
D. Class diagram

1625. A student draws a dashed arrow labelled «include» from 'Authenticate' to 'Register for Course', meaning that registration always requires authentication. What is wrong? Analysis

A. The arrow is reversed: «include» points from the base use case (Register for Course) to the included one (Authenticate)
B. Nothing; «include» arrows point from the included use case to the base
C. It should be «extend», because authentication is optional
D. It should be a generalisation, because registration is a kind of authentication

1626. Which set of relationships correctly models this code in a class diagram? Analysis

```
from abc import ABC, abstractmethod

class Payable(ABC):                 # plays the role of an interface
    @abstractmethod
    def pay(self): ...

class Player:
    pass

class Team(Payable):
    def __init__(self, players):    # players are created elsewhere
        self.players = players      # and live on if the team is deleted
    def pay(self):
        return len(self.players) * 100
```

A. Team realises Payable; Team aggregates Player (hollow diamond at Team)
B. Team inherits from Payable (generalisation); Team is composed of Player (filled diamond at Team)
C. Payable realises Team; Player depends on Team
D. Team realises Payable; Team is composed of Player (filled diamond at Team)

1627. In a sequence diagram, the 'return result' arrow from Database back to Controller is drawn as a solid line with a filled arrowhead. How would a UML reader interpret it? Analysis

A. As a reply, because any arrow going back to the caller is a reply
B. As an asynchronous signal
C. As a new synchronous call from Database to Controller, not a reply
D. As the destruction of the Database lifeline

Answer key and explanations

1. **C** — Sommerville defines software as computer programs together with associated documentation (requirements, design models, user manuals) and configuration data. A program on its own is only part of a software product.
2. **D** — IEEE Std 610.12 defines software engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software. The theory of computation belongs to computer science.
3. **D** — The NATO conference held in Garmisch, Germany, in 1968 used 'software engineering' as a deliberately provocative title and discussed the 'software crisis': large projects that were late, over budget and unreliable.
4. **A** — The software crisis described projects that ran late, overran budgets, did not meet requirements and were hard to maintain. Hardware costs were falling, not rising, and software does not physically wear out.
5. **C** — Sommerville lists maintainability, dependability and security, efficiency and acceptability as the essential attributes of good software. Coupling and cohesion are design properties, not product attributes.
6. **B** — Sommerville separates generic products (sold on the open market) from customised products (built for one customer). For generic products the developer owns the specification; for customised products the customer does.
7. **C** — A customised product is commissioned by and developed for a particular customer. The other three are generic products sold to anyone who wants them.
8. **C** — Sommerville's four fundamental activities are software specification, design and implementation (development), validation and evolution. Communication-planning-modelling-construction is Pressman's framework (which also adds deployment), and inception-elaboration-construction-transition are RUP phases.
9. **A** — Pressman's generic process framework has five activities: communication, planning, modelling, construction and deployment. Umbrella activities such as risk management and quality assurance run across all of them.
10. **C** — Pressman draws software engineering as layers: quality focus at the bottom, then process, methods and tools. The quality focus is the bedrock; the process layer is called the foundation that holds the technology layers together.
11. **B** — Umbrella activities run throughout a project: project tracking and control, risk management, quality assurance, technical reviews, measurement, software configuration management, reusability management and work-product preparation. Construction, deployment and communication are framework activities.
12. **D** — Pressman's 'actual' software failure curve shows a spike in failures after each change. Every modification can introduce new defects, and repeated changes erode the structure, so the failure rate drifts upward. This is deterioration through change, not physical wear.
13. **B** — Fred Brooks stated in The Mythical Man-Month (1975) that adding people to a late project makes it later, because of ramp-up time and growing communication overhead. The others are Parkinson's law, Moore's law and Conway's law.
14. **A** — The Code has eight principles: Public, Client and Employer, Product, Judgment, Management, Profession, Colleagues and Self. The first, Public, requires engineers to act consistently with the public interest.
15. **B** — Brooks (1986) split difficulties into essential ones, inherent in software (complexity, conformity, changeability, invisibility), and accidental ones that tools and languages can remove. He argued no single technology would give an order-of-magnitude gain against the essential ones.
16. **A** — Sommerville says computer science is concerned with theories and fundamentals, while software engineering is concerned with the practicalities of producing useful software. System engineering is wider still, covering hardware, software and processes.
17. **D** — Pressman lists 'if we get behind schedule, we can add more programmers and catch up' as a management myth. New people need training from the existing team, which slows everyone down, in line with Brooks's law.
18. **A** — Pressman lists 'a general statement of objectives is sufficient to begin writing programs' as a customer myth. Ambiguous objectives lead to costly rework once the details emerge.
19. **C** — Pressman lists 'once we write the program and get it to work, our job is done' as a practitioner myth. Much of the effort on a long-lived system is spent after first delivery, on maintenance and evolution.

20. **B** — Each pair of people is one communication path: 6 x 5 / 2 = 15. 30 counts each pair twice, 36 is 6 squared, and 12 is just 6 x 2. Brooks used this growth to explain why adding people adds overhead.
21. **D** — 5 people have 5 x 4 / 2 = 10 paths; 8 people have 8 x 7 / 2 = 28. The increase is 28 - 10 = 18. Three new people add far more than three new paths, which is the overhead behind Brooks's law.
22. **B** — The system is built for one customer, who owns the specification, so it is a customised or bespoke product. COTS and shrink-wrapped are both names for generic products sold on the open market.
23. **C** — Embedded control systems control and manage hardware devices, often with real-time constraints. ABS software reads wheel sensors and drives the brake actuators, so it is embedded.
24. **A** — Batch processing systems process large volumes of input data in batches to create corresponding outputs, with no interaction during the run. Billing and payroll systems are Sommerville's standard examples.
25. **B** — Interactive transaction-based applications run on a remote server and are accessed by users from their own devices, typically web applications such as e-commerce and booking systems backed by a database.
26. **C** — The Code says software engineers act in the best interests of their client and employer consistent with the public interest, and the public interest comes first. The engineer should raise the risk rather than stay silent.
27. **D** — Acceptability means software must be understandable, usable and compatible with the other systems its users work with. Users rejecting a confusing app is an acceptability failure, even if the code is fast and reliable.
28. **B** — Pressman defines construction as code generation, manual or automated, plus the testing needed to uncover errors in that code. Modelling produces analysis and design models, and deployment delivers the increment to the customer for feedback.
29. **B** — Hooker's principle 'what you produce, others will consume' says to design and code with the people who will maintain, extend or test the work in mind. Cryptic names ignore those future readers.
30. **D** — Pressman contrasts the idealised software curve (high early failures, then flat) with the actual curve, in which each change causes a spike and the baseline rises. The upward creep comes from side effects of change, not from physical wear.
31. **A** — Sommerville stresses there is no universal method: the right techniques depend on the type of system. A small MVP still needs to know what to build, check it works and change it, but can use lightweight, agile practices.
32. **C** — Person-months are not interchangeable with calendar time when tasks need coordination. Existing staff must train the newcomers, and pairwise paths jump from 4 x 3 / 2 = 6 to 8 x 7 / 2 = 28, so Brooks's law predicts a later finish than the simple division.
33. **C** — Pressman notes that software is engineered, not manufactured, and does not wear out. A failure comes from a defect in the design or code, so an identical copy fails in exactly the same way. The fix is to correct the defect.
34. **D** — Sommerville's list of professional responsibilities includes competence: engineers should not misrepresent their level of competence or knowingly accept work outside it. Claiming expertise they do not have breaks this.
35. **A** — Third-generation hardware made far larger systems possible. Ad hoc, individual approaches to programming did not scale to that size, so projects ran late, over budget and unreliable. Software engineering arose to bring discipline to large-scale development.
36. **A** — For customised products the customer owns and controls the specification, so the hospital decides. For generic products such as the game, the developer decides which changes to make.
37. **D** — Brooks argued that tools, languages and environments attack accidental difficulties. The essential difficulties (complexity, conformity, changeability and invisibility) remain, so no single tool brings a tenfold improvement.
38. **B** — Software configuration management controls versions of every work product, including code and configuration files, and manages change. Lost changes and unknown deployed versions are classic SCM failures.
39. **D** — Building the wrong product is a requirements failure. Communication gathers requirements from stakeholders, and modelling builds analysis models that let them confirm understanding before coding starts.
40. **A** — One team of 9: 9 x 8 / 2 = 36 paths. Three teams of 3: 3 paths each, so 9, plus 3 paths among the three leads, giving 12. Splitting into small teams sharply cuts coordination overhead.
41. **A** — Pressman defines software as instructions that deliver the required function, data structures that let programs handle information, and descriptive information, in hard copy and virtual form, about how to use and operate the programs. Like Sommerville, he refuses to reduce software to code alone.

42. **B** — The bathtub curve is Pressman's hardware failure curve. Infant-mortality failures fall as defects are fixed, the rate stays low for most of the product's life, then climbs again as physical components wear out. Software has no equivalent wear-out phase.
43. **C** — The idealised curve drops after early defects are fixed and then flattens, since software has no physical parts to wear out. The actual curve differs only because maintenance changes introduce new defects; it does not reach zero, because not every latent defect is ever found.
44. **D** — Hardware quality problems can be introduced during manufacturing, but software is effectively finished once it is developed. Duplicating it is trivial, so software projects cannot be managed as if they were manufacturing projects; the effort, and the risk, lie in engineering.
45. **A** — Sommerville names heterogeneity (systems spread across different computers and devices), business and social change, security and trust, and scale (from tiny embedded systems to cloud-scale ones). The other lists are Brooks's essential difficulties, Sommerville's attributes of good software and his fundamental activities.
46. **B** — Sommerville notes that for systems with a long life, the cost of changing software after it is in use (evolution or maintenance) is often several times the development cost. This is one reason maintainability is listed among the essential attributes of good software.
47. **C** — Hooker's first principle, 'the reason it all exists', says software exists to provide value to its users, so every decision should be judged against that. KISS is about avoiding needless complexity, 'maintain the vision' about conceptual integrity and 'be open to the future' about designing for change.
48. **D** — 'Maintain the vision' argues that conceptual integrity is essential: when the architecture's vision is compromised, the system ends up held together by the wrong kind of glue. 'Think!' is the seventh principle about clear thought before action.
49. **A** — Methods give the technical how-to's: communication, requirements analysis, design modelling, program construction, testing and support. Process is the foundation that holds the layers together, tools automate process and methods, and the quality focus is the bedrock.
50. **B** — Polya's steps map onto practice: understanding the problem is communication and analysis, planning is modelling and design, carrying out the plan is code generation, and examining the result is testing and quality assurance. The other lists are scrambled or come from other frameworks.
51. **C** — The Code has eight principles: Public, Client and Employer, Product, Judgment, Management, Profession, Colleagues and Self. Four is the number of professional responsibilities Sommerville adds separately (confidentiality, competence, IP rights and computer misuse).
52. **D** — Sommerville says engineers must respect confidentiality, not misrepresent their competence, protect intellectual property rights and not misuse other people's computers. The first wrong list is four of the ACM/IEEE Code's eight principles; the last is dependability properties.
53. **A** — Pressman groups myths by who holds them: managers (for example, adding people to catch up), customers (for example, a vague statement of objectives is enough to start) and practitioners (for example, the job ends when the program runs).
54. **B** — System software is written to service other programs: compilers, editors, file utilities, operating-system components, drivers and networking software. Application software solves a specific business need, scientific software is number-crunching, and embedded software sits inside a product to control it.
55. **C** — Legacy software is often business-critical, so retiring it carries high cost and risk. Pressman says it is evolved when it must adapt to new technology or business needs, extended for new requirements or re-architected to work in modern environments. It does not stay unchanged.
56. **D** — Brooks credits high-level languages, time-sharing and unified programming environments with removing accidental difficulty. The other options claim to eliminate essential difficulties (complexity, changeability, invisibility), which Brooks argues no single technique can do.
57. **A** — Invisibility: unlike a building with a floor plan, software is described by several overlapping graphs (control flow, data flow, dependencies), none of which shows the whole. Conformity is about fitting arbitrary external interfaces, and changeability is about constant pressure to change.
58. **B** — Dependability and security includes reliability, security and safety. The other options describe maintainability, efficiency and acceptability, the remaining three of Sommerville's four essential attributes.
59. **C** — Reusability management sets the criteria for reuse, including software components, and establishes ways to achieve it. Configuration management controls changes, work product preparation covers creating models, documents and logs, and SQA defines and runs quality activities.

60. **D** — Sommerville defines a software process as a structured set of activities needed to develop a system, and a process model as a simplified, abstract representation of that process from one perspective. Waterfall, incremental and integration-and-configuration are examples.
61. **A** — The Judgment principle covers integrity and independence, including rejecting bribery and disclosing conflicts of interest. Profession concerns advancing the profession's integrity and reputation, Self is about lifelong learning, and Product is about meeting the highest professional standards.
62. **B** — As a product, software transforms information and delivers computing power. As a vehicle, it is the basis for controlling the computer (operating systems), communicating information (networks) and creating other programs (tools). The other options invent distinctions not in the text.
63. **C** — System engineering is concerned with whole computer-based systems: hardware, software, processes and their deployment. Software engineering is one part of that wider effort. Confusing the two, or thinking system engineering means operating systems, is a common trap.
64. **D** — Sommerville stresses that methods and tools must suit the type of system: embedded safety-critical systems need rigorous analysis and documentation, while games and interactive web systems benefit from rapid prototyping. The fundamentals (process, dependability, requirements, reuse) apply everywhere, but the specific methods differ.
65. **A** — Paths = n(n - 1)/2 = 10 x 9 / 2 = 45. 100 squares n, 90 forgets to halve (counting each pair twice), and 20 doubles the headcount instead of counting pairs.
66. **B** — Each team of 4 has 4 x 3 / 2 = 6 paths, so 18 for three teams, plus 3 x 2 / 2 = 3 paths among the leads: 21 in total. A single team of 12 would have 66. 18 forgets the lead paths, and 24 counts 6 lead paths by not halving.
67. **C** — 24 / 6 = 4 calendar months is what the 'man-month' arithmetic predicts. Brooks's point is that people and months are interchangeable only for perfectly partitionable tasks, like harvesting a field; software tasks need communication and sequencing, so the real duration is longer.
68. **D** — Pressman lists this as a management myth. The reality is that the book may exist, but it may be unused, outdated or not reflect modern practice. Customer myths are held by those requesting software, practitioner myths by developers.
69. **A** — Pressman classes this as a customer myth. In reality, the impact of a change grows the later it is introduced: a change requested during requirements is cheap, but one requested after release can force design changes and costly rework.
70. **B** — This is a practitioner myth. Pressman's answer is that technical reviews of models, designs and code can be applied from project inception and often catch defects more effectively than testing. A clean compile proves only syntax, not quality.
71. **C** — Pressman lists this as a management myth. If an organisation does not know how to manage and control software projects itself, it will still struggle when the work is outsourced, because it must specify, monitor and accept what the vendor delivers.
72. **D** — Pressman counters this practitioner myth: software engineering is not about producing documents for their own sake but about creating a quality product. Quality reduces rework, and less rework means faster delivery.
73. **A** — It is sold on the open market and the developer owns the specification, so it is a generic product. A bespoke product is built for one customer who owns the specification; embedded and batch are application types, not product categories.
74. **B** — Sommerville notes that the distinction between generic and customised products is increasingly blurred: systems such as ERP packages are built as generic products, then configured and adapted for individual customers. Nothing was written from scratch here.
75. **C** — Modelling and simulation systems model physical processes or situations, often with many interacting objects, and are typically used by scientists and engineers. A weather model is a classic example. A data collection system gathers raw readings; it does not simulate.
76. **D** — Data collection systems gather data from their environment through sensors and pass it on to other systems for processing. They often operate in hostile physical conditions, such as a riverbank. They are not stand-alone, because they send their readings over a network.

77. **A** — Stand-alone applications run on a local computer and include all needed functionality without a network. Transaction-based applications run on a remote server accessed by users; systems of systems combine several independent systems; batch systems process large inputs without user interaction.
78. **B** — Conformity means software must fit arbitrary interfaces and rules created by people and institutions, rather than simple natural laws. Tax law and existing file formats are imposed from outside. Changeability is the constant pressure for change, and 'accidental complexity' is not one of the four essential difficulties.
79. **C** — Technical reviews assess work products to uncover and remove errors before they spread to the next activity. Measurement collects data about process and product, risk management assesses risks to the project, and tracking and control compares progress with the plan.
80. **D** — Software project tracking and control lets the team assess progress against the project plan and take action to keep to the schedule. SQA defines and runs quality-assurance activities, reusability management handles reuse, and technical reviews examine work products for errors.
81. **A** — The Judgment principle requires integrity and independence in professional judgment, including rejecting bribery and declining payments that would compromise an evaluation. Self covers lifelong learning, Colleagues fairness to co-workers, and Public the wider public interest.
82. **B** — The Colleagues principle requires engineers to be fair to and supportive of colleagues, which includes giving full credit for the work of others. Product concerns the quality of what is built, and Management concerns leaders' duties.
83. **C** — Engineers should respect the confidentiality of employers and clients, whether or not a formal agreement exists. Taking a client's confidential data for personal use is a breach of confidentiality. Computer misuse concerns using technical skill to interfere with other people's computers.
84. **D** — Sommerville says engineers should be aware of laws on intellectual property, such as patents and copyright, and protect the IP of employers and clients. Using pirated software violates the vendor's copyright and licence.
85. **A** — Planning creates the project's 'map': technical tasks, likely risks, required resources, work products and a schedule. Communication gathers requirements from stakeholders, modelling builds analysis and design models, and deployment delivers the software for evaluation.
86. **B** — Paths = n(n - 1)/2: 7 x 6 / 2 = 21 and 10 x 9 / 2 = 45, a rise of 24 paths or about 114%. Because the count grows roughly quadratically, coordination cost outpaces headcount, which is one reason behind Brooks's law.
87. **C** — Heterogeneity is the challenge of building systems that operate as distributed systems across different types of computers and mobile devices, often integrated with older legacy systems written in other languages. That is exactly this situation.
88. **D** — P: 2 x (6 x 5 / 2) + 1 = 31. Q: 3 x (4 x 3 / 2) + 3 = 21. One team of 12 would have 12 x 11 / 2 = 66. Smaller teams reduce paths, though they make the leads a communication bottleneck.
89. **A** — One team: 15 x 14 / 2 = 105. Split: each team of 5 has 10 paths, so 30, plus 3 paths among the leads, so 33. The saving is 105 - 33 = 72. Smaller teams cut coordination cost, at the price of relying on the leads to pass information.
90. **B** — Software does not wear out, but it does need to evolve: new requirements, new platforms and new laws force changes, and each change can introduce defects that raise the failure rate. Sommerville notes that change costs over a long-lived system's life often exceed its development cost.
91. **C** — The Management principle requires managers and leaders to promote an ethical approach to managing software development, including ensuring realistic quantitative estimates of cost, schedule and quality. Ordering misleading estimates breaks that duty.
92. **D** — Build automation removes difficulty created by our tools and production methods, which is accidental. The other options try to remove essential properties: conformity is imposed from outside, diagrams only partly reveal invisible structure, and changeability cannot be wished away.
93. **A** — For a generic product the developer owns the specification and spreads costs across many buyers, so it is cheaper and quicker but may not fit exactly. For a customised product the customer owns the specification, so it fits, but development is slower and more expensive.
94. **B** — Crashes and data loss are failures of dependability (reliability and safety of data). Difficulty in changing tangled, undocumented code is poor maintainability. Efficiency concerns resource use and response time, and acceptability concerns usability and fit with users' work.

95. **C** — Measurement defines and collects process, project and product measures that help the team deliver software that meets stakeholders' needs, and it supports the other activities. With no data, trends in quality cannot be judged. Nothing in the scenario points to lost versions, unassessed risks or reuse.
96. **D** — Pressman's reply to the practitioner myth that 'the only deliverable is the working program' is that a working program is one part of a configuration that includes models, documents and plans. The other three statements are classic practitioner, customer and management myths.
97. **A** — Brooks's law bites when work is late, hard to partition and dependent on communication and training. Data entry on independent records is perfectly partitionable and needs no shared understanding of a design, so extra people genuinely add throughput, especially if added early.
98. **B** — KISS asks for designs as simple as possible, though not simplistic, and 'what you produce, others will consume' reminds developers that others must read and maintain their code. Clever but cryptic code fails both.
99. **C** — Sommerville argues that no single method fits all systems. Games depend on user experience and benefit from iterative prototyping; safety-critical embedded systems need careful analysis, verification and records for certification.
100. **D** — Pressman's construction covers code generation (manual or automated) and the testing needed to uncover errors. In Sommerville's terms, writing the code is development (design and implementation) and testing is validation.
101. **C** — In the waterfall model each phase must finish before the next starts: requirements analysis and definition, system and software design, implementation and unit testing, integration and system testing, and finally operation and maintenance.
102. **A** — Sommerville says the waterfall model suits projects whose requirements are well understood and stable, such as some embedded and critical systems, because changes after sign-off are expensive to absorb.
103. **D** — Barry Boehm proposed the risk-driven spiral model (published in 1988). Royce described the sequential model in 1970, Kent Beck created Extreme Programming, and Ken Schwaber co-created Scrum.
104. **B** — Each loop of Boehm's spiral includes a risk assessment and reduction step, so the project is driven by its biggest risks. The other options describe waterfall or Scrum features, not the spiral.
105. **B** — Each loop has four sectors: objective setting (define objectives, constraints and alternatives), risk assessment and reduction, development and validation, and planning (review and decide whether to continue). RUP phases and Pressman's activities are distractors.
106. **B** — The V-model is a variant of the waterfall that links every development phase to a test level. Test planning for each level starts when its matching development phase is done, and execution happens on the way up the right side.
107. **A** — Unit testing sits at the bottom of the V, opposite detailed or module design, because it checks that each component meets its own design. Integration testing pairs with architectural design and system testing with the system specification.
108. **A** — Acceptance testing is at the top of the V, opposite requirements analysis. It checks that the system meets the users' needs, so its tests are planned from the user requirements.
109. **C** — In incremental development, specification, development and validation are interleaved, and the system is developed as a series of increments, each adding functionality. The other options describe waterfall, throwaway prototyping and the spiral.
110. **B** — A throwaway prototype exists only to learn, typically about unclear requirements, and is discarded. An evolutionary prototype is refined into the delivered system.
111. **C** — Verification asks whether the software conforms to its specification ('building the product right'). Validation asks whether it meets the customer's real needs ('building the right product').
112. **B** — RUP's four phases are inception (business case), elaboration (understanding the domain and an architecture baseline), construction (design, code and test) and transition (deploying to users).
113. **B** — Sommerville's three general models are the waterfall model, incremental development, and integration and configuration (reuse-oriented, built from existing components). Most real processes mix them.
114. **A** — Royce's 'Managing the Development of Large Software Systems' showed the sequential flow, called it risky and invited failure, and recommended iteration and a preliminary pilot. The 'waterfall' label came later from other authors.

115. **C** — Sommerville lists two management problems: progress is less visible because documents are not produced for every version, and regular change tends to degrade structure unless time is spent on refactoring.
116. **D** — In Boehm's spiral there are no fixed phases. Each loop chooses its development approach based on risk: prototyping when user-interface risk dominates, a more waterfall-like approach when requirements are well understood.
117. **A** — Stable, well-understood requirements and a fixed-price contract favour the plan-driven waterfall model. Prototyping and spiral pay off mainly when requirements or risks are uncertain.
118. **A** — The V-model links each development phase to a test level and plans tests early, which gives the traceability that safety certification demands. That is why it is common in medical, automotive and defence work.
119. **B** — The spiral model is designed for large, high-risk projects: each loop identifies the biggest risks and reduces them (for example with prototypes or simulations) before committing more money.
120. **D** — Prototyping lets stakeholders see and react to a working model, which is the best way to pin down fuzzy user-interface requirements. The waterfall and V-model both expect requirements fixed up front.
121. **C** — Delivering a usable core first and adding features in later releases is incremental development (with incremental delivery). Customers get value early and their feedback shapes later increments.
122. **D** — In the waterfall model the system goes into operation only after integration and system testing, so the first delivery is at 2 + 3 + 4 + 2 = 11 months. Month 9 is the end of implementation, before system testing.
123. **C** — Risk exposure: A = 0.2 x 500,000 = 100,000; B = 0.5 x 150,000 = 75,000; C = 0.1 x 900,000 = 90,000. A has the highest exposure, so the risk-driven spiral addresses it first. B has the highest probability and C the biggest loss, but neither has the largest product.
124. **D** — Integration testing checks interfaces between components, which are defined during architectural design, so the two face each other in the V. Unit testing pairs with detailed design and acceptance testing with requirements.
125. **A** — In Sommerville's description, the planning sector reviews the project and decides whether to continue with a further loop; if so, plans are drawn up for the next phase.
126. **D** — The waterfall model's inflexible partitioning into stages makes late changes expensive, since earlier phases must be redone. The other three accommodate change through iteration.
127. **A** — Elaboration develops an understanding of the problem domain, establishes an architectural framework and addresses the main risks. Inception builds the business case, construction builds the system, and transition deploys it.
128. **C** — Integration and configuration (reuse-oriented development) relies on reusable components or systems that are configured and integrated, with only the gaps built from scratch. It cuts cost and delivery time but can compromise requirements.
129. **D** — Incremental delivery puts the highest-priority services in the first increment. The 'must' features total 5 + 10 + 4 = 19 person-weeks, which fits in 20; adding any other feature (at least 6 more) would exceed capacity.
130. **C** — The mock-ups were used only to learn requirements and were not turned into the product, which is throwaway prototyping. In evolutionary prototyping the prototype itself grows into the delivered system.
131. **D** — Rejected features mean the requirements were wrong, and the waterfall only shows working software at the end. Early versions or prototypes let users correct misunderstandings while change is still cheap.
132. **A** — Boehm's spiral pays off on large, risky projects. For a small, well-understood job, formal risk analysis each loop is overhead with little benefit; a simple plan-driven or agile approach is enough.
133. **C** — The V-model keeps the waterfall's sequence but makes verification and validation explicit: acceptance tests are planned from requirements, system tests from the specification, and so on. Tests still run after coding, moving up the right side.
134. **B** — Incremental means adding new pieces of functionality one at a time. Iterative means revisiting and refining existing functionality in repeated passes. Most modern processes combine both, but the scenario separates them.
135. **A** — X: (100,000 - 20,000) / 20,000 = 4. Y: (75,000 - 50,000) / 5,000 = 5. Y returns more exposure reduction per rupee spent, so it has higher leverage, even though X removes more exposure in total.
136. **D** — Sommerville notes that large, distributed teams, long-lived systems and external regulation favour plan-driven approaches with documented requirements. Agile practices can still be used inside individual teams.

137. **D** — Pressman warns that stakeholders may mistake a prototype for a working product built with quality in mind. The fix is to agree up front that the prototype defines requirements and is then discarded or properly re-engineered.
138. **B** — Sommerville notes that regular change in incremental development degrades structure unless time and money go into refactoring. Refactoring restores maintainability without abandoning the benefits of increments.
139. **B** — The product was built right (it matches the specification, so verification passed) but it is not the right product (validation failed). The error lies in the requirements, the top-left of the V, which acceptance testing is paired with.
140. **C** — The V-model keeps the waterfall's one-pass structure, so late requirement changes ripple through every paired phase and test plan. The other pairings contradict the defining feature of each model.
141. **A** — Code-and-fix means writing a program, then fixing it whenever something goes wrong, with no requirements or design phase. It can work for tiny one-person programs, but code structure degrades quickly with each fix, which is why disciplined process models were developed.
142. **B** — Sommerville's prototype process starts by deciding what the prototype is for (for example, to test the UI), then deciding what to include and leave out, building it, and finally evaluating it with users against the objectives. The other lists are scrambled or belong to waterfall and RUP.
143. **C** — In Boehm's 1988 spiral, distance from the origin shows the cumulative cost spent, and the angular direction shows progress through each loop. Each full turn is one phase, ending with a review and a commitment to the next loop.
144. **D** — A parallel process flow executes activities in parallel, for example modelling one part of the software while another part is being constructed. A linear flow runs the five framework activities in sequence, an iterative flow repeats activities before moving on, and an evolutionary flow cycles through them, producing a more complete version each time.
145. **A** — In the concurrent model, all activities exist concurrently but reside in different states: inactive, under development, awaiting changes, under review, under revision, baselined and done. Events, such as a customer requesting a change, move an activity from one state to another, so the model gives an accurate picture of the project's current state.
146. **B** — The dynamic perspective shows RUP's phases over time, the static perspective shows the workflows (disciplines) carried out, and the practice perspective lists good practices to follow. Inception, elaboration and construction are three of the four phases, not perspectives.
147. **C** — Transition deploys the system to users: beta testing, training, data conversion and roll-out. Inception establishes the business case, elaboration builds the architecture and tackles key risks, and construction builds and tests the bulk of the system.
148. **D** — RUP's six good practices are: develop software iteratively, manage requirements, use component-based architectures, visually model software (UML), verify software quality and control changes. The wrong options contradict iterative development and continuous quality verification.
149. **A** — Reuse-based development draws on stand-alone application systems (often called COTS), object collections developed as a package to integrate with a framework, and web services developed to service standards. The other options are process models, test levels and documents.
150. **B** — Reusing existing systems means less software to write, lower cost and risk, and faster delivery. However, compromises on requirements are inevitable, and new versions of reused components are outside the organisation's control.
151. **C** — Change tolerance means the process lets changes be absorbed cheaply, usually because development is incremental, so only the current or next increment is affected. Change anticipation, by contrast, uses activities such as prototyping to foresee changes before costly rework is needed.
152. **D** — Pressman calls the waterfall the classic life cycle: a systematic, sequential approach that begins with customer requirements and moves through planning, modelling, construction and deployment. It is the oldest paradigm for software engineering.
153. **A** — Because each phase depends on the previous one, waterfall projects can leave people idle while they wait for upstream work. Pressman observes that the time spent waiting can exceed the time spent on productive work.

154. **B** — Sommerville's testing process moves from component (unit) testing by developers, to system testing of the integrated system, to customer testing with real data before the system is accepted for use. The other lists mix up testing techniques with stages.
155. **C** — Sommerville's design activities are architectural design (overall structure), database design, interface design (between components) and component selection and design. The other options are RUP phases, Pressman framework activities and testing stages.
156. **D** — PSP is a personal process in which each engineer plans, designs, reviews the design, develops (code, review, compile, test) and runs a postmortem using the time and defect data collected. The other lists are Pressman's Unified Process phases, his generic framework and the spiral's sectors.
157. **A** — The staged CMMI model rates an organisation's process maturity at five levels: 1 initial, 2 managed, 3 defined, 4 quantitatively managed and 5 optimising. The first wrong option swaps levels 2 and 3 and levels 4 and 5, a common slip.
158. **B** — Formal methods specify, develop and verify software using mathematical notation. Ambiguity, incompleteness and inconsistency can be found through mathematical analysis. Pressman notes they are time-consuming and need special training, so they are used mainly for safety-critical software.
159. **C** — AOSD provides a process and methods for defining, specifying, designing and building aspects: mechanisms that localise concerns cutting across many modules. Without aspects, code for concerns such as logging is scattered throughout the system.
160. **D** — Sommerville says the waterfall remains appropriate for embedded systems interfacing with inflexible hardware, critical systems needing extensive safety and security analysis of the specification and design, and large systems developed by several partner companies. Volatile or exploratory projects suit incremental approaches better.
161. **A** — Sommerville lists three benefits: lower cost of change, easier customer feedback on working software, and earlier delivery of useful software. Lower visibility and degrading structure are listed as problems of incremental development, not benefits.
162. **B** — Most systems need basic facilities (such as data handling) used by several parts. Because detailed requirements emerge only as each increment is implemented, it is hard to identify these shared facilities in advance.
163. **C** — Pressman lists the UP phases as inception, elaboration, construction, transition and production. In production, the ongoing use of the software is monitored, support is provided and defect reports and change requests are evaluated. Sommerville's RUP description stops at transition.
164. **D** — Royce suggested building a pilot (preliminary) version early to uncover problems, alongside other advice such as documenting the design and involving the customer. His paper criticised the single-pass sequential model now called waterfall.
165. **A** — The original plan ends at month 2 + 3 + 5 + 2 = 12. Rework adds 1 + 1.5 + 2 + 2 = 6.5 months, giving 18.5. 16.5 forgets the retest, and 24 assumes every phase is repeated in full. Late discovery of requirements defects is the waterfall's costliest failure.
166. **B** — Increment 1 ends at month 4, increment 2 at 4 + 2.5 = 6.5, and increment 3 at 6.5 + 2.5 = 9. 7.5 assumes every increment takes 2.5 months, 12 assumes every increment takes 4 months, and 6.5 is when increment 2 arrives.
167. **C** — Risk exposure = probability x loss: R1 = 120,000, R2 = 100,000, R3 = 90,000. The total is 310,000. 1,550,000 adds the losses without weighting them, 103,333 is the average exposure rather than the total, and 120,000 is only the largest single exposure.
168. **D** — Leverage = (150,000 - 100,000) / 60,000 = 0.83. A value below 1 means each rupee spent removes less than a rupee of exposure. 2.5 divides the before-exposure by the cost, and 1.2 inverts the ratio.
169. **A** — Sommerville explains that the spiral chooses a development model per loop: prototyping if user-interface risks dominate, formal transformations if safety risks dominate, and a waterfall model if the main identified risk is sub-system integration.
170. **B** — System testing checks the whole integrated system against the system specification and is paired with the system requirements phase in the V-model. Unit testing pairs with detailed design and integration testing with architectural design; acceptance testing is planned from user requirements, not code.

171. **C** — The V-model plans each test level alongside its matching development phase. Acceptance tests are derived from user requirements, so they are planned during requirements analysis. Planning tests early exposes ambiguous requirements before they become defects.
172. **D** — The customer's change request is an event that triggers a transition from 'under development' to 'awaiting changes'. 'Baselined' and 'done' describe completed, agreed work, and 'inactive' applies before an activity starts.
173. **A** — Inception establishes the business case: it identifies external entities (people and systems) that interact with the system and assesses whether the system is worth building. Elaboration develops the architecture and project plan; construction builds the system; transition deploys it.
174. **B** — Beta testing, user training and data conversion move the system into its working environment, which is the transition phase. Construction is mainly system design, programming and testing of the bulk of the product.
175. **C** — Writing code first and fixing faults as they appear, with no specification or design, is code-and-fix. A throwaway prototype is deliberately built to learn and then discarded; here the patched code is the product.
176. **D** — In evolutionary prototyping the prototype is not discarded: it is refined repeatedly until it becomes the delivered system. A throwaway prototype is used only to clarify requirements and is then replaced by a properly engineered system.
177. **A** — Expected cost without prototype = 0.4 x 1,000,000 = 400,000. With it: 150,000 + 0.1 x 1,000,000 = 250,000. Saving = 400,000 - 250,000 = 150,000. 300,000 ignores the prototype's own cost, 250,000 is the expected cost with prototyping, and 400,000 is the cost without it.
178. **B** — Musts first: A + B = 14. C (5) would make 19, over capacity, so it is skipped. E (4) makes 18 and fits. D (3) would make 21, so it is skipped. Result: A, B and E. A, B and C exceeds capacity; A, B and D wrongly takes a could-have ahead of a should-have.
179. **C** — In requirements refinement, requirements are modified to reflect the components that were discovered, and the specification is redefined around them. Discovery and evaluation is the earlier search for candidates; configuration and adaptation come later, once a choice is made.
180. **D** — In PSP's postmortem, the engineer uses the collected measures (time spent, defects found) to judge how effective the process was and to improve it. Planning produces estimates beforehand, and the design review checks the design for errors.
181. **A** — At level 3, defined, the organisation has standard processes used across projects, tailored to each. Level 2 means individual projects are planned and managed; level 5 focuses on continuous process improvement using quantitative feedback.
182. **B** — Running modelling for one part at the same time as construction for another is a parallel process flow. A linear flow would finish all modelling before any construction starts.
183. **C** — Increment 1 is in use for 9 months (900,000), increment 2 for 6 (600,000), increment 3 for 3 (300,000) and increment 4 for 0. Total: 1,800,000, against nothing from the waterfall by then. This early value is a key benefit of incremental delivery.
184. **D** — Sommerville warns that reuse means the organisation does not control new versions of reused components. The vendor can change or drop features. Lower cost and faster delivery are advantages of reuse, not drawbacks.
185. **A** — Without: 0.25 x 2,000,000 = 500,000. With: 80,000 + 0.05 x 2,000,000 = 180,000. The prototype lowers expected cost by 320,000, so the spiral's risk-reduction step is justified. 100,000 forgets the prototype's cost; 580,000 adds it to the unmitigated exposure.
186. **B** — When a new system replaces an old one, users need all the old functionality before they can switch, so a partial early increment is of little use and hard to get feedback on. The other options describe exactly the conditions where incremental delivery works well.
187. **C** — Sommerville notes that incremental delivery conflicts with procurement models in which a complete specification is part of the development contract, because the specification evolves with each increment. Waterfall and the V-model, which produce a full specification first, fit this contract style.
188. **D** — Each spiral loop begins by identifying and resolving the biggest risks, then chooses a development approach. Incremental development plans increments by business value and priority. A spiral loop need not deliver a working increment to users.
189. **A** — The spiral is risk-driven, the V-model pairs each development phase with a planned test level, and incremental development delivers features in priority order. The wrong pairings shuffle these drivers.

190. **B** — Sommerville says prototypes built to explore requirements should be discarded: it may be impossible to tune them for non-functional requirements, they lack documentation, rapid changes degrade their structure, and quality standards are relaxed during prototyping. A long-lived safety-critical system needs a properly engineered base.
191. **C** — Pressman says the concurrent model applies to all types of software development and defines a network of activities rather than a sequence. Each activity exists concurrently in some state, and events trigger transitions.
192. **D** — Sommerville stresses that RUP separates phases (dynamic perspective) from workflows (static perspective). Any workflow can be active in any phase; for example, elaboration includes some implementation to build an executable architecture baseline.
193. **A** — Expected loss from continuing blindly = 0.6 x 800,000 = 480,000. A 60,000 prototype resolves the uncertainty, and the spiral allows a stop-or-continue decision at the end of each loop. Money already spent is a sunk cost and should not drive the decision.
194. **B** — A mismatch in the data passed between sub-systems is an interface defect. Interfaces are defined in architectural design, which is paired with integration testing. Each unit may work correctly on its own, so unit testing would not reveal it.
195. **C** — The spiral's formal risk analysis adds overhead and needs risk expertise; for a small, low-risk project it is wasted effort. The other recommendations each match a model's strength: traceable test planning, early value delivery and reuse.
196. **D** — Sommerville notes that most practical processes include elements of both plan-driven and agile approaches, and choosing the balance depends on factors such as system size, regulation, lifetime and team skills. Plan-driven models can be iterative, and agile methods do plan.
197. **A** — Sommerville notes that freezing parts of development prematurely can produce a system that does not meet users' needs, and that problems left for later are ignored, programmed around or patched, giving a badly structured system.
198. **B** — The V-model is a waterfall variant: development still runs in sequence, but test planning is tied to each phase, improving early defect detection. It inherits the waterfall's weakness with changing requirements. Risk-driven loops belong to the spiral, and increments to incremental delivery.
199. **C** — Sommerville's integration-and-configuration model trades requirement compromises and loss of control over evolution for lower cost, lower risk and faster delivery. Custom development fits requirements more closely but is usually slower and costlier, and it still needs full testing.
200. **D** — Exposure = probability x loss: A = 100,000, B = 120,000, C = 150,000. So C comes first, then B, then A, exactly the reverse of ranking by probability. Low-probability, high-impact risks can dominate, which is why the spiral uses exposure.
201. **C** — The four values are: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan. The items on the right still have value, but the left is valued more.
202. **C** — Seventeen practitioners wrote the Agile Manifesto at Snowbird, Utah, in February 2001. Scrum (first presented in 1995) and XP (late 1990s) are older than the manifesto itself.
203. **A** — The 2020 Scrum Guide says the Scrum Team consists of one Scrum Master, one Product Owner and Developers, with no sub-teams or hierarchies. Scrum has no project manager accountability.
204. **B** — The 2020 Guide replaced the earlier term 'Development Team' with 'Developers' to stress one Scrum Team with one goal. 'Development Team' is the 2017 wording and a common exam trap.
205. **A** — Sprints are fixed-length events of one month or less. Two weeks is a popular choice, but it is a team decision, not a rule in the Guide.
206. **D** — The Daily Scrum is a 15-minute event for the Developers, held at the same time and place each working day to inspect progress toward the Sprint Goal and adapt the Sprint Backlog.
207. **C** — The Product Owner is accountable for effective Product Backlog management, including ordering items. Others may help, but the Product Owner remains accountable.
208. **D** — The 2020 Guide gives each artifact a commitment: the Product Goal for the Product Backlog, the Sprint Goal for the Sprint Backlog and the Definition of Done for the Increment.
209. **A** — Scrum is founded on empiricism, which rests on transparency, inspection and adaptation. Commitment, focus and courage are among the five Scrum values; visualise and limit come from Kanban.

210. **D** — The Scrum Guide's values are commitment, focus, openness, respect and courage. Communication, simplicity, feedback, courage and respect are XP's values, which is a common mix-up.
211. **C** — Kanban visualises work on a board, sets explicit WIP limits per column and manages flow so work is pulled when capacity frees up. It prescribes no iterations or roles.
212. **D** — In pair programming one developer writes code while the other reviews each line as it is typed, and they switch roles often. Collective ownership means anyone may change any code; a spike is a short experiment to reduce uncertainty.
213. **A** — Kent Beck's XP values are communication, simplicity, feedback and courage, with respect added in the second edition. Planning, testing, coding, listening and designing are XP activities, not values.
214. **C** — A Sprint may be cancelled if its Sprint Goal becomes obsolete, and only the Product Owner has the authority to do so.
215. **D** — One of the twelve principles asks teams to deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.
216. **A** — For a one-month Sprint the maximums are 8 hours for Sprint Planning, 4 hours for the Sprint Review and 3 hours for the Sprint Retrospective. For shorter Sprints these events are usually shorter.
217. **D** — Average velocity = (21 + 25 + 26) / 3 = 72 / 3 = 24. 72 is the total, not the average, and 26 is only the latest Sprint. Velocity is a common forecasting practice, not part of the Scrum Guide.
218. **C** — 150 / 24 = 6.25, and a Sprint cannot be split, so the forecast rounds up to 7 Sprints. Six Sprints would complete only 144 points.
219. **B** — Available person-days = 5 x 10 - 2 = 48. Capacity = 48 x 6 = 288 hours. 300 ignores the leave, and 384 uses 8-hour days instead of the 6 focused hours.
220. **B** — The ideal line burns 120 / 10 = 12 points per day, so after 4 days 120 - 48 = 72 points remain. 48 is the amount burned, not the amount remaining.
221. **D** — Six days are left, so 90 / 6 = 15 points per day. The ideal rate was 12, and the team is 90 - 72 = 18 points behind the ideal line, which is a backlog gap, not a daily rate.
222. **A** — Little's law: average lead time = average WIP / average throughput = 15 / 5 = 3 days. Multiplying gives 75, and inverting gives 0.33.
223. **B** — Only Done work counts: 5 + 8 + 3 + 5 = 21. Partial credit (adding 7.2 for 90% of 8) would give 28.2, and full credit would give 29. The unfinished item returns to the Product Backlog.
224. **A** — The team delivered 24 / 240 = 0.1 points per hour, so 200 x 0.1 = 20 points. Forecasting 24 ignores the reduced capacity.
225. **C** — The Daily Scrum is an event for the Developers, who decide how to turn backlog items into an Increment. The Product Owner and Scrum Master join as Developers only if they are actively working on Sprint Backlog items.
226. **B** — During a Sprint no changes are made that endanger the Sprint Goal and quality does not decrease. New work goes through the Product Owner, who orders the Product Backlog.
227. **D** — WIP limits make Kanban a pull system: Testing pulls work only when it has capacity. Helping clear the bottleneck improves flow, whereas breaking the limit or starting more work increases WIP and lead time.
228. **A** — XP's test-first development (the basis of test-driven development) writes the test before the code, so the test defines the expected behaviour and fails until the code is written.
229. **B** — Continuous integration means integrating each completed task into the whole system as soon as possible, then running all tests. It catches integration problems while they are small.
230. **B** — The Sprint Backlog is made up of the Sprint Goal (why), the selected Product Backlog items (what) and an actionable plan for delivering the Increment (how). It is a plan by and for the Developers.
231. **D** — Story points are relative, team-owned estimates. Turning velocity into a target rewards inflating estimates, so the number rises while real output does not. Velocity is a planning aid, not a productivity measure.
232. **B** — A flat line then a cliff means nothing reached Done until the end. That usually comes from items too large to finish within a few days, which hides risk. Splitting work into smaller items makes progress visible.
233. **C** — The 2020 Guide describes self-managing teams that decide who does what, when and how. The Scrum Master serves the team by coaching, causing impediments to be removed and ensuring events happen, not by acting as a task manager.

234. **B** — The 2020 Guide says that if an item does not meet the Definition of Done it cannot be released or even presented at the Sprint Review; it returns to the Product Backlog for future consideration. Sprints are never extended.
235. **A** — The Guide is explicit: the Product Owner is one person, not a committee. They may represent many stakeholders, and anyone wanting a change in order must convince the Product Owner.
236. **D** — Kanban has no time-boxed iterations: work is pulled as capacity frees up, so urgent tickets can enter immediately while WIP limits stop overload. Fixed Sprints suit planned product work better.
237. **C** — Each team calibrates its own point scale, so a 5 for one team may be a 3 for another. Velocity helps a team forecast its own work; comparing teams with it is a well-known misuse.
238. **A** — Little's law gives WIP = throughput x lead time = 2 x 10 = 20 now. For a 5-day lead time at 2 items per day, WIP must be 2 x 5 = 10. Lowering WIP shortens lead time when throughput is constant.
239. **B** — XP's practices reinforce each other. Refactoring changes structure without changing behaviour, and only a fast automated test suite can confirm behaviour stayed the same. Without tests, refactoring becomes risky.
240. **C** — The Guide names three artifacts. It mentions burn-downs, burn-ups and cumulative flows as practices for forecasting progress, adding that they do not replace the importance of empiricism. They are optional, neither required nor banned.
241. **A** — The four values are individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. The other options reverse or mix up the pairs.
242. **B** — The manifesto states that while there is value in the items on the right, the signatories value the items on the left more. Agile teams still write documentation, plans and contracts; they simply give priority to people, working software, collaboration and change.
243. **C** — The manifesto has four values and twelve supporting principles, such as welcoming changing requirements and maintaining a sustainable pace. Four is the number of values, not principles.
244. **D** — One of the twelve principles says working software is the primary measure of progress. Story points, plan adherence and documentation are proxies that can rise without delivering anything usable.
245. **B** — Sprint Planning is timeboxed to a maximum of eight hours for a one-month Sprint, and is usually shorter for shorter Sprints. Four hours is the Sprint Review limit, three hours the Retrospective limit and 15 minutes the Daily Scrum.
246. **C** — The Sprint Review is timeboxed to a maximum of four hours for a one-month Sprint. Eight hours belongs to Sprint Planning; the Guide sets no two-hour or one-day limit for the Review.
247. **D** — The 2020 Guide frames Sprint Planning around three topics: why the Sprint is valuable (leading to the Sprint Goal), what can be Done, and how the work will get done. The 'yesterday/today/blockers' list is the old Daily Scrum format, and 'went well/badly' is a Retrospective pattern.
248. **A** — The Guide calls the Sprint Review a working session, not just a presentation. The Scrum Team shows results to key stakeholders, progress toward the Product Goal is discussed, and the Product Backlog may be adjusted. It is held every Sprint, release or not.
249. **C** — The Product Goal describes a future state of the product and is the commitment for the Product Backlog. The team works toward one Product Goal at a time, fulfilling or abandoning it before taking on the next. The Sprint Goal is set each Sprint, and the Definition of Done is the Increment's commitment.
250. **D** — If the Definition of Done is part of the organisation's standards, all Scrum Teams must follow it as a minimum; teams may add stricter criteria. Only if no organisational standard exists must the Scrum Team create its own. Nobody can waive it, because work that misses it is not part of the Increment.
251. **A** — Refinement is an ongoing activity, not a formal Scrum event. It breaks items into smaller, more precise ones and adds details such as description, order and size, so that items are ready for selection at Sprint Planning.
252. **B** — The Guide describes the Sprint Backlog as a plan by and for the Developers. It is updated throughout the Sprint as more is learned. The Product Owner orders the Product Backlog, not the Sprint Backlog, and the Scrum Master does not manage it.
253. **D** — The Scrum Master is accountable for establishing Scrum and for the Scrum Team's effectiveness. Maximising product value is the Product Owner's accountability, and creating a usable Increment belongs to the Developers.

254. **A** — During Sprint Planning the Product Owner proposes how the product could increase its value and utility in the Sprint, then the whole Scrum Team collaborates to define the Sprint Goal. It must be finalised before Sprint Planning ends.
255. **B** — The Guide says multiple Increments may be created within a Sprint and the Sprint Review should never be treated as a gate to releasing value. Work becomes part of an Increment only when it meets the Definition of Done.
256. **C** — The Guide says a Scrum Team is small enough to remain nimble and large enough to complete significant work, typically 10 or fewer people. The older '7 plus or minus 2' guidance was removed; there is no exact fixed number.
257. **A** — The 2020 Guide states that Scrum is founded on empiricism (knowledge comes from experience and decisions based on what is observed) and lean thinking (reduce waste and focus on essentials). Pair programming and TDD are XP practices.
258. **B** — The Sprint is the container: Sprint Planning, the Daily Scrums, the Sprint Review and the Sprint Retrospective all happen inside it. A new Sprint starts immediately after the previous one ends.
259. **C** — In a pull system, downstream stages pull work when they have free capacity, and WIP limits make that capacity visible. Pushing work regardless of capacity creates queues and longer lead times.
260. **D** — A cumulative flow diagram plots cumulative counts of items in each state over time. The vertical gap between arrivals and departures is the work in progress; the horizontal gap approximates lead time.
261. **B** — Under collective ownership, the whole team owns all of the code, so any pair can improve any part. This spreads knowledge and avoids bottlenecks around individual owners; it relies on tests and continuous integration to stay safe.
262. **C** — A spike is a small throwaway investigation, such as a quick prototype of an unfamiliar library, used to reduce uncertainty before estimating or designing a story. It is timeboxed so that it does not grow into full development.
263. **D** — Sustainable pace (originally the '40-hour week') limits overtime so that the team can keep a steady, healthy speed. The Agile principles also say teams should be able to maintain a constant pace indefinitely.
264. **A** — INVEST stands for Independent, Negotiable, Valuable, Estimable, Small and Testable. An independent story can be developed and delivered without depending on other stories, which makes ordering the backlog easier.
265. **C** — At the best velocity, 100 / 22 = 4.5, so 5 Sprints. At the worst, 100 / 18 = 5.6, so 6 Sprints. A range is more honest than a single number because velocity varies.
266. **D** — Wide spreads usually mean people understand the story differently. In planning poker the outliers explain their thinking, the team discusses, and then re-estimates until the estimates converge. Averaging hides the misunderstanding.
267. **A** — Vertical slices cut through every layer and deliver something usable and testable, so each can meet the Definition of Done. Splitting by layer or by activity produces pieces with no standalone value, and Sprints are never extended.
268. **B** — The Retrospective inspects how the last Sprint went with regard to individuals, interactions, processes, tools and the Definition of Done, and plans ways to increase quality and effectiveness. The Review inspects the product outcome with stakeholders.
269. **D** — The Guide allows scope to be clarified and renegotiated with the Product Owner as more is learned, as long as the Sprint Goal is not endangered and quality does not decrease. Sprint length is fixed, and cancelling is reserved for an obsolete Sprint Goal.
270. **A** — The 2020 Guide removed the three questions. The Developers can select whatever structure and techniques they want, as long as the Daily Scrum inspects progress toward the Sprint Goal and produces an actionable plan for the next day of work.
271. **B** — The Guide says that if the Product Owner or Scrum Master are actively working on items in the Sprint Backlog, they participate as Developers. The Daily Scrum is otherwise an event for the Developers.
272. **C** — After n Sprints, done = 120 + 20n and scope = 200 + 5n. Done catches scope when 15n >= 80, so n >= 5.33, which rounds up to 6. Ignoring scope growth gives 80 / 20 = 4, and rounding 5.33 down gives 5.
273. **A** — Little's law says average WIP = throughput x lead time, so throughput = WIP / lead time = 24 / 8 = 3 items per day. Multiplying gives 192, and inverting gives 0.33.

274. **B** — Cycle times are 3, 6, 3 and 6 days, totalling 18, so the average is 18 / 4 = 4.5 days. 3 and 6 are single items, and 18 is the total, not the average.
275. **C** — In pair programming the driver controls the keyboard and the navigator reviews the code as it is written and thinks ahead. Pairs swap roles often so both stay engaged.
276. **D** — Refactoring improves the internal structure of code without changing its external behaviour, and the passing tests confirm the behaviour is unchanged. Debugging fixes wrong behaviour, which is not the case here.
277. **B** — XP puts a customer representative with the team, so questions about requirements and priorities are answered in minutes rather than days. The other practices are engineering practices that do not supply business knowledge.
278. **C** — The Given/When/Then format comes from behaviour-driven development and the Gherkin language. It states a starting context, an action and an expected outcome, which can be turned directly into automated acceptance tests.
279. **D** — WSJF scores are A = 20/10 = 2, B = 12/3 = 4, C = 30/15 = 2 and D = 8/1 = 8. D has the highest score, so it delivers the most value per unit of effort. C has the largest cost of delay but is also the largest job.
280. **A** — The Scrum Master serves the organisation as well as the team, including leading, training and coaching it in its Scrum adoption and helping employees and stakeholders understand an empirical approach.
281. **C** — The Guide says the Product Backlog may be adjusted at the Sprint Review to meet new opportunities. The Product Owner then orders the new item against the rest for future Sprints.
282. **D** — The Guide says an Increment may be delivered to stakeholders before the end of the Sprint and that the Sprint Review should never be considered a gate to releasing value. The work must still meet the Definition of Done.
283. **A** — The Guide says shorter Sprints generate more learning cycles and limit the risk of cost and effort to a smaller time frame. With fast-changing needs, more frequent inspection and adaptation reduces wasted work.
284. **B** — 30 / 4 = 7.5 weeks, which rounds up to 8 because the last items still need a week to finish. Rounding down to 7 under-forecasts, and 120 comes from multiplying instead of dividing.
285. **D** — XP's simple design says build the simplest thing that meets today's stories, and YAGNI warns against building for guessed future needs. If more formats are needed later, refactoring with tests in place makes adding them safe.
286. **A** — The 2020 Guide says that within a Scrum Team there are no sub-teams or hierarchies; the team is cross-functional and works together toward the Product Goal. Specialist sub-teams create hand-offs that slow the flow of Done work.
287. **B** — The Daily Scrum is a 15-minute event to inspect progress toward the Sprint Goal and adapt the plan. The Guide notes that Developers often meet throughout the day for detailed discussions, so problem-solving belongs in a follow-up with only the people who need to be there.
288. **C** — The Guide says the team identifies the most helpful changes to improve its effectiveness and addresses the most impactful ones as soon as possible; they may even be added to the Sprint Backlog for the next Sprint. Without a concrete, owned action, inspection never becomes adaptation.
289. **A** — Each Sprint should produce a usable Increment that meets the Definition of Done. Needing a separate testing Sprint means untested, unintegrated work was counted as done, which hides risk and delays feedback. The fix is a stronger Definition of Done and integrating and testing inside each Sprint.
290. **B** — Agile approaches usually fix time and cost through timeboxes and a stable team, and treat scope as the variable, delivering the most valuable items first. Quality should never be the variable, and overtime breaks sustainable pace.
291. **C** — A widening band means items enter a stage faster than they leave, so work piles up there; a flattening Done line confirms throughput is falling. Limiting WIP before testing and swarming on test work restores flow.
292. **D** — The Guide makes the Product Owner accountable for clearly communicating Product Backlog items. They may delegate this work to others but remain accountable. Here no one could clarify items, so the team built the wrong thing.
293. **B** — A burndown shows only the remaining work, so scope added at the same rate as completed work keeps the line flat. A burnup plots completed work and total scope separately, making both progress and scope growth visible.

294. **C** — By Little's law, at the same throughput more WIP means longer lead times. WIP limits turn the board into a pull system and force the team to finish work before starting new work.
295. **D** — The Guide says that when several Scrum Teams work on the same product, they should share the same Product Goal, Product Backlog and Product Owner. One ordered backlog removes conflicting priorities.
296. **A** — Sommerville notes that plan-driven approaches, or hybrids that add plan-driven elements to agile practices, suit systems that need detailed analysis before implementation or must be approved by an external regulator. Agile practices can still be used inside that framework.
297. **C** — Sprints are fixed-length events of one month or less, and a new Sprint starts immediately after the previous one ends. Keeping the cadence preserves regular inspection; unfinished items go back to the Product Backlog, where the Product Owner decides their order.
298. **D** — Studies of pair programming report that total effort rises only modestly while defects drop and knowledge spreads across the team. Measuring only individual output ignores rework and the cost of defects found later.
299. **A** — Lead time = WIP / throughput. Team A: 30 / 5 = 6 days. Team B: 12 / 3 = 4 days. Higher throughput does not guarantee shorter lead time when WIP is much higher.
300. **B** — The Sprint Goal is the single objective for the Sprint. It creates coherence and focus, and gives the Developers flexibility over the exact work needed to achieve it. With unrelated items, nothing guides which work to cut or change when plans shift.
301. **B** — A functional requirement states a service the system must provide or how it reacts to a particular input. Downloading a fee voucher is a service. The others are a performance constraint, an availability constraint and an organisational (implementation) constraint, which are all non-functional.
302. **A** — Non-functional requirements are constraints on the services or functions offered by the system, including timing, development-process and standards constraints. They often apply to the system as a whole (emergent properties such as reliability) rather than to one feature. Specific services are functional requirements; module order and class structure are design or planning decisions.
303. **B** — Sommerville's NFR classification has three branches: product requirements (behaviour of the product, e.g. speed, reliability), organisational requirements (policies and procedures, e.g. process standards, delivery) and external requirements (factors outside the system and its development, e.g. legislation and ethics). Usability, efficiency and dependability are sub-types of product requirements.
304. **C** — External requirements come from factors outside the system and its development process, such as regulatory, legislative and ethical requirements. Compliance with data protection law is a legislative requirement, so it is external. Product requirements describe the product's own behaviour, and organisational requirements come from the customer's or developer's own policies.
305. **B** — Sommerville separates the two by audience and detail. User requirements are statements in natural language, plus diagrams, of the services the system provides and its constraints, written for customers and managers. System requirements expand them into a detailed specification of functions, services and constraints, often used as a basis for the contract and for developers. Both levels contain functional and non-functional requirements.
306. **B** — Ethnography is an observational technique: the analyst immerses themselves in the workplace and watches how work is really done. It is good at revealing implicit practices and social factors that people do not mention in interviews. Questionnaires and JAD workshops rely on what people say; prototyping relies on their reaction to a model.
307. **A** — Interviews are good for getting an overall picture of what stakeholders do and want, but people often find it hard to describe domain knowledge they use every day, and specialist terminology can be misunderstood. That is why interviews are usually combined with observation (ethnography), scenarios or prototypes. Interviews can capture both functional and non-functional requirements and need no prototype.
308. **A** — SRS stands for Software Requirements Specification. It is the official statement of what the system developers should implement, covering both user and system requirements. IEEE 830 (now superseded by ISO/IEC/IEEE 29148) gives a recommended structure for it.
309. **C** — IEEE 830 defines a requirement as verifiable when there is some finite, cost-effective process by which a person or machine can check that the software meets it. Vague terms such as 'fast' or 'user-friendly' fail this test. One interpretation only is 'unambiguous', recorded origins is 'traceable', and easy change is 'modifiable'.

310. **A** — Traceability links requirements to their sources (stakeholders, regulations) and to the artefacts derived from them (design elements, code, test cases). It supports impact analysis when something changes and lets testers show that every requirement is covered. Tracking programmers, meetings or effort are project-management activities, not traceability.
311. **A** — You first discover requirements from stakeholders (elicitation and analysis), then write them down (specification), then check that they describe the system the customer wants (validation). Sommerville shows these as an iterative spiral, with requirements management running alongside. Older editions also put a feasibility study first.
312. **D** — Validation asks 'are we building the right product?'. It checks the requirements document for validity, consistency, completeness, realism and verifiability, using reviews, prototyping and test-case generation. Checking code against design is verification, collecting requirements is elicitation, and estimating cost is project planning.
313. **D** — Domain requirements come from the application domain rather than from individual users, for example a railway braking system's deceleration formula or a banking rule about interest calculation. They can be functional or non-functional. They are a risk because domain experts treat them as obvious and software engineers may not understand them, so they are often left out or misunderstood.
314. **C** — Requirements management distinguishes enduring requirements, which stay stable because they relate to the core business (a hospital will always have patients and doctors), from volatile requirements, which change as the environment, policies or customer understanding change. Planning for volatile requirements reduces rework. Neither kind is tied to functional or non-functional status.
315. **D** — MoSCoW sorts requirements into Must have (essential for the release), Should have (important but not vital), Could have (desirable if time allows) and Won't have this time (agreed to be out of scope for now). It makes scope negotiation explicit, which is useful in time-boxed and agile projects.
316. **A** — «extend» models optional or conditional behaviour: the extending use case (B) is inserted into the base use case (A) at an extension point when its condition is true, and A is complete without it. «include» is the mandatory one: the included behaviour always runs as part of the base. The arrow for «extend» points from the extension to the base.
317. **D** — The statement does not add a new service; it constrains how fast an existing service (search) must respond. That makes it a non-functional product requirement of the efficiency/performance type. It is also measurable, so it is verifiable.
318. **D** — Sommerville's realism check asks whether the requirements can actually be implemented with available technology, budget and schedule. A consistency check looks for conflicting requirements, a completeness check looks for missing functions or constraints, and a validity check asks whether the system provides the functions that best support the customer's needs.
319. **D** — Allowed unavailability is 0.1%. A 30-day month has 30 × 24 × 60 = 43,200 minutes, and 0.001 × 43,200 = 43.2 minutes. 8.76 hours is the yearly figure for 99.9% (0.001 × 8,760 h), and the other two options slip the decimal point.
320. **B** — A verifiable NFR names a measurable criterion that a test can check, here training time and error rate (Sommerville uses exactly this style of usability metric). 'Very user-friendly' and 'satisfied' remain subjective, and naming a framework is a design decision that says nothing testable about usability.
321. **C** — When users cannot articulate needs for something new, a prototype gives them something concrete to react to, which surfaces requirements and misunderstandings quickly. A questionnaire and a signed SRS both depend on users already knowing what they want, and policy documents describe rules rather than how people will use a new interface.
322. **A** — Questionnaires scale to large, dispersed groups and give quantitative data that can be analysed. Interviewing or observing 5,000 people is not feasible, and a workshop works for a small group of representatives, not thousands of participants.
323. **C** — 'Quickly' gives no measurable criterion, so there is no finite, cost-effective way to test whether the software meets it. It therefore fails the verifiable property. A fix is a measured statement such as 'within 2 seconds for 95% of actions'. Traceability, modifiability and ranking are about references, document structure and priority.
324. **D** — Two requirements that cannot both be satisfied make the specification inconsistent. Each one is measurable, so they are verifiable, and nothing is missing. A consistency check during validation is meant to catch exactly this kind of conflict.

325. **C** — Organisational requirements come from the policies and procedures of the customer's or developer's organisation, including development-process standards and delivery dates. It does not describe the product's behaviour (product requirement) or factors from outside, such as laws (external requirement).
326. **C** — POFOD (probability of failure on demand) is the probability that a request for service fails. A POFOD of 0.001 means 1 failure per 1,000 demands. Failures per unit of time is ROCOF (rate of occurrence of failures), recovery time relates to MTTR, and percentage up-time is availability.
327. **B** — ROCOF is a rate per unit of operating time. One month of continuous running is 30 × 24 = 720 hours, so expected failures = 2 / 1,000 × 720 = 1.44. 2 ignores the length of the period, 14.4 is a tenfold slip, and 0.06 uses days (30) instead of hours.
328. **A** — The common user-story template is 'As a <role>, I want <goal>, so that <benefit>'. The 'so that' clause gives the reason or value, which helps the team prioritise and find better solutions. 'As a student' is the role and 'I want to reset my password' is the goal.
329. **D** — Forward traceability from requirements to tests shows coverage. A requirement with no linked test is a coverage gap: nobody can show it is met. The fix is to add a test, not delete the requirement. Missing tests do not say anything about whether R7 is functional or non-functional.
330. **B** — An SRS states what the system must do, not how it will be built. Sending a reminder 24 hours ahead is observable behaviour. The Singleton class, the hash map and the cron job are all implementation or design choices that belong in the design.
331. **D** — The software meets its specification, so verification succeeded; the problem is that the specification itself was wrong. Checking that requirements reflect real needs is validation ('are we building the right product?'), helped by reviews with users, prototypes and observing real work.
332. **C** — 'Secure' cannot be verified as it stands. Good practice is to decompose a security goal into concrete requirements, some functional (account lockout, audit logging) and some measurable constraints (encryption standard). These can be traced and tested. Leaving interpretation to testers hides a decision, and security requirements do belong in the SRS.
333. **B** — Every later artefact depends on the requirement. Fixing it late means redesigning, recoding, retesting, updating documents and redeploying to users, so cost multiplies with each phase it passes through. That is why early validation through reviews and prototypes pays off.
334. **A** — The correct option names one capability with measurable limits (10 seconds, 2,000 students), so it can be tested. The others are vague, bundle three needs into one statement, or prescribe a design solution instead of a requirement.
335. **D** — Conflicting requirements are resolved through negotiation and prioritisation with stakeholders, and the agreed decision is documented so it can be traced. A session-based 2FA keeps the security goal while meeting the response target for each record. Neither NFR automatically outranks the other.
336. **B** — Behaviour that always runs is modelled with «include», with the arrow from the base (Withdraw Cash) to the included use case (Authenticate PIN). Optional, condition-dependent behaviour is «extend», with the arrow from the extension (Print Receipt) to the base. Mixing them up is a common exam trap.
337. **C** — With highly volatile requirements, a frozen up-front specification goes out of date quickly and causes expensive change requests. An iterative approach keeps requirements visible, prioritised and regularly validated, which is the agile RE practice Sommerville and Pressman both describe. Iterative still means managing requirements, not dropping them.
338. **C** — Impact analysis starts at the source (the regulation), finds the requirements derived from it (source traceability), then follows forward traceability to the design elements, code modules and tests that implement and check them. The other records may help, but none links the regulation to the affected artefacts.
339. **A** — Sommerville notes that non-functional requirements often generate functional ones. Logging each attempt is a specific behaviour the system performs, and you can test it by attempting a login and checking the log, so it is functional. The broad goal 'the system shall be secure' is the non-functional part.
340. **B** — X: 999 / 1,000 = 99.90%. Y: 2,000 / 2,004 = 99.80%. Z: 400 / 400.25 = 99.94%. So X and Z meet the target and Y does not. A longer MTTF alone is not enough; fast repair (low MTTR) matters just as much.
341. **A** — Sommerville describes an iterative cycle: discover requirements by interacting with stakeholders, classify and organise related requirements, prioritise them and negotiate conflicts, then document them. The cycle repeats as understanding grows.

342. **B** — Stakeholders include end users, managers, customers, regulators, maintainers, domain experts and anyone else the system affects or who influences its requirements. Talking only to payers or users misses important requirements.
343. **C** — Sommerville lists these parts of a scenario: what the system and users expect at the start, the normal flow, what can go wrong and how it is handled, other activities happening at the same time, and the system state at the end. Scenarios are concrete stories that stakeholders find easy to discuss.
344. **D** — An actor is outside the system boundary. It can be a human role such as 'Student' or another system such as a payment gateway. Internal classes and developers are not actors.
345. **B** — IEEE 830 asks that each requirement be marked with its importance (such as essential, conditional or optional) and/or its stability (how likely it is to change). This helps planning, trade-offs and change management.
346. **C** — A modifiable SRS has a coherent organisation, such as a table of contents and an index, and avoids stating the same requirement in several places. Redundancy causes inconsistencies when one copy is changed and the other is not.
347. **D** — IEEE 830's completeness covers all significant requirements, responses to all realisable classes of input in both valid and invalid situations, full labels and references, and definitions of all terms. Budgets, staffing and code modules are outside an SRS.
348. **A** — ISO/IEC/IEEE 29148 (first published in 2011) replaced IEEE 830 and covers requirements engineering across the life cycle. ISO 9001 is quality management, IEEE 1012 covers verification and validation, and ISO/IEC 25010 is a product quality model.
349. **C** — Sommerville's process first analyses the problem or change proposal, then assesses its effect using traceability and estimates its cost, and only then implements it by changing the requirements document and, as needed, the design and code.
350. **D** — Pressman lists seven tasks: inception, elicitation, elaboration, negotiation, specification, validation and management. Inception establishes a basic understanding of the problem, the stakeholders and the nature of the solution.
351. **A** — Kano classifies basic needs (expected, so missing them angers users), performance needs (more is better) and excitement needs (unexpected delighters). The last option describes an indifferent attribute.
352. **B** — A feasibility study is a short early study that checks whether the system would support the organisation's goals and whether it can be built within the technology, budget and time available. It happens before detailed specification and design.
353. **D** — By convention, 'shall' marks a mandatory requirement, while 'should' marks a desirable but optional one. Using the words consistently is part of writing unambiguous requirements.
354. **A** — Amalgamation means several requirements are bundled into a single statement, which makes them hard to test and trace. Requirements confusion is the mixing of functional and non-functional requirements or goals; contradiction is inconsistency.
355. **B** — Sommerville's structured specification describes each function with its inputs and their source, outputs and their destination, information needed for the computation, the action, pre- and post-conditions and side effects. Design and project details belong elsewhere.
356. **C** — Sommerville lists requirements reviews, prototyping and test-case generation as validation techniques. If writing a test for a requirement is hard, the requirement is probably vague or unverifiable. The other options relate to code or operations.
357. **A** — Tacit knowledge is so familiar to stakeholders that they find it hard to explain or forget it entirely. It is one reason interviews miss requirements, and observation or prototypes help surface it.
358. **B** — Non-functional requirements often apply to the system as a whole. Users can sometimes work around a missing function, but if a system is unreliable or too slow it may be rejected entirely.
359. **C** — Scope creep happens when new requirements keep being added without formal change control, so cost and schedule grow. Requirements management, including a baseline and a change process, keeps it under control.
360. **D** — A goal states a general intention such as ease of use. A verifiable requirement uses a measure that can be objectively tested, like training time, response time or availability.

361. **B** — Sommerville's table measures robustness by time to restart after failure, percentage of events causing failure and probability of data corruption. Transactions per second measures speed, training time measures ease of use and target-dependent statements measure portability.
362. **C** — A user story ('As a ... I want ... so that ...') is a brief placeholder for a requirement, refined through conversation and acceptance criteria. A use case documents the main and alternative flows of an interaction with the system.
363. **D** — Once requirements are reviewed and approved they are baselined. Later changes go through change control, which makes their cost and impact visible and helps prevent scope creep.
364. **A** — Formal specification uses mathematics to define behaviour precisely, which supports rigorous analysis and is mostly used for safety- or security-critical parts. It is harder for customers to read, which is why it is usually combined with natural language.
365. **C** — Sommerville places ethical requirements under external requirements, because they come from outside the system and its developing organisation and ensure the system is acceptable to users and the public. Here the constraint protects patient privacy.
366. **D** — A limit on how much storage the software itself uses constrains the product's behaviour, so it is a product requirement in the efficiency or space category. It does not describe a service the system provides.
367. **A** — Sommerville gives this exact kind of example as an operational requirement, which is a type of organisational requirement: it reflects how the organisation operates and the procedures users must follow. It is not mandated by law or derived from the medical domain.
368. **B** — Must-have items are those without which the release would be unworkable, unsafe or illegal. A legal requirement cannot be traded away, so it is a Must have.
369. **D** — Delighters are unexpected features that raise satisfaction when present but cause no dissatisfaction when absent, because users did not expect them. Over time, delighters often become basic expectations.
370. **A** — Set MTTF / (MTTF + 2) = 0.999. Then MTTF = 0.999 x 2 / 0.001 = 1,998 hours. 2,000 hours would also meet the target but is not the minimum, and 999 hours only works when MTTR is 1 hour.
371. **B** — Probability of failure on demand = failures / demands = 4 / 2,000 = 0.002. 0.0005 and 500 come from dividing the other way round.
372. **C** — Sommerville recommends POFOD for systems whose services are demanded at unpredictable, relatively long intervals, where a failure on demand has serious consequences, such as protection systems. ROCOF suits systems with frequent, regular demands.
373. **A** — Coverage = 34 / 40 = 0.85, or 85%. The 6 requirements without tests (15%) need test cases before the system can be shown to meet them.
374. **B** — A + B + D costs 5 + 3 + 2 = 10 and gives 9 + 7 + 4 = 20 value. A + C gives 15, B + C + D gives 17 and A + B gives 16. Checking every combination within budget confirms 20 is the maximum.
375. **C** — When the existing documents are the main source of knowledge, studying them reveals business rules, data items and reports the new system must support. The other techniques depend on people who are no longer available.
376. **D** — Joint application development workshops bring stakeholders together with a facilitator to discuss conflicts and agree requirements in one place. Separate interviews keep the conflict hidden, and students are not the source of the departmental rules.
377. **B** — 'The last month' could mean the previous calendar month or the last 30 days, so two readers could build different systems. The other options define the period or the limit precisely.
378. **C** — The first statement bundles two different needs, emailing and ledger updating, and it is unclear whether the 5-second limit applies to both. Each need should be a separate requirement so it can be tested and traced on its own.
379. **D** — Sommerville's consistency check looks for conflicts or contradictions between requirements. Completeness asks whether all required functions and constraints are included, realism whether they can be built, and verifiability whether they can be tested.
380. **A** — Actors are external roles that interact with the system, and they can be other systems as well as people. The gateway sits outside the boundary and is associated with the 'Pay fee' use case.
381. **C** — A use case description has a main success scenario and alternative or exception flows for deviations such as a rejected card. Mixing failures into the main flow makes it hard to read and test.

382. **D** — A user story should deliver value to a user or customer. A technical task like setting up a schema has no value on its own and is better treated as a task inside a user-facing story.
383. **A** — 28,800 / 60 = 480 transactions per second, which is below the required 500, so the requirement is not met. A measurable requirement lets this be decided objectively.
384. **B** — Following the forward links gives T1, T2, T3 from D1 and T3, T4 from D2. T3 is shared, so the distinct set is T1, T2, T3 and T4, which is 4 tests. Counting T3 twice gives 5.
385. **D** — Volatility = changed / baselined = 30 / 120 = 25% in a single quarter, which is high. It signals unstable requirements that need firm change control and benefit from iterative development with frequent validation.
386. **A** — No real system can guarantee zero downtime, and a requirement that can never be shown to be met fails realism and verifiability. A measurable availability target with defined maintenance windows can be designed for and checked.
387. **B** — Each stakeholder group holds part of the requirements. Managers may not know the details of daily nursing work, so leaving out the real users produced a system that does not fit their practice.
388. **C** — IEEE 830 allows TBDs only with a note on why the information is missing, who will resolve it and by when. Designing on unknowns risks rework, but work that does not depend on the TBDs can usually continue.
389. **A** — Throwaway prototypes are built to clarify requirements, so they cut corners on structure, error handling, security and performance. Shipping one carries those weaknesses into production. The lessons from it should still guide the real system.
390. **B** — Every piece of code should trace back to a requirement. Features with no requirement link are gold plating: they add cost, risk and test effort without agreed value.
391. **C** — Ethnographic observation reveals how work is really done, including workarounds people do not mention. Requirements must decide how the system supports both the rule and the real pressure, rather than assume the official procedure is followed.
392. **D** — An SRS should normally say what, not how, but genuine constraints imposed by stakeholders, contracts or existing systems are valid design constraints. Without such a reason, naming a technology is premature design.
393. **B** — MoSCoW works only if some items can be dropped when time runs short. DSDM guidance keeps Must haves to about 60% of the effort, leaving Should and Could haves as contingency. With everything a Must, any slip means failure.
394. **C** — A use case diagram is an overview. Detailed use case descriptions are needed for the flows and rules, and non-functional requirements such as security and performance must be specified separately.
395. **D** — Sommerville separates user requirements, written in natural language for customers and users, from detailed system requirements for developers. Stakeholders can only validate requirements they can understand.
396. **A** — Sommerville notes that domain requirements are written in the language of the domain and experts may leave out information they think is obvious. Analysts must probe for such implicit domain rules.
397. **C** — Non-functional requirements apply across stories and are easily lost if only user stories are written. Making them explicit and measurable, for example 'p95 response under 2 s at 3,000 users', ensures they are designed and tested.
398. **D** — Missing forward links from R1 suggest it was never designed or built. R3 went to code without a documented design. An orphan test such as T7 points to an obsolete requirement or an unrequested feature.
399. **A** — When the client and developer are different organisations on a fixed price, the requirements document becomes the basis of the contract and of acceptance. Sommerville notes that detailed specification is needed when outsourcing development.
400. **B** — Questionnaire results are only as good as the sample and the wording. An unbalanced sample and leading questions both distort the findings. Follow up with balanced sampling, neutral wording, or interviews at the smaller campus.
401. **B** — Pressman's management spectrum is the four Ps: people (the most important factor), product (the software to be built and its scope), process (the framework of activities) and project (the planning and control of the work). 'Price, place and promotion' come from the marketing mix, not software engineering.
402. **B** — COCOMO is Barry Boehm's Constructive Cost Model, published in 1981. It is an algorithmic cost model: it estimates effort and development time from the size of the software (in KLOC, thousands of lines of code) using constants fitted to historical project data.

403. **D** — The three basic COCOMO modes are organic (small teams, familiar problems), semi-detached (mixed experience, medium size) and embedded (tight hardware, software and operational constraints). 'Simple, average and complex' are function-point weights, and application composition, early design and post-architecture are COCOMO II sub-models.

404. **A** — Organic mode means a relatively small team working in a familiar environment on a well-understood problem with flexible requirements. Flight-control and air-traffic systems with tight constraints are embedded mode; a mixed-experience team with some rigid requirements describes semi-detached mode.

405. **B** — Basic COCOMO gives effort E = a(KLOC)^b in person-months, the work one person does in one month. Development time TDEV = c(E)^d is in calendar months, and average staffing is E / TDEV. Function points are a size measure, not an effort unit.

406. **B** — Fred Brooks wrote in The Mythical Man-Month that adding manpower to a late software project makes it later. New people need training from the existing team, and communication paths grow as n(n−1)/2. 'Work expands to fill the time available' is Parkinson's law.

407. **B** — A Gantt chart (a bar chart) lists activities down the side and time along the top, with a bar for each activity's start and finish. It is easy to read for progress tracking but shows dependencies less clearly than an activity network (PERT/CPM chart).

408. **B** — The critical path is the longest-duration path from start to finish. Its activities have zero total float, so any delay to one of them delays the whole project. The number of activities and their cost do not decide it; total duration does.

409. **A** — PERT (Program Evaluation and Review Technique) handles uncertainty with three estimates per activity: optimistic (o), most likely (m) and pessimistic (p). It combines them using a beta distribution as te = (o + 4m + p) / 6. Earliest and latest start belong to CPM calculations, and planned, earned and actual are earned-value terms.

410. **A** — Total float is how long an activity can be delayed without delaying the project: LS − ES, which equals LF − EF. Earliest finish minus earliest start is just the duration. LF − ES adds the duration back in and overstates the float.

411. **B** — A WBS breaks the total scope of the project into manageable pieces, level by level, down to work packages that can be estimated, scheduled and assigned. It is the basis for estimation and scheduling. An organisation chart shows people, not work.

412. **D** — EV (budgeted cost of work performed) measures progress in money terms: the planned cost of the work that has actually been done. Planned value (PV) is the budgeted cost of the work scheduled by now, and actual cost (AC) is what has really been spent. CPI = EV / AC and SPI = EV / PV.

413. **D** — Wideband Delphi is a form of expert judgement. Independent estimates avoid anchoring on one loud voice, and the discussion rounds surface assumptions. A formula from LOC is algorithmic cost modelling (such as COCOMO), and using a past project is estimation by analogy, which still needs adjusting for differences.

414. **C** — COCOMO II has sub-models for different stages. Application composition suits prototyping with reusable components (object points); early design is used after requirements, before architecture; post-architecture is used once the architecture is known and gives the most detailed estimate. 'Basic organic' is a mode of the original 1981 COCOMO.

415. **C** — This is often shown as the 'cone of uncertainty': early on, scope, technology and team are not fully known, so estimates can be off by a large factor. As requirements and design firm up, the range narrows. That is why estimates should be given as ranges and revised at milestones.

416. **D** — Sommerville defines milestones as points in the schedule against which progress can be assessed, such as the handover of a system for testing. Deliverables are work products delivered to the customer, such as a requirements document or a release. Every deliverable usually marks a milestone, but not every milestone produces a deliverable.

417. **B** — E = 2.4 × 32^1.05 ≈ 91.33 person-months. 76.8 drops the exponent (2.4 × 32), 145.5 uses the semi-detached constants (3.0 × 32^1.12), and 13.9 months is the development time TDEV, not the effort.

418. **D** — TDEV = 2.5 × 91.33^0.38 ≈ 13.9 months. Average staffing = E / TDEV = 91.33 / 13.9 ≈ 6.57, so about 6.6 people. 13.9 is the duration itself, 7.6 divides effort by 12 months, and 36.5 divides effort by 2.5.

419. **C** — E = 3.0 × 50^1.12 ≈ 239.87 person-months. 150 forgets the exponent, 145.9 uses the organic constants (2.4 × 50^1.05), and 17.0 months is the semi-detached development time (2.5 × 239.87^0.35).

420. **C** — te = (o + 4m + p) / 6 = (2 + 16 + 12) / 6 = 30 / 6 = 5 days. 6 is the simple average of the three numbers, and 4 is just the most likely value. The long pessimistic tail pulls the expected time above the most likely time.
421. **C** — PERT standard deviation σ = (p − o) / 6 = (15 − 3) / 6 = 2 weeks, so variance $\sigma^2 = 4$. Choosing 2 confuses the standard deviation with the variance, and 12 is just p − o.
422. **D** — Paths: A-B-D-F = 3 + 4 + 5 + 2 = 14 and A-C-E-F = 3 + 2 + 6 + 2 = 13. The longest path, A-B-D-F, is the critical path, so the project needs 14 days. 22 is the sum of all durations, which ignores that B and C run in parallel.
423. **A** — Forward pass: E starts at 5 (after A and C) and finishes at 11. F cannot start until D finishes at 12, so E's latest finish is 12 and its latest start is 6. Float = LS − ES = 6 − 5 = 1 day. Equivalently, path A-C-E-F (13) is one day shorter than the critical path (14).
424. **D** — CPI = EV / AC = 40,000 / 50,000 = 0.8. A CPI below 1 means you are getting 0.80 of value for every unit spent, so the project is over budget. 1.25 inverts the ratio, and 1.0 wrongly uses PV / AC.
425. **D** — SPI = EV / PV = 60 / 80 = 0.75, so only 75% of the work planned by now has been done; the project is behind schedule. 0.80 is the CPI (60 / 75), 1.33 inverts SPI, and 1.07 is PV / AC.
426. **C** — EAC = BAC / CPI = 200,000 / 0.8 = 250,000. Multiplying instead of dividing gives 160,000, which wrongly predicts savings. 200,000 assumes the overspend stops, which is not what 'current performance continues' means.
427. **C** — Effort = size / productivity = 12,000 / 400 = 30 person-months. 48 comes from multiplying 12 by 4, and 3 and 300 are tenfold slips. This simple LOC-based approach relies on productivity data from similar past projects.
428. **A** — Variances add along a path, standard deviations do not. Variances: $(10/6)^2 = 2.78$, $(12/6)^2 = 4$ and $(4/6)^2 = 0.44$, total 7.22. Path σ = √7.22 ≈ 2.69 days. 4.33 adds the three standard deviations, and 7.22 is the variance itself.
429. **B** — Earliest finish = 5 + 4 = 9. Float = LF − EF = 12 − 9 = 3 days (equivalently LS − ES = 8 − 5). Taking LF − ES = 7 forgets to subtract the duration.
430. **C** — CV = EV − AC = 60 − 75 = −15. A negative cost variance means more has been spent than the work done is worth, so the project is over budget. Schedule status comes from SV = EV − PV, not from CV.
431. **A** — C is not critical but has only 1 day of float. A 2-day delay makes path A-C-E-F = 3 + 4 + 6 + 2 = 15 days, longer than A-B-D-F (14), so the project grows to 15 days and A-C-E-F becomes the new critical path.
432. **A** — Both paths are critical: P-R-T = 5 + 7 + 2 = 14 and P-Q-S-T = 5 + 3 + 4 + 2 = 14. Shortening R alone makes P-R-T 12, but P-Q-S-T still needs 14 days. To gain time you must shorten activities on both paths, or a shared one (P or T).
433. **D** — CPI = EV / AC = 200 / 180 ≈ 1.11 (above 1, under budget). SPI = EV / PV = 200 / 250 = 0.8 (below 1, behind schedule). Spending less than planned here is a warning sign rather than good news, because less work has been done than planned.
434. **D** — 189.1 / 91.3 ≈ 2.07, slightly more than double. Because the exponent b (1.05) is above 1, effort per KLOC rises with size, reflecting extra communication and integration overhead. The effect is stronger in embedded mode (b = 1.20, a factor of about 2.30 for doubling).
435. **B** — With 5 people there are 5 × 4 / 2 = 10 paths; with 10 people there are 10 × 9 / 2 = 45. Newcomers also need ramp-up time and mentoring. This is Brooks's law: adding people late in a project tends to make it later, because people and months are not interchangeable.
436. **A** — Z = (18 − 15) / 2.69 ≈ 1.12, and the standard normal table gives P(Z ≤ 1.12) ≈ 0.87. 50% is the probability for the expected time itself (15 weeks), 13% is the probability of overrunning 18 weeks, and 97% corresponds to about 20 weeks (Z ≈ 1.86).
437. **A** — Effort and duration are different quantities. Some tasks cannot start until others finish, and every extra person adds communication and coordination paths. That is why COCOMO estimates development time separately (TDEV = c·E^d) rather than dividing effort by any team size.
438. **C** — Early on, size is unknown, so an exact-looking COCOMO figure gives false precision. Analogy and expert judgement (for example Wideband Delphi) give a reasonable range, and estimates should be refined as uncertainty narrows. A deadline is a constraint, not an estimate.
439. **C** — Paths: A-C-E-F = 13, B-D-F = 13 and A-D-F = 11. There are two critical paths of 13 days. Only an activity on both of them can shorten the project, and F is the only one common to both, so crashing F gives 12 days.

440. **A** — SPI = 110 / 100 = 1.10 (ahead) and CPI = 110 / 130 ≈ 0.85 (over budget). This pattern fits overtime: more work was done than planned, but each unit of work cost more than budgeted.
441. **A** — PV (also called BCWS, budgeted cost of work scheduled) is what the plan says should have been earned by now. EV is the budgeted cost of work actually done, AC is the actual spending, and BAC is the total budget.
442. **C** — AC (also called ACWP, actual cost of work performed) is what the project has truly spent so far. Comparing it with EV, the budgeted value of that same work, gives the cost variance and the CPI.
443. **B** — SPI = EV / PV. A value above 1 means the earned value is larger than the planned value, so the team is ahead of schedule. SPI says nothing about spending; that is the job of CPI = EV / AC.
444. **D** — Schedule variance compares the value of work done with the value of work planned: SV = EV − PV. If less work has been earned than planned, SV is negative and the project is behind schedule. EV − AC is the cost variance.
445. **B** — BAC is the sum of all planned values for the project, that is, the original total budget. The forecast total is the estimate at completion (EAC), the money still needed is the estimate to complete (ETC), and BAC − EAC is the variance at completion (VAC).
446. **B** — Free float is the delay an activity can absorb while every successor can still start at its earliest start. Total float is the larger allowance that only protects the project end date, so it may use up float that successors share.
447. **C** — Critical activities have no slack: their earliest and latest start times are equal, so total float = LS − ES = 0. Any delay to them delays the project. A critical activity need not be the longest single activity or have any particular resources.
448. **B** — Crashing buys time with money: extra staff, overtime or equipment shorten an activity. It only helps when applied to critical activities. Running planned-sequential work in parallel is fast tracking, and delaying work to fit resources is resource levelling.
449. **C** — Fast tracking overlaps activities that the plan placed in sequence, for example starting coding before design is fully signed off. It costs little money but raises the risk of rework. Adding people to shorten an activity is crashing.
450. **C** — Resource levelling moves activities so no resource is booked beyond its availability. If critical activities must wait for a busy person, the end date slips. Resource smoothing is the variant that only uses float and keeps the end date fixed.
451. **A** — Function point analysis sizes software by the user-visible functions it provides, counted from inputs, outputs, inquiries and files. Because it does not count code, it can be used early and compared across languages, unlike LOC.
452. **A** — Function points count five user-visible types: external inputs (EI), external outputs (EO), external inquiries (EQ), internal logical files (ILF) and external interface files (EIF). Each is weighted by complexity to give the unadjusted function point count.
453. **D** — Sommerville describes the COCOMO II application composition model for prototyping and systems built from reusable components. It counts application (object) points from the number of screens, reports and 3GL components. The post-architecture model uses KLOC.
454. **D** — Intermediate COCOMO rates 15 cost drivers such as required reliability, product complexity and analyst capability. Their multipliers are combined into an effort adjustment factor (EAF), and the nominal effort is multiplied by the EAF.
455. **B** — Detailed COCOMO keeps the intermediate cost drivers but gives each one different multipliers for each phase (such as design, coding and integration), and it can be applied module by module. Effort is still in person-months.
456. **B** — Planning poker is a consensus technique, similar in spirit to Wideband Delphi. Revealing estimates together avoids anchoring on the first number spoken, and the high and low estimators explain their reasoning before re-voting.
457. **D** — A timebox, such as a two-week sprint, has a fixed end date. If the team cannot finish everything, the lowest-priority scope is dropped or moved, rather than the deadline being moved.
458. **C** — The iron (triple) constraint links scope, time and cost: changing one forces a change in at least one of the others, with quality depending on the balance. People, product and process are three of Pressman's four Ps, a different idea.

459. **D** — Scope creep happens when new features or changes are added informally, without change control and without re-planning time and cost. It is a common cause of overruns, which is why projects use a change control process.

460. **B** — A milestone is a checkpoint, not work, so it takes no time. On a Gantt chart it is usually drawn as a diamond, and in a network it is a node or zero-duration activity that depends on the work it marks as finished.

461. **B** — In activity-on-arrow diagrams, activities are arrows and events are nodes. Some dependencies cannot be drawn without a dummy: a dashed arrow with zero duration and no resources. Activity-on-node diagrams do not need dummies.

462. **A** — The chief programmer team, proposed by Mills and described by Brooks, is a centralised structure built around one expert. Weinberg's egoless team is the opposite, with shared decisions and collective ownership of code.

463. **A** — TCPI = (BAC − EV) / (BAC − AC): the remaining work divided by the remaining money. A TCPI above 1 means the team must perform more efficiently than planned from now on to meet the budget.

464. **C** — The (o + 4m + p) / 6 formula comes from approximating a beta distribution. When many independent activity times are added along a path, their sum tends towards a normal distribution, which is why Z-scores are used for deadline probabilities.

465. **C** — E = 2.4 × 10^1.05 = 2.4 × 11.22 ≈ 26.9 person-months. Dropping the exponent gives 24.0, multiplying the exponent instead of raising to it gives 2.4 × 10.5 = 25.2, and 10.5 forgets the coefficient.

466. **B** — E = 3.6 × 100^1.20 = 3.6 × 251.2 ≈ 904 person-months. 360 ignores the exponent, 432 multiplies 360 by 1.2 instead of raising to the power, and 251 forgets the 3.6 coefficient.

467. **A** — TDEV = 2.5 × 120^0.35 = 2.5 × 5.34 ≈ 13.4 months. 42.0 multiplies 120 by 0.35, 48.0 divides 120 by 2.5, and 5.3 forgets the 2.5 coefficient. Average staff would be about 120 / 13.4 ≈ 9 people.

468. **B** — Productivity = size / effort = 20,000 / 55.8 ≈ 359 LOC per person-month. 2.8 divides effort by KLOC (the inverse), 1,116 multiplies 55.8 by 20, and 1,740 divides by the 11.5-month duration instead of the effort.

469. **D** — The effort adjustment factor is the product of the multipliers: 1.15 × 0.85 × 1.30 ≈ 1.271. Adjusted effort = 60 × 1.271 ≈ 76.2 person-months. 78.0 uses only 1.30, 66.0 averages the three multipliers (1.10), and 198 adds the multipliers (3.30) instead of multiplying them.

470. **A** — te = (o + 4m + p) / 6 = (6 + 36 + 18) / 6 = 60 / 6 = 10 days. The plain average (6 + 9 + 18) / 3 = 11 is the common mistake, and 9 is just the most likely value.

471. **C** — σ = (p − o) / 6 = (23 − 5) / 6 = 3 days. The variance would be 9 days^2. 18 is the range without dividing by 6, and 2.17 wrongly uses (p − m) / 6.

472. **D** — Z = (T − te) / σ = (46 − 40) / 4 = 1.5. The standard normal table gives P(Z ≤ 1.5) ≈ 0.933, so about 93%. 7% is the probability of missing the deadline, Z = 6 forgets to divide by σ, and 0.375 divides by the variance of 16.

473. **D** — A probability of about 84% corresponds to Z = 1 in the standard normal table. T = te + Zσ = 30 + 1 × 3 = 33 weeks. 30 weeks gives only 50%, 36 weeks (Z = 2) gives about 97.7%, and 39 weeks (Z = 3) about 99.9%.

474. **C** — The paths are A-C-E-G = 4 + 5 + 6 + 2 = 17, A-D-F-G = 11 and B-D-F-G = 10. The longest, A-C-E-G, is critical, so the project needs 17 days. 25 is the sum of all durations, which ignores parallel work.

475. **A** — B can finish at day 3 at the earliest, and its only successor D cannot start before day 4 (it also waits for A). So B can slip 1 day without delaying D's earliest start: free float = 4 − 3 = 1. Its total float is 7 (LS of D is 10, so LS of B is 7), which is a common confusion.

476. **B** — Work backwards from the 17-day finish. G must start by 15, so F must finish by 15 and start by 12, so D must finish by 12 and start by 12 − 2 = 10. Day 4 is D's earliest start and day 12 is its latest finish.

477. **D** — SV = EV − PV = 90,000 − 120,000 = −30,000. A negative SV means less work has been earned than planned, so the project is behind schedule. It says nothing about cost, and 0.75 is the SPI, not the variance.

478. **D** — When the variance is atypical, EAC = AC + (BAC − EV) = 250,000 + 300,000 = 550,000. The formula BAC / CPI = 500,000 / 0.8 = 625,000 assumes the overspend will continue at the same rate, which the stem rules out.

479. **A** — CPI = 300 / 360 ≈ 0.833, so EAC = BAC / CPI = 900,000 / 0.833 = 1,080,000. ETC = EAC − AC = 1,080,000 − 360,000 = 720,000. 1,080,000 is EAC itself, 600,000 is BAC − EV (budgeted remaining work), and 540,000 is BAC − AC.

480. **A** — VAC = 900,000 − 1,080,000 = −180,000. A negative VAC forecasts a cost overrun of that amount at the end. VAC is a cost measure, so it does not show lateness, and 1.2 is the ratio EAC / BAC rather than the variance.

481. **C** — EV = 35% of BAC = 0.35 × 400,000 = 140,000. CPI = EV / AC = 140,000 / 175,000 = 0.8, so the project is over budget. 1.25 is AC / EV (inverted), and 2.29 is BAC / AC.
482. **C** — Paths = 6 × 5 / 2 = 15. Forgetting to divide by 2 gives 30 (each pair counted twice), and 36 is n^2. The quadratic growth of paths is one reason behind Brooks's law.
483. **D** — UFP = 6×4 + 5×5 + 4×4 + 3×10 + 2×7 = 24 + 25 + 16 + 30 + 14 = 109. The value adjustment factor = 0.65 + 0.42 = 1.07, so FP = 109 × 1.07 ≈ 116.6. 109 is the unadjusted count, and 45.8 multiplies 109 by 0.42 alone.
484. **C** — Cost = effort × cost per person-month = 36 × 250,000 = PKR 9,000,000. Dividing gives 6,944, and 3,000,000 wrongly treats 36 person-months as 12 months.
485. **D** — Sprints = 240 / 30 = 8, and 8 sprints × 2 weeks = 16 weeks. Answering 8 weeks confuses sprints with weeks, and 16 sprints applies the two-week multiplier twice.
486. **A** — TCPI = (BAC − EV) / (BAC − AC) = 300,000 / 250,000 = 1.2. The team has 250,000 left for 300,000 worth of work, so it must earn 1.2 units of value per unit spent, while its CPI so far is only 0.8. That gap shows the target is unrealistic without scope or budget changes.
487. **A** — X: SPI = 70 / 100 = 0.70 and CPI = 70 / 74 ≈ 0.95, so it is badly behind schedule but only slightly over budget. Y: SPI = 0.98 and CPI = 98 / 93 ≈ 1.05, close to plan and slightly under budget. Spending more is fine when more work has been earned.
488. **D** — There are two critical paths, A-B-D and A-C-D, both 11 days. A day saved must come off both. Options: A (shared) costs 500, D (shared) costs 700, or B and C together cost 550. Crashing A is cheapest.
489. **A** — F is on A-D-F-G, which is only 11 days. F can finish as late as day 15, and its earliest finish is 9, so its total float is 6 days. A 4-day delay makes that path 15 days, still below 17, so the end date and critical path do not change.
490. **C** — Uncertainty is measured by variance = $((p - o) / 6)^2$. P and R: $(4 / 6)^2 \approx 0.44$; Q: $(10 / 6)^2 \approx 2.78$; S: $(8 / 6)^2 \approx 1.78$. Q has the widest spread. S has the largest expected time (7 days versus Q's 6), but that is length, not uncertainty.
491. **D** — Path 1: Z = (22 − 20) / 1 = 2, P ≈ 0.977. Path 2: Z = (22 − 19) / 3 = 1, P ≈ 0.841. Both paths must finish on time, so P ≈ 0.977 × 0.841 ≈ 0.82. The shorter but more variable path is the main threat, a known blind spot of looking only at the critical path.
492. **C** — Organic: 2.4 × 50^1.05 ≈ 146 PM. Embedded: 3.6 × 50^1.20 ≈ 394 PM. The ratio is about 2.7, because both the coefficient and the exponent grow. Flight control has tight hardware and real-time constraints, so embedded is the right mode, and the organic estimate would badly under-budget it.
493. **B** — At completion EV = BAC (all work earned) and PV = BAC (all planned work is due), so SPI = 1.0 even though the project is 3 months late. This is a known limitation of cost-based SPI late in a project; earned schedule methods address it.
494. **B** — Without the constraint the project takes max(4 + 2, 7) = 7 days. The specialist cannot do A and B at once, so they run one after another: 4 + 3 = 7, then C takes 2, giving 9 days, which now exceeds D's 7. Resource limits can create a longer, resource-critical path.
495. **B** — Pricing to win sets the cost from what the customer will pay. Sommerville notes it can win the contract but often leaves too little money, so functionality, schedule or quality must later be renegotiated. Parkinson's law says work expands to fill the time available.
496. **A** — Brooks's law concerns late projects, where ramp-up time, training load and extra communication paths outweigh the new effort. Independent, partitionable work added early, with clear documentation, can absorb extra people with little overhead.
497. **A** — Effort and duration are not freely interchangeable. Squeezing a schedule well below the nominal TDEV needs more people and more communication, which adds effort and risk (COCOMO II models this with the SCED driver). The realistic options are less scope, more time or a phased release.
498. **B** — A plain Gantt chart shows timing but not which tasks depend on which. An activity network (or a Gantt chart with links derived from it) lets the critical path method show whether the slipped task has float or is critical.
499. **D** — This is the "90% syndrome": estimates of percent complete are optimistic, and the last 10% takes as long as the first 90%. Earned value should be based on objective evidence, such as completed sub-deliverables or 0/100 or 50/50 rules.

500. **C** — Completed = 240 − 150 = 90 points in 3 sprints, so velocity = 30 per sprint against the 40 planned. Remaining 150 / 30 = 5 more sprints, giving 3 + 5 = 8 sprints, 2 beyond the 6 planned.
501. **A** — Cohesion is an intra-module property: the degree to which the elements of one module are functionally related. High cohesion means a module does one well-defined job. Dependence between modules is coupling, and the number of callers is fan-in.
502. **D** — Coupling is an inter-module property: how much one module relies on another's data, control or internals. Low coupling means a change in one module rarely forces changes in others. Relatedness inside a module is cohesion.
503. **B** — Each module should do one clear job (high cohesion) and depend on others as little as possible (low coupling). This makes modules easier to understand, test, reuse and change, because effects of a change stay local.
504. **A** — In the classic scale from worst to best (coincidental, logical, temporal, procedural, communicational, sequential, functional), functional cohesion is the best: every element contributes to a single, well-defined task such as computing a square root. Sequential is the second best.
505. **D** — Coincidental cohesion means the elements are grouped for no meaningful reason, as in a 'misc utilities' module. It is at the bottom of the scale. Logical cohesion (elements of the same general category, picked by a flag) is the next worst.
506. **D** — The classic coupling scale from loosest to tightest is data, stamp, control, external, common, content. Content coupling, where one module directly reads or changes another module's internal data or jumps into its code, is the worst because any internal change breaks the other module.
507. **C** — Data coupling means modules communicate only by passing the elementary data items they need as parameters. Stamp coupling passes a whole data structure when only part is used, control coupling passes flags that steer the other module's logic, and common coupling shares global data. All three are tighter than data coupling.
508. **A** — Gamma, Helm, Johnson and Vlissides classify patterns by purpose. Creational patterns deal with object creation (Singleton, Factory Method), structural patterns with composing classes and objects (Adapter, Decorator), and behavioural patterns with algorithms and responsibilities between objects (Observer, Strategy).
509. **B** — Singleton is a creational pattern whose intent is to ensure a class has only one instance and provide a global point of access to it, typically via a static method and a private constructor. Creating families of related objects is Abstract Factory. Restricting threads is a concurrency mechanism such as a lock.
510. **D** — Observer is a behavioural pattern. A subject keeps a list of observers and notifies them when its state changes, so the subject does not need to know their concrete classes. A surrogate is Proxy, interchangeable algorithms is Strategy, and interface conversion is Adapter.
511. **C** — Adapter is a structural pattern that lets classes with incompatible interfaces work together by wrapping one to match the interface the client expects. Adding responsibilities dynamically is Decorator, a simplified interface to a subsystem is Facade, and separating abstraction from implementation is Bridge.
512. **A** — UML 2 splits its diagrams into structural ones (class, object, component, deployment, package, composite structure, profile) and behavioural ones (use case, activity, state machine and the interaction diagrams such as sequence). A class diagram shows the static structure: classes, attributes, operations and relationships.
513. **A** — Both are whole-part associations. In composition the whole owns the part exclusively and controls its lifetime, so deleting an Order deletes its OrderLines; it is drawn with a filled diamond at the whole's end. In aggregation the part has its own lifetime (a Team and its Players), drawn with a hollow diamond. Inheritance is generalisation, a hollow triangle.
514. **A** — David Parnas (1972) argued that modules should be decomposed around design decisions likely to change, and should hide those decisions so other modules depend only on the interface. When the hidden decision changes, only that module needs editing. Encryption and source-code secrecy are unrelated meanings of 'hiding'.
515. **C** — Communicational cohesion groups activities that use the same data, for example a module that reads a customer record and computes both the balance and the credit rating from it. When one element's output feeds the next, that is sequential cohesion; running at the same time is temporal; same category is logical.
516. **B** — Decorator is a structural pattern. A decorator has the same interface as the object it wraps and adds behaviour before or after delegating to it, so features can be stacked at run time without a subclass for every combination. Tree structures are Composite, and an algorithm skeleton is Template Method.

517. **C** — Temporal cohesion groups elements because they execute at the same time, typically initialisation or shutdown. It is better than coincidental (where there is no link at all) but weaker than functional, because the tasks do not form one single job.
518. **A** — The functions share no purpose, data or timing; they just ended up in the same file. That is coincidental cohesion, the weakest kind. Logical cohesion would need them to belong to one category, such as all input routines.
519. **B** — When one module passes a control element (a flag or switch) that directs the internal logic of another, the modules are control-coupled. A must know how B works inside to set the flag. Data coupling passes only the data needed, and content coupling means touching another module's internals directly.
520. **D** — Common (global) coupling occurs when modules share global data. A change to the global's meaning or format can break every module that touches it, and bugs are hard to trace because any module could have changed the value. It sits just below content coupling on the scale.
521. **D** — Stamp (data-structure) coupling happens when a composite data structure is passed but only part of it is used. The function now depends on the Student structure, so changes to it may affect print_label. Passing only roll_no would reduce it to data coupling.
522. **B** — Singleton ensures only one instance exists and gives a global access point to it, which matches a single shared pool. Prototype creates new objects by copying, Builder constructs complex objects step by step, and Observer handles change notifications.
523. **B** — An Adapter class implements PaymentProcessor and translates each pay(amount) call into makePayment(amount, 'PKR') on the library object. Facade simplifies a whole subsystem rather than matching one interface, Decorator adds behaviour without changing the interface, and Strategy swaps algorithms.
524. **B** — Strategy defines a family of algorithms, encapsulates each one, and makes them interchangeable, so the client holds a reference to a SortStrategy and can swap it at run time. Singleton limits instances, Composite builds part-whole trees, and Abstract Factory creates families of related objects.
525. **D** — Facade provides a unified, higher-level interface to a set of interfaces in a subsystem, making it easier to use. Clients call convert() and the facade coordinates the codec classes. Adapter converts one interface to another expected one, Proxy controls access to an object, and Bridge separates abstraction from implementation.
526. **A** — Multiplicity is read at the far end of the association. Next to Teacher, 1..* says a department is linked to at least one teacher. Next to Department, 1 says each teacher is linked to exactly one department. Reading the numbers at the near end instead is a common mistake.
527. **D** — A sequence diagram is an interaction diagram that shows objects as lifelines and the messages between them in time order, top to bottom. Class, deployment and component diagrams are structural and show no time ordering of messages.
528. **D** — Config overrides __new__ so it creates the object only the first time and returns the same stored instance afterwards. a and b are the same object, so a is b is True. a.value is set to 5, then b.value += 1 changes that same object to 6. This is one way to implement Singleton in Python.
529. **C** — Each decorator wraps a drink and adds to it. The innermost object is Tea (100). Milk adds 30, the second Milk adds 30, and Honey adds 20: 100 + 30 + 30 + 20 = 180. desc() builds the string from the inside out, so the text starts with Tea. This is the Decorator pattern: features are stacked at run time.
530. **C** — When the part's lifetime depends on the whole, the relationship is composition, drawn with a filled diamond on the whole's (House) end. Aggregation (hollow diamond) is for parts that can outlive the whole. Generalisation means 'is-a', and a room is not a kind of house.
531. **C** — The class mixes four unrelated reasons to change (database schema, HTML layout, email provider, logging policy), which is low cohesion and breaks the single responsibility principle. Splitting it gives focused classes that can change and be tested independently.
532. **C** — X is data coupling (loosest), Z is control coupling, and Y is common coupling (tighter, because any module can change shared globals). On the scale data < stamp < control < external < common < content, the order is X, Z, Y.
533. **C** — process([4, 9, 2], 2) returns max = 9, and process([4, 9, 2], 3) falls through to len = 3, so it prints 12. The caller passes mode to choose which branch runs inside process, which is control coupling. Separate functions (total, largest, count) or a Strategy object would remove the flag.

534. **B** — Bridge separates an abstraction (Shape) from its implementation (Renderer) so the two hierarchies can vary independently. Shapes hold a reference to a Renderer, so you need 3 + 2 classes instead of 3 × 2, and adding a shape or renderer adds one class.
535. **B** — Each of the 4 extras is either present or absent, giving 2^4 = 16 combinations, each needing its own subclass. With Decorator you write one class per extra (4) plus the base, and wrap them at run time in any combination. This is why the GoF call Decorator a flexible alternative to subclassing.
536. **A** — Composite composes objects into tree structures and lets clients treat individual objects (File) and compositions (Folder) uniformly through one interface. Folder.size() sums the sizes of its children recursively. Iterator walks a collection, Observer handles notifications, and Facade simplifies a subsystem.
537. **B** — A Singleton reached through a static call is a hidden dependency with global state: tests cannot easily swap it for a fake, and state can leak between tests. Passing the logger in (dependency injection), even if only one instance is created, keeps the 'one instance' benefit while making the dependency visible and mockable.
538. **A** — The filled diamond is composition: the Engine is exclusively owned and its lifetime is bound to the Car. The hollow diamond is aggregation: Passengers are part of the Car's trip but have their own lifetimes. Neither is inheritance, which uses a hollow triangle.
539. **D** — Both counters receive 3 (c1 = 3, c2 = 3). c1 is detached, so only c2 receives 4 (c2 = 7). c1 is attached again and both receive 5: c1 = 8 and c2 = 12. In the Observer pattern only currently attached observers are notified.
540. **C** — Replacing shared globals with explicit parameters turns common coupling into data (or stamp) coupling, so each module depends only on what it receives. More globals or direct modification keep or worsen common coupling, and merging modules destroys cohesion.
541. **B** — In logical cohesion a module bundles related-sounding operations, such as all input handling, and the caller picks one with a control parameter. It ranks low on the cohesion scale and usually goes with control coupling. Running at the same time is temporal, chaining outputs is sequential, and one task is functional.
542. **C** — Procedural cohesion ties elements together by the order of control flow, not by shared data. When the steps share data it becomes communicational, and when each step's output is the next step's input it becomes sequential, both of which rank higher.
543. **D** — In sequential cohesion the module is a data pipeline, such as parse, then validate, then store, where each step consumes the previous step's result. It ranks just below functional cohesion. A fixed order without shared data is procedural cohesion.
544. **D** — External coupling arises when modules share something outside the software, such as a fixed file format, a protocol or a hardware device. It sits between control and common coupling on the usual scale. Parameter-only links are data coupling, and reaching into another module's internals is content coupling.
545. **A** — Pressman lists abstraction as a core design concept: at each level you describe the solution in terms that ignore lower-level details. Procedural abstraction names a sequence of steps (such as open a door), and data abstraction names a collection of data (such as a door object).
546. **A** — Stepwise refinement is top-down elaboration: a broad statement is decomposed into more detailed instructions until they can be coded. It complements abstraction. Improving structure without changing behaviour is refactoring.
547. **D** — Refactoring improves design, for example by removing duplication or splitting a long method, while the observable behaviour stays the same. Automated tests are run before and after to confirm nothing changed.
548. **A** — Pressman's modularity curve shows that development cost per module drops as modules get smaller, while integration cost grows with the number of interfaces. The total is lowest somewhere in between, so neither one huge module nor hundreds of tiny ones is ideal.
549. **B** — A functionally independent module does one job and has little interaction with others. Cohesion measures the first property and coupling the second, so functional independence means high cohesion and low coupling.
550. **A** — The I in SOLID favours several small, role-specific interfaces over one large "fat" interface. A class then implements only the roles it really has, and clients see only the methods they need.
551. **C** — The D in SOLID says policy code and detail code should both depend on an interface. For example, an OrderService depends on a Repository interface, and a MySQL class implements it. The direction of dependency is inverted towards the abstraction, not simply reversed between layers.
552. **C** — The GoF book lists 5 creational patterns (such as Singleton and Builder), 7 structural ones (such as Adapter and Composite) and 11 behavioural ones (such as Observer and Strategy). Behavioural is the largest group.

553. **A** — Factory Method defers instantiation to subclasses through an overridable creation method. Singleton limits instances, Abstract Factory creates whole families of related objects, and Builder assembles a complex object in steps.

554. **B** — Abstract Factory produces a matching set of products, such as a dark-theme button, menu and scrollbar, so that parts from different families are not mixed. Factory Method concerns one product, Prototype clones objects and Facade simplifies a subsystem.

555. **B** — Builder constructs an object step by step through a director and builder interface, so the same steps can yield different results. It suits objects with many optional parts. Attaching responsibilities is Decorator, and one-to-many notification is Observer.

556. **B** — A proxy has the same interface as the real object and stands in for it, for example to delay loading (virtual proxy), check permissions (protection proxy) or reach a remote object. Converting interfaces is Adapter, trees are Composite, and state-dependent behaviour is State.

557. **A** — Template Method fixes the order of steps in a base-class method and lets subclasses override individual steps, using inheritance. Strategy swaps a whole algorithm through composition at run time, Command wraps a request, and Iterator traverses a collection.

558. **A** — Class patterns work through inheritance, so the relationships are static. Object patterns, which are the majority, use composition and delegation between objects, so the relationships can change at run time. Factory Method, class Adapter, Interpreter and Template Method are the class-scope patterns.

559. **D** — UML visibility markers are + for public, − for private, # for protected and ~ for package. Static members are shown underlined, not with a symbol.

560. **D** — Realisation is drawn as a dashed line with a hollow triangle at the interface, which you can remember as a dashed generalisation. A solid line with a hollow triangle is generalisation (inheritance), a filled diamond is composition, and a plain dashed open arrow is a dependency.

561. **B** — Fan-in counts callers and fan-out counts callees. Structured design guidance favours high fan-in for reusable low-level modules and keeps fan-out moderate, because a very high fan-out suggests a module controls too much.

562. **C** — Transaction analysis suits data flow diagrams where an incoming item is classified and routed to one of several action paths, such as a menu-driven system. Transform analysis suits a linear flow with afferent (input), central transform and efferent (output) parts.

563. **B** — Both displays receive notification 1. Then a is detached, so only b receives 2 and 3. a has seen 1 message and b has seen 3. This is the Observer pattern: the subject keeps a list and calls update() on each observer currently attached.

564. **A** — Percent(10) gives 500 − 500 × 10 // 100 = 450. The cart's strategy is then swapped to Flat(600), which returns max(0, 500 − 600) = 0. The Cart code never changes; only the strategy object does, which is the Strategy pattern.

565. **C** — run() is the template method: it fixes the order read, clean, save. SafeCsvImporter inherits read() from CsvImporter ("csv"), overrides clean() ("trim+dedupe") and inherits save() from Importer ("db").

566. **A** — docs = 10 + 20 = 30. photos = 100 + (5 + 5) + 50 = 160, because File(50) is appended to photos' list before size() is called, and root holds a reference to the same photos object. root = 30 + 160 + 1 = 191. root has 3 direct items. This is the Composite pattern: files and folders share size().

567. **A** — The turnstile starts Locked: push is blocked. A coin switches it to Unlocked. A second coin in Unlocked is refunded. Push passes and switches back to Locked, so the last push is blocked. Behaviour depends on the current state object, which is the State pattern.

568. **A** — Each ImageProxy creates its RealImage only on the first show(). Image a is shown twice but loaded once, image b is never shown, and image c is loaded once, so loads = 2. This is a virtual proxy that delays an expensive operation.

569. **D** — The commands give 1 + 4 = 5, 5 × 3 = 15, 15 + 2 = 17. Undo pops the last command first: Add(2) gives 15, then Mul(3) gives 5. One command, Add(4), remains in history. Storing requests as objects with undo() is the Command pattern.

570. **B** — Each approver handles the amount if it is within its limit (<=), otherwise passes it on. 800 stops at lead, 5000 stops at manager (5000 <= 5000), 7500 reaches director, and 25000 exceeds every limit, so the last handler returns "board". This is Chain of Responsibility.

571. **D** — The three tasks are only alike in category (printing) and the caller selects one with a flag, which defines logical cohesion. The flag also creates control coupling with the caller. Splitting it into three functions gives each one functional cohesion.
572. **C** — Each step uses the previous step's output as its input: marks, then GPA, then grade. That data chain defines sequential cohesion, which is the second-strongest level after functional cohesion.
573. **B** — The steps are linked only by the order of execution, not by shared data or a data chain, so the routine is procedurally cohesive. Sequential cohesion would need each output to feed the next step, and communicational cohesion would need the steps to use the same data.
574. **A** — Reaching into another module's internal data or code is content coupling, the tightest kind. Any change to Account's internals can now break Billing. The fix is to go through Account's public methods, such as debit().
575. **D** — The modules depend on a format imposed from outside the software, which is external coupling. If the bank changes the format, both modules must change together. Isolating the format in one adapter module limits the damage.
576. **B** — A LightFactory and a DarkFactory each implement the same interface (create_button, create_checkbox, create_menu). The client picks one factory and gets a consistent family of widgets, which is exactly Abstract Factory's intent.
577. **D** — Builder assembles a complex object step by step, for example RequestBuilder(url).header(...).timeout(5).build(). It avoids constructors with long lists of optional arguments.
578. **C** — Prototype creates new objects by copying a fully configured instance and then adjusting a few fields, which avoids repeating the costly set-up. In Python this is often done with copy.deepcopy.
579. **D** — Mediator encapsulates how a set of objects interact, so they refer to the mediator instead of to each other. That replaces a tangled many-to-many web of links with simple one-to-one links to the tower.
580. **C** — Flyweight uses sharing to support large numbers of fine-grained objects. The shared intrinsic state (glyph and style) is stored once, and the extrinsic state (position) is passed in when needed.
581. **B** — Iterator provides sequential access to a collection's elements without exposing its internal representation. Python's for loop uses this pattern through __iter__ and __next__.
582. **C** — Multiplicity is read at the far end. The 0..1 next to Passport says how many passports one person has: zero or one. The 1 next to Person says how many persons one passport belongs to: exactly one.
583. **A** — Three modules (read, process and write) call log, so its fan-in is 3. main directly calls three modules, so its fan-out is 3. Fan-out counts only direct callees, not modules called further down.
584. **C** — Bold(Upper(Text)): Upper makes "HI", then Bold wraps it as HI. Upper(Bold(Text)): Bold makes hi, then Upper capitalises the whole string, tags included, giving <B>HI</B>. Decorators compose, so their order matters.
585. **A** — stretch() expects that setting width 5 and height 4 gives area 20, which holds for Rectangle. Square keeps its sides equal, so set_height(4) also sets the width to 4, giving 16. A Square cannot safely replace a Rectangle, which violates Liskov substitution.
586. **B** — Because new behaviour requires modifying tested code, the function is not closed for modification. Polymorphism lets you add a Hexagon class with its own area() without touching existing code, which makes the design open for extension.
587. **D** — The high-level OrderService depends on a low-level concrete class. If it depends on an abstraction such as Repository, passed in through the constructor, tests can pass an in-memory fake and PostgreSQL can be added without editing OrderService.
588. **B** — This is a fat interface that forces Robot to depend on methods it cannot support, breaking interface segregation. Role-specific interfaces let each class implement only what it really does.
589. **D** — The Law of Demeter says a method should talk only to its immediate collaborators. Chaining through Order to Customer to Address couples InvoicePrinter to the structure of three classes, so a change to any of them can break it. A method such as order.shipping_city() hides that structure.
590. **B** — State and Strategy have similar class diagrams, but in State the object's own transitions change the behaviour, and the states know about each other. In Strategy the client chooses an interchangeable algorithm, and strategies are independent of each other.

591. **C** — A wrapper with the same interface whose job is to control access to the real object is a protection proxy. Adapter changes the interface, Facade offers a new simpler interface over many classes, and Composite builds part-whole trees.

592. **C** — Q is sequential (each output feeds the next step), which is near the top of the scale. P is temporal (grouped only by time of execution), in the middle. R is coincidental, the weakest. So the order is Q, P, R.

593. **C** — Passing a whole structure when only part is used is stamp coupling. Passing just the needed values gives data coupling, the loosest form. A global gives common coupling, a flag adds control coupling, and editing private fields is content coupling, all of which are tighter.

594. **C** — Fully connected widgets need $n(n - 1) / 2 = 6 \times 5 / 2 = 15$ links. With a Mediator, each widget links only to the mediator, giving 6 links. Adding a 7th widget then adds 1 link instead of 6.

595. **D** — Inheriting behaviour that some subclasses must cancel is a sign of misused inheritance. The GoF advise favouring object composition over class inheritance: FlyWithWings, NoFly and RocketFly classes can be plugged in, even at run time.

596. **A** — This is the lapsed listener problem. The subject's observer list keeps closed windows alive, so they are never garbage collected and still receive notifications. Detaching on close, or holding observers through weak references, fixes it.

597. **C** — High fan-in on a small utility module is good: it is reused rather than duplicated. A very high fan-out means one module coordinates many others, which usually signals low cohesion and too much control logic, so an intermediate level of modules should be introduced.

598. **B** — Duplicated code is a classic refactoring smell: every fix must be repeated, and copies drift apart. Extracting it to one place (Extract Method or Extract Class) means a fix is made once. Merging four classes would destroy their cohesion.

599. **B** — Bass, Clements and Kazman define architecture as the set of structures needed to reason about the system, made of elements, relations and their properties. Algorithms inside classes are detailed design, and tool or hardware choices are not the architecture itself.

600. **A** — In a strict, closed layering each layer uses only the services of the layer immediately beneath it. Allowing calls to any lower layer is a relaxed (open) layering, and upward calls break the layering rule altogether.

601. **B** — In client-server, clients know the server and send it requests; the server offers a service and replies. Equal roles describe peer-to-peer, and communicating through a shared store is the repository style. Servers do not need to know clients in advance.

602. **B** — The controller handles user interaction (clicks, key presses, HTTP requests) and translates it into actions on the model. The model holds data and business rules, and the view presents the model to the user.

603. **B** — Microservices are small services built around business capabilities, each independently deployable and usually owning its data store (database per service). A single release unit describes a monolith, a shared schema couples services, and services may use different languages.

604. **C** — In the repository pattern all data is held in a central store that every component can access, so components do not interact directly. Sommerville gives IDEs and CASE tools as typical examples. The other options describe event-driven, pipe-and-filter and layered styles.

605. **D** — The physical (deployment) view maps software onto processors and network nodes. The logical view shows key abstractions such as classes, the process view shows run-time processes and their interaction, and the development view shows how the code is organised into modules for developers.

606. **C** — The +1 is the scenarios view: a small set of use cases that illustrate and validate the logical, process, development and physical views. There is no separate security or data view in Kruchten's original model.

607. **B** — Pipe and filter organises a whole system as a chain of transformations, so it is an architectural pattern. Singleton, Adapter and Factory Method are Gang of Four design patterns that structure a few classes inside one component.

608. **A** — These are quality attributes: measurable properties that show how well the system does its job. Architecture is chosen largely to meet them. Functional requirements say what the system does, and views are ways of documenting the architecture.

609. **D** — In Bass et al., heartbeat (like ping/echo) is a fault-detection tactic for availability: if the heartbeat stops, the monitor assumes the component has failed and starts recovery. It does nothing to make code easier to change, use or test.

610. **B** — Publishers emit events to a channel or broker without knowing the subscribers, so new consumers can be added without changing the publisher. The price is harder-to-trace control flow, and most systems use a broker; global ordering is not automatic.
611. **B** — A tradeoff point is a decision that is a sensitivity point for more than one quality attribute, improving one while harming another (for example, encryption improving security but costing performance). A decision critical for a single attribute is a sensitivity point, and an unanalysed risky decision is a risk.
612. **D** — Sommerville distinguishes thin-client and fat-client two-tier models. In a fat client, the client runs both presentation and application processing, and the server mainly manages data. A thin client only presents; a separate middle tier makes it three-tier.
613. **A** — In classic (Smalltalk) MVC the view registers with the model, and the model notifies its observers when its state changes; the view then queries the model and redraws. The model does not know how it is drawn, and views do not bypass the model to reach the database. Many web frameworks instead let the controller pass data to the view, which is why textbook descriptions vary.
614. **B** — An API gateway sits between clients and the services: it routes and sometimes aggregates requests and handles cross-cutting concerns such as authentication and rate limiting. It holds no shared business data and does not merge the services into one deployment.
615. **C** — Each program is a filter that reads a stream, transforms it and passes the result through a pipe to the next filter. Filters do not share state or know about each other, which is the pipe-and-filter style.
616. **D** — Presentation (browser), application logic (application server) and data management (database server) each sit on their own tier, so it is three-tier client-server. A two-tier fat client would put the business rules on the client.
617. **C** — Network stacks are the textbook example of layering: application, transport, network and link layers each provide services to the layer above and use the one below, so a layer can be replaced without changing the others.
618. **A** — Sensors publish to topics and consumers subscribe to the topics they care about, with neither side knowing the other. That decoupling of producers from consumers is the publish-subscribe form of event-driven architecture.
619. **A** — All tools work on one central shared data structure and do not talk to each other directly. Sommerville uses exactly this kind of IDE as his example of the repository pattern.
620. **D** — The template only presents data, so it plays the MVC view role. The `Course` class is the model, and the request-handling function acts as the controller. Django calls its own scheme Model-Template-View, where its "view" function is the controller, which is a common source of confusion.
621. **C** — Independent scaling of one capability and independent releases by separate teams are the main reasons to choose microservices. Copying the whole monolith wastes resources and still couples releases, and a shared repository couples the teams through one schema.
622. **C** — With a tiny team, low load and unclear service boundaries, the operational cost of microservices (deployment, monitoring, distributed data) outweighs the benefits. A modular, layered monolith is quick to build and can be split later once boundaries are clear.
623. **C** — In strict layering every request passes through each layer in turn, so the UI asks the business layer, which asks data access, which queries the database. Skipping layers is allowed only in a relaxed layering, and the database never initiates calls upward.
624. **A** — Each node acts as both client and server and there is no central file server, which is the peer-to-peer style used by BitTorrent-like systems. Client-server would put the file on dedicated servers.
625. **B** — Putting the logic on a central application server (a thin client or three-tier design) means a rule change is deployed once on the server. Duplicating rules creates two copies to keep in step, and more RAM does not touch the deployment problem.
626. **C** — With database-per-service, a service's data is reachable only through its API (or events it publishes). Reading or writing another service's tables couples their schemas and destroys independent deployment.
627. **B** — Availability = MTBF / (MTBF + MTTR) = 990 / (990 + 10) = 990 / 1000 = 0.99, or 99%. Bass et al. use this ratio to express availability.
628. **D** — Both must be up, so multiply: 0.999 × 0.999 = 0.998001, or about 99.80%. Chaining services lowers availability, which is one hidden cost of long synchronous call chains in microservices.

629. **D** — The defining benefit of microservices is independent change and deployment. A shared schema couples every service, so the system has the costs of distribution without the benefits, often called a distributed monolith. Each service should own its data.
630. **A** — Sommerville notes that layering supports change and portability, but each extra layer adds processing and indirection, which can hurt performance. Relaxing some layers or caching are typical responses. The slowness is a known cost of layering, not evidence against it.
631. **A** — Sommerville links each critical attribute to a structure: for security, a layered architecture that protects the most critical assets in the inner layers with high levels of validation. Large-grain co-located components target performance, fine-grained components target maintainability, and wide replication increases the attack surface.
632. **C** — To maximise performance, Sommerville advises localising critical operations in a small number of large components, ideally on one computer, to reduce communication. Microservices, extra layers and remote repositories all add network hops or indirection that cost time.
633. **C** — MVC separates concerns: the model holds data and business rules, the view presents, and the controller mediates. "Fat controllers" duplicate rules and are hard to test, so the logic belongs in the model (or a service layer beneath it).
634. **A** — Most brokers give at-least-once delivery, so duplicates after a crash are expected. The consumer must make repeated processing harmless (idempotent), usually by de-duplicating on an event or payment ID. Synchronous calls also get retried, and more consumers or longer retention do not prevent duplicates.
635. **D** — Sommerville lists this as the main drawback of the repository pattern: components share one data model, so a change to it ripples to all of them (and the repository can be a single point of failure). Direct communication is what the pattern avoids, and data sharing is its strength.
636. **A** — The cost is mostly round trips: 30 calls over a high-latency link. An aggregating gateway or backend-for-frontend makes one client request and fans out over the fast internal network, cutting client round trips sharply. Smaller services would mean even more calls.
637. **D** — Sommerville notes that pipe and filter fits batch and transaction processing where inputs flow through transformations, but is not suitable for interactive systems that must respond to events. Reuse of filters is actually one of its strengths.
638. **D** — A circuit breaker tracks failures and slow calls to B; once a threshold is crossed it stops calling B for a while and returns a fallback immediately, so A's threads stay free. Longer timeouts make the thread exhaustion worse, and logging does not stop it.
639. **B** — In relaxed layering a layer may use any lower layer, skipping intermediate ones, which can improve performance. Strict layering allows calls only to the layer directly below. Upward calls are not allowed in either form.
640. **C** — A sensitivity point is a design property that strongly affects one quality attribute, such as encryption key length affecting security. A decision that affects several attributes in opposite directions is a tradeoff point, a different ATAM term.
641. **C** — The development (implementation) view shows the static organisation of the software in its development environment: subsystems, packages and libraries. The physical view maps software to hardware, and the process view covers runtime concurrency.
642. **D** — The process view captures the runtime behaviour: processes, threads, how they communicate and synchronise, and properties such as performance and availability. Hardware mapping is the physical view, and classes belong to the logical view.
643. **D** — The logical view describes what the system provides to end users, typically with class and object diagrams. The development view organises code modules, the process view covers runtime concurrency, and the physical view covers hardware.
644. **D** — In the thin-client model the client handles the user interface and the server does both application processing and data management. This keeps clients simple but puts a heavy load on the server and network.
645. **B** — A broker decouples clients from servers by handling location and communication. Clients send requests to the broker, which finds a suitable server and returns the result. That gives location transparency at the cost of an extra hop.
646. **A** — Tactics are smaller building blocks than patterns. Each one targets a single quality attribute response, such as heartbeat for availability or caching for performance. Patterns usually bundle several tactics.

647. **A** — The six parts are source of stimulus, stimulus, environment, artifact, response and response measure. A cost estimate is not part of the scenario, although cost may be weighed later when choosing between options.
648. **C** — Modifiability tactics aim to reduce the cost of change, for example by increasing cohesion, encapsulating, or using an intermediary so a change in one module does not ripple. Heartbeat and spares are availability tactics; authentication is a security tactic.
649. **D** — Maintaining multiple copies of data or computations (caching, replicas) is a classic performance tactic. Increasing cohesion targets modifiability, audit trails target security, and voting targets availability.
650. **C** — Authenticating actors is a security tactic: it resists attacks by confirming who the user is. It does nothing for uptime or ease of change and actually adds a little time to each login.
651. **B** — Instances register with a service registry, and clients or a load balancer look up a healthy instance at request time. Hard-coded addresses break as soon as instances change.
652. **C** — Distributed ACID transactions across services are costly and couple them. A saga breaks the process into local transactions; on failure it applies compensating actions, such as cancelling an order, giving eventual consistency.
653. **C** — Conway observed that system structure tends to copy the communication structure of the teams that build it. Microservice advice often aligns team boundaries with service boundaries for this reason. 'Adding people makes it later' is Brooks's law.
654. **C** — The strangler fig approach routes some requests to new components while the rest still go to the legacy system. Functionality moves over piece by piece, which avoids the risk of a big-bang rewrite.
655. **A** — Sommerville (after Bass et al.) says an explicit architecture helps stakeholders discuss the system, allows early analysis of properties such as performance, and enables reuse of the architecture across similar systems.
656. **C** — Architecture in the small is about how a single program is split into components. Architecture in the large is about enterprise systems made of other systems, often distributed and managed by different companies.
657. **A** — MVC lets data change independently of its representation and supports several views, but for simple systems the separation adds code and complexity that may not pay off.
658. **B** — Sommerville notes that the format for data transfer must be agreed between transformations, and each one must parse its input and unparse its output. Reuse and easy addition of filters are actually strengths of the style.
659. **C** — Client-server is easy to distribute and extend, but a server is a single point of failure unless replicated, and network conditions make performance unpredictable. Clients use a server's interface, not its internal data.
660. **D** — Horizontal scaling (scaling out) adds instances behind a load balancer. Vertical scaling (scaling up) adds resources to a single machine and is limited by the largest machine available.
661. **D** — When a service keeps no client session state in memory, a load balancer can send each request to any instance, and instances can be added or removed freely. State is kept in a shared store or sent with the request.
662. **D** — CQRS uses different models, and sometimes different stores, for writes and reads, so each can be optimised and scaled separately. It adds complexity and often leads to eventual consistency between the two sides.
663. **D** — The service fails only if both servers fail: 0.01 × 0.01 = 0.0001. Availability = 1 − 0.0001 = 0.9999, or 99.99%. 98.01% is the series result (0.99 × 0.99), which applies when both must be up.
664. **A** — Availability = MTBF / (MTBF + MTTR). Solving 0.999 = MTBF / (MTBF + 1) gives MTBF = 0.999 / 0.001 = 999 hours.
665. **D** — Unavailability is 0.001 of 8,760 hours, which is 8.76 hours. 52.6 minutes corresponds to 99.99%, and 87.6 hours to 99%.
666. **B** — In series, every service must be up, so multiply: $0.995^3 = 0.98507$, or 98.51%. Simply subtracting 3 × 0.5% gives 98.50%, a close but wrong approximation.
667. **A** — Skipping a layer is allowed only in relaxed (open) layering. The design is still layered, but the UI now depends on Data Access, so a change there can affect the UI directly.
668. **B** — Threads, processes, queues and synchronisation are runtime concerns, which belong to the process view. The physical view would show which servers host those processes.
669. **C** — Layering isolates change: the upper layers depend only on the Data Access interface, so replacing the database affects the layer that implements that interface. This is the modifiability benefit of layers.

670. **A** — The figures are the model; the table and chart are two views of it. When the model changes, both views are notified and refresh. Supporting multiple views of the same data is a key reason to use MVC.
671. **D** — Publishing an event lets any number of subscribers react, and new ones can subscribe without changing the publisher. Direct calls would mean editing the order service every time a consumer is added.
672. **B** — The response measure makes the response testable. Here the stimulus is the crash, the environment is normal operation, the artifact is the server, the response is detection and recovery, and 30 seconds is the measure.
673. **A** — The data is read often and changes rarely, so keeping a copy (a cache) avoids repeating the costly query. Heartbeat helps availability, extra layers usually slow things down, and authentication is a security tactic.
674. **A** — A saga cannot use one ACID transaction across two databases. When a later step fails, it undoes earlier steps with compensating actions, here marking the order as cancelled.
675. **A** — The source yields 3, 8, 5, 12, 7. keep_odd passes only 3, 5 and 7, and square turns them into 9, 25 and 49. Each generator is a filter that reads from the one before it.
676. **B** — set() stores 3, then calls update on each attached view in attachment order: TextView prints 'text 3', then BarView prints 'bar' followed by three hash signs. The model never needs to know the views' concrete types.
677. **D** — Publishing 'order' calls both order subscribers in subscription order. Publishing 'refund' finds no subscribers, so nothing happens, and the 'payment' subscriber never runs.
678. **B** — The first failure sets fails to 1; the success resets it to 0. The next two failures reach the limit of 2, so the breaker opens after returning 'error'. The last call is rejected immediately with 'fast-fail'.
679. **C** — Round robin cycles A, B, C, A, B, C, A, B, C, A. A gets requests 1, 4, 7 and 10, so 4 requests; B and C get 3 each.
680. **A** — A routing layer sends selected requests to new services while the rest still reach the monolith. Features migrate one at a time until the old system can be retired, which keeps risk low.
681. **B** — Conway's law says the system mirrors the organisation's communication structure. Teams split by technology produce layer-shaped systems; to get capability-based services, teams are usually organised around capabilities.
682. **A** — With dynamic instances, addresses must be resolved at call time. Instances register on start-up and deregister on shutdown, and clients query the registry or go through a load balancer that uses it.
683. **C** — In active redundancy the spare is kept fully in step and can take over almost instantly. A warm (passive) spare gets periodic updates and needs some catch-up, and a cold spare must be started from scratch.
684. **B** — This is the layering trade-off: strictness buys modifiability but costs performance. If a layer adds nothing for some requests, letting the layer above skip it (relaxed layering) cuts latency while keeping the layers for the rest.
685. **A** — Loose coupling in publish-subscribe means no single place shows who reacted to what. Tagging every event with a correlation ID and logging centrally lets developers rebuild the chain of events.
686. **D** — Series calls add their latencies and multiply their availabilities, so a long chain is both slow and fragile. Fewer synchronous hops, or asynchronous messaging where possible, improves both.
687. **D** — Every cache can serve stale data. The fix is to bound staleness where it matters: invalidate on price updates, shorten the TTL, or read the authoritative price at the step that charges money.
688. **D** — Sommerville recommends the repository pattern when large volumes of information are shared and stored for a long time. Sending the whole model around as events would be wasteful and error-prone.
689. **B** — In MVC the view presents data; changing state and applying rules belong to the model, coordinated by the controller. Mixing them makes the logic hard to find, test and reuse.
690. **C** — A tradeoff point is a property that affects more than one quality attribute and improves one while harming another. Here security rises and performance falls, so architects must weigh the two.
691. **B** — A single server is a single point of failure. Redundant instances behind a load balancer let updates roll one at a time without downtime.
692. **C** — With long periods offline, the logic and data must live on the device. Syncing later reconciles with the server. The usual cost of fat clients is deploying updates, which app stores make easier.
693. **C** — Only one capability has a different scaling need, so extracting that one gives independent scaling without the cost of a full microservices rewrite.

694. **D** — Sharing domain data classes ties every service to one release cycle, which is a hidden form of the distributed monolith. Services should own their models and share only stable contracts, such as API schemas.
695. **A** — When filters run concurrently, the slowest one is the bottleneck: one item every 10 ms, which is 100 items per second. Summing times (15 ms) gives the latency of one item, not throughput. Parallelising the slow filter would help most.
696. **A** — Asynchronous delivery does not guarantee order. Per-key ordering (such as partitioning by product) or version numbers let consumers discard stale events, so the latest value always wins.
697. **B** — Layers should depend only downwards. An upward dependency means Business Logic cannot be reused with another UI, such as an API. Returning results or raising exceptions keeps the direction correct.
698. **B** — Microservices pay off when parts need independent scaling or independent team releases. With a small team, steady load and no hot spots, the costs of distribution outweigh the gains.
699. **D** — Visibility of system status (heuristic 1) asks for appropriate feedback within reasonable time, such as progress bars or a "Saved" message. Matching the real world is about familiar language and conventions.
700. **D** — Recognition is easier than recall, so the design should make objects, actions and options visible (menus, recent files, autocomplete) instead of making users remember them. The other options describe minimalist design, flexibility and efficiency, and help and documentation.
701. **C** — Permitting easy reversal (undo) is one of Shneiderman's rules; it lowers anxiety and encourages exploration. The others contradict his rules on informative feedback and reducing short-term memory load.
702. **D** — Learnability is first-time ease of use. Speed for experienced users is efficiency, returning after a break is memorability, and pleasantness is satisfaction. The fifth component is errors: how many users make and how easily they recover.
703. **D** — Low-fidelity prototypes (paper, sketches, wireframes) are cheap and fast, so they are used early to explore ideas and gather feedback on structure and flow. High-fidelity prototypes look and behave much closer to the final product.
704. **B** — ISO 9241-11 defines usability through three measures: effectiveness (can users complete the goal), efficiency (resources spent) and satisfaction, always for particular users, goals and a context of use.
705. **A** — Heuristic 2 asks the system to speak the user's language and follow real-world conventions, so familiar metaphors such as a cart and checkout fit it. Consistency and standards is about following platform and internal conventions so the same thing always looks and works the same way.
706. **D** — Pressman's three golden rules, taken from Mandel, are: place the user in control, reduce the user's memory load and make the interface consistent. They overlap with Shneiderman's and Nielsen's principles on control, memory and consistency.
707. **B** — Sequences of actions should be organised into groups so users get a sense of completion, for example a confirmation page after an online purchase. It says nothing about auto-closing windows, and blocking cancellation would break his rule on easy reversal.
708. **C** — An affordance is the possible action (a door can be pushed); a signifier is the visible cue that communicates it (a flat push plate). Mapping is the control-to-effect relationship, and a constraint limits the possible actions.
709. **B** — Nielsen's scale: 0 not a problem, 1 cosmetic, 2 minor, 3 major (important to fix, high priority), 4 catastrophe (imperative to fix before release). Severity combines frequency, impact and persistence.
710. **B** — A horizontal prototype covers the breadth of the interface (many screens, little functionality behind them), while a vertical prototype implements a narrow slice fully, down to real data. Fidelity and whether it is thrown away are separate dimensions.
711. **C** — Sommerville and Pressman describe throwaway prototypes as a way to explore unclear requirements, after which the code is discarded. Evolutionary prototypes start with understood requirements and grow into the final product, so they must be built to production quality.
712. **D** — The gulf of evaluation is the gap between the system's output and the user's ability to interpret it; good feedback and visible status bridge it. The gulf of execution is the gap between intention and the available actions.
713. **C** — A cognitive walkthrough focuses on learnability for first-time users, step by step through a task. Heuristic evaluation checks the interface against a list of principles, A/B testing compares two live versions with real traffic, and card sorting helps design information architecture.

714. **B** — In later editions, rule 2 became "Cater to universal usability" (5th) and then "Seek universal usability" (6th), which covers diverse users, including novices and experts (shortcuts), disabilities and different devices. Consistency, feedback and memory load were already rules in earlier editions. Different course books may quote either version.
715. **A** — Users cannot see that the upload is in progress, so they repeat the action. A progress bar or spinner with status text would satisfy visibility of system status.
716. **C** — The same action looks and sits differently from screen to screen, so users must relearn it. Consistency (Nielsen's heuristic 4, Shneiderman's first rule) asks for the same terms, layout and behaviour for the same actions.
717. **A** — Instead of reporting an invalid date after the fact, the picker makes the error impossible. Nielsen ranks preventing errors above writing good error messages.
718. **B** — Error messages should be in plain language, state the problem precisely and suggest a fix, such as "Phone number can contain digits only". An internal code helps no ordinary user recover.
719. **D** — Paper screens operated by a person are a low-fidelity prototype. They are cheap to change, so they suit early exploration of flow and layout before any code is written.
720. **D** — Accelerators speed up expert users without getting in the way of novices, which is exactly what flexibility and efficiency of use asks for. Shneiderman's earlier rule on shortcuts for frequent users makes the same point.
721. **A** — Users often choose functions by mistake and need a clearly marked "emergency exit" (back, cancel, undo). Trapping them in a flow violates user control and freedom, and also Shneiderman's rules on reversal and keeping users in control.
722. **A** — In a Wizard of Oz prototype a human secretly simulates functions that are not built yet, so the team can study user behaviour early. No code is kept, so it is not evolutionary prototyping.
723. **D** — 1 − 0.69^5 = 1 − 0.156 = 0.844, so about 84% (Nielsen rounds this to "about 85%"). Returns fall with each extra user, which is why Nielsen suggests several small tests rather than one big one.
724. **C** — For 7 options: 0.2 × log2(8) = 0.6 s. For 15 options: 0.2 × log2(16) = 0.8 s. The difference is 0.2 s, not double, because decision time grows logarithmically with the number of choices.
725. **A** — Once the placeholder vanishes, users must remember what each field asked for, especially when they go back to check. Persistent labels keep the information visible, supporting recognition rather than recall.
726. **A** — Using the WCAG relative-luminance formula, #767676 on white gives about 4.54:1, just above 4.5:1. #777777 gives about 4.48:1, which fails, and #999999 and #AAAAAA are lower still (about 2.85:1 and 2.32:1). Large text needs only 3:1.
727. **A** — D/W + 1 = 256/16 + 1 = 17, and log2(17) ≈ 4.09 bits. MT = 0.1 + 0.2 × 4.09 ≈ 0.92 s. 0.90 s comes from dropping the +1, 1.10 s from Fitts's original log2(2D/W) form, and 3.50 s from forgetting the logarithm.
728. **B** — Preventing the slip is better than handling it afterwards: moving the destructive action away from common ones, styling it differently and confirming it (or offering undo) prevents accidental deletion. Messages or help pages after the fact do not stop the error, and smaller targets make mis-clicks more likely by Fitts's law.
729. **B** — Heuristic evaluation is a discount method that needs no users; Nielsen recommends three to five evaluators because one finds only about a third of the problems. The other options need users, live traffic or a release, which the constraints rule out.
730. **B** — Throwaway prototypes are meant for unclear requirements; once the needs are understood, the quick-and-dirty code is discarded and the real system is engineered to the required reliability. Evolving prototype code into a safety-relevant product risks carrying shortcuts into production.
731. **C** — Q blocks a core task and loses orders, a usability catastrophe (4). R is a minor problem (around 2) and P is cosmetic (1). Severity reflects frequency, impact and persistence, so problems are not equal just because each violates a heuristic.
732. **B** — Users stop once their goal (cash) is met, a post-completion error. Ordering the steps so the card comes back before the cash makes the error impossible (a forcing function), which applies error prevention and Shneiderman's closure rule. Bigger messages or notices still depend on attention.
733. **A** — The evidence points to unclear requirements (visibility and error prevention) and punishing error recovery (losing data breaks recovery from errors). Clear required markers, inline validation and preserving input fix exactly those problems. Colour or a manual does not touch the causes.

734. **D** — With little time and an untested concept, the goal is fast feedback on the idea and flow. Low-fidelity prototypes are quick and cheap and invite honest criticism. Coding, schemas or style guides commit effort before the concept is known to work.
735. **C** — Progressive disclosure shows the frequent options for novices and keeps advanced ones one step away (plus search) for experts, satisfying both heuristics. Deleting settings removes power users' functions, a smaller font worsens clutter, and command-line flags force recall.
736. **A** — Frequent confirmation dialogs become habit and get clicked through. A short undo window keeps the common path fast and still lets users reverse mistakes (Shneiderman's easy reversal, Nielsen's user control). It does not stop mistakes, it makes them recoverable, and it relies on feedback.
737. **C** — Automated tools catch only some issues, such as missing alt text or low contrast. Keyboard operability and meaningful labels on custom widgets need manual testing with assistive technology and real users. This reflects Shneiderman's universal usability rule.
738. **C** — In Y the system state (unsaved changes) is not visible and work is lost when users leave, violating visibility of status and closure. X applies changes visibly and keeps undo for mistakes, so it meets feedback and reversal principles. X still needs undo; it does not make it unnecessary.
739. **D** — Each extra unit of information competes with the relevant units and lowers their visibility. The heuristic is about keeping content focused, not about a particular visual style or colour count.
740. **B** — Nielsen says it is best if the system can be used without documentation, but help may be needed. When provided, it should be searchable, task-focused, concrete and not too large.
741. **B** — Memorability concerns returning users. Learnability is first-time use, efficiency is speed after learning, and satisfaction is how pleasant the design is.
742. **D** — Efficiency measures expert speed after learning. Errors, memorability and learnability are separate components of Nielsen's model.
743. **C** — An affordance is what actions are possible, such as a chair affording sitting. A signifier is the perceivable cue that communicates where the action should take place. Norman added signifiers because designers often confused the two.
744. **A** — With natural mapping, the spatial relationship between controls and effects is obvious, for example stove knobs laid out in the same pattern as the burners. Users then need no labels to know which control does what.
745. **A** — The gulf of execution concerns doing: can users work out how to perform the action they intend? The gulf of evaluation concerns understanding the result. Signifiers and mapping bridge execution; feedback bridges evaluation.
746. **B** — After the goal, execution has plan, specify and perform; evaluation has perceive, interpret and compare. Feedback and a clear conceptual model help users through the evaluation stages.
747. **D** — Constraints restrict what users can do, guiding them towards the right actions. Physical constraints work through shape; logical and cultural constraints also exist.
748. **A** — Think-aloud reveals why users act as they do, exposing misunderstandings that logs cannot show. It is a core method in formative usability testing.
749. **D** — Formative studies are diagnostic and iterative, often with few users. Summative studies judge a design against goals or competitors and usually need more participants and metrics.
750. **C** — Users sort cards naming content items into groups that make sense to them. In an open sort they name the groups; in a closed sort they use given categories.
751. **C** — A/B testing splits real traffic between variants and compares metrics such as conversion rate. It shows which version performs better but not why, so it is often paired with qualitative testing.
752. **A** — For every action there should be feedback. Frequent minor actions need modest responses, while rare major actions need more substantial ones.
753. **D** — Because short-term memory is limited, displays should be kept simple and users should not have to remember information from one screen to use on another.
754. **B** — Earlier editions said 'Support internal locus of control', meaning users should feel they are in charge of the system. 'Offer simple error handling' was the older wording of 'Prevent errors', a different rule.
755. **A** — Proximity explains why a label belongs to the field nearest to it. Similarity groups items that look alike, closure fills gaps in shapes, and continuity follows smooth lines.

756. **B** — Similarity groups elements that share visual properties such as colour, shape or size. Here the shared colour and underline mark the links as one kind of thing, regardless of position.
757. **D** — WCAG groups its guidelines under four principles: content must be perceivable, operable, understandable and robust enough to work with assistive technologies.
758. **C** — Large text is easier to read, so AA lowers the requirement to 3:1. 7:1 is the AAA level for normal text, and 4.5:1 is the AAA level for large text.
759. **B** — Users build mental models from what the system shows them (Norman's system image). If the design communicates the conceptual model poorly, the user's mental model will be wrong and errors follow.
760. **B** — Personas summarise user research into concrete archetypes so the team designs for specific needs rather than for an undefined 'everyone'.
761. **D** — Pressman's UI process analyses users, tasks and environment, designs the interface, builds it (often as a prototype) and validates it with users, repeating as needed.
762. **A** — Dragging a file to a folder is direct manipulation: the object is visible, the action is physical, and the result appears at once and can be undone.
763. **D** — log2(31 + 1) = log2(32) = 5, so T = 0.15 × 5 = 0.75 s. Multiplying 0.15 by 31 treats the law as linear, which it is not.
764. **B** — D/W + 1 = 15 + 1 = 16, and log2(16) = 4 bits. MT = 0.05 + 0.1 × 4 = 0.45 s. Forgetting the intercept a gives 0.40 s.
765. **A** — Before: log2(448/32 + 1) = log2(15) ≈ 3.91. After: log2(448/64 + 1) = log2(8) = 3. The drop is about 0.91 bits. It is not exactly 1 bit because of the +1 in the Shannon form.
766. **C** — $1 - 0.69^3 = 1 - 0.3285 = 0.6715$, about 67%. Adding 3 × 31% gives 93%, which ignores overlap between users. 84% is the result for 5 users.
767. **C** — Five users give about 84.4% and six give about 89.2%, both below 90%. Seven give $1 - 0.69^7 \approx 92.6\%$, so 7 is the smallest number that reaches 90%.
768. **A** — AA requires 4.5:1 for normal text and 3:1 for large text. 4.48:1 just misses the normal-text threshold, and WCAG does not round up, so it passes only for large text.
769. **A** — KLM adds the operator times: 0.4 + 1.35 + 1.1 + 0.2 = 3.05 s. Leaving out H gives 2.65 s. KLM predicts expert, error-free task time.
770. **D** — Error prevention means stopping the mistake before it happens. Showing the constraints up front and checking live prevents the invalid submit, rather than just reporting it better.
771. **B** — Users should not have to wonder whether different words mean the same thing. One term for one action, used everywhere, follows consistency and standards.
772. **C** — The system should speak the user's language, with familiar words rather than system-oriented terms. 'Save' matches how users think about the action.
773. **B** — Help should be easy to search, focused on tasks and concise. A huge unsearchable manual technically exists but fails the heuristic.
774. **C** — Users now recognise the right command from a visible list instead of recalling it from memory, which reduces memory load. Experts can still type quickly, so efficiency is kept.
775. **D** — The relationship between knobs and burners is unclear. Arranging the knobs in the same square layout as the burners gives a natural mapping, so no labels are needed.
776. **A** — The plate signals 'push' and the handle signals 'pull'. They are perceivable cues, so they are signifiers. They do not physically block the wrong action, so they are not constraints.
777. **C** — In an open card sort, participants both group the items and name the groups, which reveals their own vocabulary. In a closed sort, the categories are given in advance.
778. **A** — Splitting live users between two variants and comparing a metric is A/B testing. It needs real traffic and gives quantitative results.
779. **B** — High-fidelity prototypes look and behave close to the final product, which suits detailed testing and stakeholder sign-off. Low-fidelity and paper prototypes are rough sketches.
780. **A** — A vertical prototype implements a narrow slice of functionality in full depth. A horizontal prototype would show many features with little or no working behaviour behind them.

781. **C** — The confirmation marks the end of a sequence of actions, giving users a sense of completion and telling them they can move on. That is closure.
782. **C** — People group elements that are close together. Putting each label right next to its field, with more space between field groups, makes the pairs obvious.
783. **C** — Without alternative text, the button has no accessible name, so a screen reader announces only 'button'. alt="Delete" (or an aria-label on the button) gives it a name. Size and borders help sighted users only.
784. **D** — Comparing task times with statistical confidence needs a summative study, a larger sample and clean timing. Think-aloud slows users, and expert methods give no measured user times.
785. **A** — Heuristic evaluation predicts problems; user testing observes them. When most users fail a task, its severity is high whatever the experts guessed. Severity, not count, sets the priority.
786. **D** — Users learn to click OK in routine dialogs without reading. For a rare, severe and irreversible action, a confirmation that requires attention and names the target prevents slips. It deliberately trades speed for safety.
787. **C** — Because the cursor cannot overshoot the edge, users can throw the pointer at it without slowing down. In Fitts's terms the effective width along the movement is huge, so movement time drops.
788. **C** — Without concrete user models, each decision is made for an imaginary average user who does not exist. Personas for the main segments show where needs differ and must be designed for.
789. **C** — Nielsen's 'consistency and standards' includes following platform conventions that users already know. Matching each platform's navigation patterns usually beats pixel-identical screens across platforms.
790. **B** — Throwaway prototypes are built quickly to clarify requirements and usually ignore non-functional qualities such as reliability and maintainability. Shipping one locks those weaknesses into the product.
791. **A** — The method is valuable before the technology exists, but the wizard should follow rules that mimic the real system's limits. Otherwise users experience a smarter system than the one that ships.
792. **D** — WCAG's first principle is perceivable: non-text content needs a text alternative. Giving each button an accessible name lets the screen reader announce what it does.
793. **D** — Good error messages use plain language, name the problem precisely and suggest a solution. Only the chosen message does all three; the others are codes, vague or unhelpful.
794. **B** — Nielsen's five-user guidance assumes a fairly uniform user group. With distinct groups doing different tasks, each group needs its own participants, often three or more each.
795. **A** — Think-aloud is excellent for finding why problems happen, but verbalising adds load and changes behaviour. Reliable task times need a separate summative test without think-aloud.
796. **B** — A cognitive walkthrough steps through a specific task from a first-time user's view, which matches the need. Heuristic evaluation is broader but less focused on learnability of one critical task.
797. **B** — Nielsen's limits say delays over 10 seconds lose the user's attention, so they need percent-done feedback and a way to interrupt. A busy cursor suits delays of a few seconds only.
798. **D** — Users can perform the action, so execution is fine. They cannot tell whether it worked, which is the gulf of evaluation. Clear feedback about the system's state bridges it.
799. **D** — A coding standard sets shared rules for naming, layout, comments and safe constructs. Consistent code is quicker to read, review and maintain by anyone on the team. It does not guarantee speed and never replaces testing.
800. **A** — PEP 8 uses CapWords (also called PascalCase) for class names, lowercase_with_underscores for functions and variables, and UPPER_CASE_WITH_UNDERSCORES for module-level constants.
801. **C** — PEP 8 recommends 4 spaces per indentation level and prefers spaces to tabs. Mixing tabs and spaces for indentation is not allowed in Python 3.
802. **B** — Refactoring improves structure while keeping external behaviour the same. Adding features or fixing bugs changes behaviour, so it is not refactoring, even if it is done at the same time.
803. **A** — The driver types the code while the navigator reviews each line, thinks ahead and spots problems; the pair switch roles often. Author and moderator are roles in a formal inspection, not in pairing.
804. **A** — Static analysis tools (linters, type checkers, security scanners) inspect source code without executing it, catching problems such as unused variables, type errors or risky calls. Running the program with inputs is dynamic testing.

805. **A** — The moderator, a trained person who is not the author, organises the inspection and runs the meeting. The reader paraphrases the material, the recorder logs defects, and the author answers questions and later fixes them.
806. **D** — In a walkthrough the author leads colleagues through the code, often for understanding and feedback. An inspection is the most formal review: defined roles, preparation, checklists, entry and exit criteria and recorded defect metrics. Neither runs the code.
807. **C** — A code smell, such as a long method or duplicated code, is not a bug in itself and the code still works. It hints that the design may be hard to change, and it suggests where refactoring could help.
808. **C** — In shotgun surgery a single change is scattered across many classes, so it is easy to miss one. The usual cure is Move Method and Move Field to bring the related behaviour together. Divergent change is the opposite: one class changed for many different reasons.
809. **C** — DRY (Don't Repeat Yourself), from Hunt and Thomas, says each piece of knowledge or logic should live in one place. Duplicated logic means a fix or change must be repeated everywhere, and copies drift apart.
810. **B** — Ward Cunningham's metaphor: quick-and-dirty choices speed delivery now but add "interest" later as the code becomes harder to change. Refactoring pays the debt down.
811. **B** — Refactoring must preserve behaviour, and fast automated tests are how you confirm it after every small step. Fowler stresses this as the precondition for safe refactoring.
812. **B** — The reader walks the team through the material in their own words, which exposes misunderstandings. The recorder (scribe) logs defects, the author does the rework, and inspections are never used to appraise individuals.
813. **D** — Divergent change means one class has several responsibilities that change independently, a breach of single responsibility. The cure is Extract Class to split them. Shotgun surgery is the reverse: one change spread over many classes.
814. **C** — Sommerville's implementation chapter stresses reusing existing software, managing configurations (versions and builds) and developing on a host machine for a different target platform. Requirements and testing are covered in their own stages.
815. **A** — range(1, len(marks)) gives indices 1 and 2 only, so it adds 85 + 90 = 175 and skips the first mark (70). This off-by-one error is a classic code-review finding; looping directly over the list, or using sum(marks), avoids it.
816. **C** — An unexplained literal (a magic number) has been given a named constant, which documents its meaning and means a change is made in one place. The code below runs and prints the same value both ways.
817. **B** — PEP 8 names functions and variables in lowercase with words separated by underscores. CapWords is for classes and UPPER_CASE for constants; camelCase is not the Python convention.
818. **B** — Default argument values are evaluated once, when the function is defined, so every call without a basket shares the same list. The first call leaves 'pen' in it, so the second returns ['pen', 'ink']. The fix is basket=None and creating a new list inside.
819. **C** — 0.1 and 0.2 have no exact binary representation, so 0.1 + 0.2 is 0.30000000000000004 and the == test is False. Comparing with a tolerance (or math.isclose) gives True. Many coding standards ban == on floating-point values for this reason.
820. **B** — 6000 is valid, the user is not a member, but 6000 ≥ 5000 gives free shipping: 0. A total of 0 hits the first guard: invalid, even though the user is a member. 1200 for a non-member falls through to 250. Guard clauses return early, so their order matters.
821. **D** — The variable named sum shadows the built-in sum() function. The loop works and prints 6, but sum([4, 5]) then tries to call the integer 6. Linters warn about redefining built-ins, which is why coding standards forbid it.
822. **C** — Values that always travel together form a data clump. Grouping them into one object such as DateRange shortens parameter lists and gives a natural home for related behaviour such as contains(date).
823. **A** — address_label uses only the Customer's fields and none of Invoice's own data, so it "envies" Customer. Moving it to Customer (and calling customer.address_label()) puts behaviour next to the data it uses. The code runs and prints the label.
824. **D** — Repeated type-switching is the switch-statements smell. Making Circle, Square and Triangle subclasses that each implement area(), perimeter() and draw() means a new shape is one new class, with no edits to existing conditionals.

825. **B** — PEP 8 says comparisons to singletons such as None should use is or is not, never the equality operators. == can be overridden by a class's __eq__, while is checks identity. A single = is assignment and is a syntax error in an if.

826. **D** — Each lambda looks up i when it is called, not when it is created (late binding). By then the comprehension has finished and i is 2, so all three return 2. Binding the value as a default, lambda i=i: i, gives [0, 1, 2].

827. **C** — Duplicated code is the most common smell, and Extract Method is the standard cure: move the lines into one well-named function and call it from each place. Future fixes are then made once, in line with DRY.

828. **B** — Running python -O sets __debug__ to False and strips assert statements, so validation built on them vanishes in optimised runs. Use assertions for internal invariants and raise a proper exception, such as ValueError, for bad input. The code below shows __debug__ is True in a normal run.

829. **A** — For 5 and 11 units both versions agree. For 10 units the total is exactly 1000: the original uses > 1000 and charges 1000.0, but the extracted discount() uses >= 1000 and charges 900.0. The change altered behaviour at the boundary, so it is not a refactoring; a boundary test would have caught it.

830. **D** — The bare except turns both bad inputs ('abc' and '') into 0, so the average is (21 + 0 + 0 + 19) / 4 = 10.0. The program runs but gives a wrong answer silently. Catch the specific ValueError and skip or report bad values instead of hiding them.

831. **B** — The f-string pastes user input into the SQL, so the input turns the WHERE clause into name = 'x' OR '1'='1', which is true for both rows. This is SQL injection, a security defect that outranks any style comment. Use a parameterised query: db.execute("... WHERE name = ?", (name,)).

832. **D** — Reviewers cannot inspect thousands of lines carefully at once, so attention drops and defects slip through. Small, focused changes with a checklist keep each review thorough. Adding more reviewers to huge changes spreads responsibility thinner rather than fixing the size problem.

833. **C** — Mechanical style rules are best enforced by tools (for example a formatter such as Black and a linter such as flake8) that run on every commit. That ends the debates and frees reviewers to look at design, logic and security, where human judgement is needed.

834. **A** — Without tests you cannot tell whether a change preserves behaviour. Feathers and Fowler recommend first pinning down current behaviour with characterisation tests, then refactoring in small, verified steps. Big rewrites and untested refactoring risk breaking hidden behaviour.

835. **A** — Good comments explain why, which the code cannot say by itself. The first records the reason for the value. The others repeat what the code already shows or say nothing, and they add noise that can drift out of date.

836. **D** — The logic is duplicated, so the fix had to be repeated and two copies were missed. Extracting one shared function means the next fix happens once, applying DRY and the Extract Method refactoring.

837. **A** — This is a large class (often called a God class) with several reasons to change. Extract Class splits it by responsibility, for example FeeCalculator, Mailer and ReportGenerator, so each has one reason to change. Renaming or commenting leaves the structure untouched.

838. **D** — Readable code is cheaper to maintain, and Knuth's warning about premature optimisation applies: a function run once a day is not a bottleneck. Optimise where profiling shows the time actually goes, and keep tests to protect correctness.

839. **A** — PEP 8 writes constants in capital letters with underscores between words, such as MAX_CREDIT_HOURS. camelCase is not a PEP 8 style, CapWords is for classes, and lowercase_with_underscores is for functions and variables.

840. **D** — PEP 8 limits code lines to 79 characters, and docstrings and comments to 72. Teams may agree a longer limit, but the PEP 8 default is 79, so side-by-side diffs and narrow editors stay readable.

841. **B** — YAGNI stands for 'You Aren't Gonna Need It', an Extreme Programming rule: do not build features or hooks for imagined future needs. Unused code still has to be read, tested and maintained. A single authoritative copy of knowledge is DRY, a different principle.

842. **B** — Defensive programming means writing code that does not trust its inputs: it validates arguments, checks preconditions and handles error cases explicitly, so a problem is caught close to where it starts. Silently hiding errors or returning defaults has the opposite effect, because it lets bad data travel further.

843. **A** — A linter is a static analysis tool: it reads the code without running it and flags problems such as unused variables, undefined names and breaches of the coding standard. Running tests and measuring coverage is dynamic, and a linter does not compile or encrypt anything.

844. **C** — Continuous integration means integrating small changes into the shared mainline frequently, at least daily, with every change built and tested automatically. Problems then show up within minutes and are easy to trace to one small change, instead of piling up in a painful late merge.
845. **B** — TDD repeats a short cycle: write a small test that fails (red), write the simplest code that makes it pass (green), then refactor the code while the tests stay green. Writing the test first means every piece of code is driven by a test that was seen to fail.
846. **B** — A checklist lists frequent defect types, such as uninitialised variables, off-by-one loop bounds or unchecked return values, built from past defect data. It steers each inspector towards likely problems during preparation and the meeting. It supports careful reading rather than replacing it.
847. **A** — The recorder logs each defect raised in the meeting, with its location, type and severity, so the author can fix it during rework. The moderator leads the inspection, the reader paraphrases the material aloud, and the author does the rework.
848. **A** — Fagan's stages run planning, overview, preparation, inspection meeting, rework and follow-up. After the meeting, the author fixes the logged defects (rework). Follow-up comes after that, when the moderator checks that every defect was resolved.
849. **D** — In preparation, each inspector works individually through the code or document, usually with a checklist, so they arrive at the meeting already familiar with it. Fixing defects is rework, checking fixes is follow-up, and logging defects together is the inspection meeting itself.
850. **C** — Sommerville notes that production software is rarely built on the machine where it will run. Developers use a host system with an IDE, compilers and debuggers, then deploy to a target platform, which may have a different processor, operating system or memory. Simulators and cross-compilers bridge the gap.
851. **C** — Problem tracking (issue tracking) records bug reports and other problems, who is fixing each one and when it was fixed. Version management keeps track of versions of components, system integration defines which versions make up each build, and release management prepares versions for customers.
852. **D** — PEP 8 groups imports as standard library first (for example os, sys), then related third-party packages (for example requests), then the project's own modules, with a blank line between groups. Readers can then see at a glance what a module depends on.
853. **B** — Primitive obsession is using basic types for ideas that deserve their own small class. A Money or Cnic class (Fowler's Replace Primitive with Object) can hold the validation in one place. Feature envy is about a method using another class's data, and the other two smells concern unneeded classes.
854. **D** — A message chain is a client navigating a long series of objects to reach what it wants, so it depends on the structure of every link. Fowler's fix is Hide Delegate, giving the client one method to call, in line with the Law of Demeter. Middle man is the opposite problem: a class that only delegates.
855. **D** — A bare except: catches every exception, including KeyboardInterrupt and SystemExit, so a user cannot even stop the program with Ctrl+C, and genuine bugs such as a TypeError are silently swallowed. It is still valid syntax. PEP 8 says to catch specific exceptions, or at least use except Exception:.
856. **D** — In The Psychology of Computer Programming, Weinberg argued that programmers who see their code as part of themselves resist criticism and hide defects. Egoless programming treats code as shared work, so reviews find defects without blame. This is the attitude formal reviews depend on.
857. **C** — Python type hints are not enforced by the interpreter: area(2.5) runs normally. The hints are stored in the function's __annotations__ and used by static checkers such as mypy, by IDEs and by readers. Teams get type safety by running the checker in CI.
858. **C** — The rule of three, which Fowler credits to Don Roberts, avoids both premature abstraction and endless copying. One copy is fine, two copies are tolerated, and at the third similar piece of code the pattern is clear enough to extract into one shared abstraction.
859. **D** — Sommerville lists four reuse levels: the abstraction level (reusing knowledge such as design patterns), the object level (library objects), the component level (collections of objects and classes, for example a UI framework) and the system level (whole application systems, configured for use).
860. **B** — Speculative generality is machinery built for imagined future needs. It makes code harder to understand and maintain without paying back. Fowler's cures include Collapse Hierarchy, Inline Function and Remove Dead Code. It is the smell that YAGNI warns against.

861. **C** — Fowler calls comments a 'deodorant': when a block needs a long comment to explain what it does, that often signals code that should be clearer. Extracting a well-named function or renaming variables often removes the need. Comments that explain why something is done, or what is still uncertain, remain valuable.
862. **B** — days_overdue says exactly what the value means and follows PEP 8's lowercase_with_underscores style. Single letters and names such as temp2 force the reader to work out the meaning, and encoding the type into the name adds noise without meaning.
863. **C** — PEP 8 puts one space around the assignment operator and after each comma, and no space just inside parentheses or before the opening parenthesis of a call. Only result = compute(a, b) satisfies all of these.
864. **A** — Python strings are immutable, so name[0] = 'A' raises TypeError before print runs. To change a string you build a new one, for example name = 'A' + name[1:] or name.capitalize().
865. **A** — When subclasses share an identical method, Pull Up Method moves it into the common superclass so it exists once and both inherit it. Push Down Method does the reverse, and Inline Class or renaming leaves the duplication in place.
866. **B** — Defect density = defects / size in thousands of lines of code (KLOC). 4,500 lines is 4.5 KLOC, so 18 / 4.5 = 4.0 defects per KLOC. Dividing by 45 or inverting the ratio gives the wrong values.
867. **D** — range(2, 11, 3) starts at 2 and adds 3 each step while the value stays below the stop value 11: 2, 5, 8. 11 is excluded because the stop value is never included, which is a common off-by-one slip.
868. **A** — Python's // rounds down (towards minus infinity), so -7 // 2 is -4, not -3. The % result takes the sign of the divisor, so 7 % -2 is -1. C and Java truncate towards zero instead, which is why the difference is worth knowing when porting code.
869. **C** — 10 // 0 raises ZeroDivisionError, so the except block prints 'zero' and starts to return -1. The finally block always runs before the function actually returns, printing 'done'. Only then does print show -1. The else block runs only when no exception occurred.
870. **B** — and stops at the first False: check('a') prints a and returns False, so check('b') on the left is never called. or stops at the first True: check('b') prints b and returns True, so check('c') never runs. The arguments are evaluated before print shows False True.
871. **D** — Because visit() assigns to count, Python treats count as a local variable throughout the function. count += 1 then reads the local before it has a value, raising UnboundLocalError. Declaring global count would fix it, but passing and returning the value is usually cleaner.
872. **C** — Binary floats cannot store 0.1 exactly, so a is False. Decimals built from strings are exact, so b is True. Decimal(0.1) copies the float's binary error (0.1000000000000000055...), so c is False. Build Decimals from strings, never from floats.
873. **C** — Encapsulate Field replaces direct access to a data field with access through methods, here a Python property. The class now controls every change and can validate it, so an impossible CGPA of 5.2 is rejected. No new class is created and the field does not move.
874. **B** — Decompose Conditional extracts the condition and each branch into well-named functions, so the code reads like the business rule. No class hierarchy is introduced, no nesting is flattened into guard clauses, and no parameters are grouped.
875. **D** — Total known defects = 45 + 12 + 3 = 60. Inspection effectiveness = 45 / 60 = 75%. Leaving out the 3 field defects gives 45 / 57, about 79%, which overstates it. 25% is the share the inspection missed.
876. **A** — int(None) raises TypeError, not ValueError, because None is the wrong type rather than a badly formed number. ValueError's handler does not match, so the next handler, except Exception, catches it and prints 'other'. ValueError would come from something like int('12a').
877. **B** — With a string operand, + concatenates, giving '53'. - only works on numbers, so '5' becomes 5 and the result is 2. == converts types before comparing, so 1 == '1' is true; === also compares the type, so it is false. That hidden conversion is why standards ban ==.
878. **C** — Multiplying the outer list copies the reference, so both rows are the same inner list object. Changing grid[0][1] therefore shows up in both rows. Build independent rows with a comprehension: [[0] * 3 for _ in range(2)].
879. **D** — squares is a generator, which can be consumed only once. The first sum adds 0 + 1 + 4 + 9 = 14 and exhausts it, so the second sum sees no items and returns 0. Use a list, or recreate the generator, when the values are needed twice.

880. **C** — Removing items from a list while looping over it shifts the remaining items left, so the loop skips the item after each removal. 40 is removed and 55 slides into its slot, so 55 is skipped; then 30 is removed and 20 is skipped. Build a new list instead: [m for m in marks if m >= 50].

881. **A** — Python 3's round() uses round-half-to-even (banker's rounding): an exact .5 goes to the nearest even integer, so 2.5 gives 2, 3.5 gives 4 and -0.5 gives 0. For money with a 'round half up' rule, use Decimal.quantize with ROUND_HALF_UP.

882. **D** — Replace Temp with Query turns a temporary variable holding an expression into a method that computes it. Other methods can then reuse the logic, and it often makes Extract Method easier. Extract Variable goes the other way, and Split Temporary Variable is for a temp assigned for two purposes.

883. **B** — Security defects outrank style. Passing roll_no = x' OR '1'='1 makes the WHERE clause always true, and in our sqlite3 test it returned every student. The fix is cur.execute('... WHERE roll_no = ?', (roll_no,)), so the driver treats the input as data, never as SQL.

884. **D** — The empty list raises ZeroDivisionError and the mixed list raises TypeError (int + str), but the bare except turns both into 0. A data bug is silently reported as an average of 0. Handle the empty case explicitly and let unexpected errors surface, or catch only ZeroDivisionError.

885. **B** — if discount: treats 0 as false, but discount is not None treats 0 as a real value. For d = 0 the old function returns 'full price' and the new one 'discounted', so the first result is False. A refactoring must keep behaviour identical for every input; a test with 0 would catch the slip.

886. **A** — Safety-critical code justifies the most rigorous, defect-focused review. Inspections use defined roles, individual preparation, checklists and measured results, and they find defects that tests can miss. Lighter reviews suit lower-risk code.

887. **D** — items = [] in the class body creates one list owned by the class. self.items.append() finds that shared list through the instance, so both carts see both items. Assigning self.items = [] in __init__ gives each object its own list.

888. **B** — Python checks except clauses from top to bottom and uses the first match. ValueError is a subclass of Exception, so the general handler always wins and 'not a number' is unreachable. Put specific handlers before general ones.

889. **A** — 0.1 has no exact binary representation, and adding it ten times gives 0.9999999999999999, which we confirmed in Python. The exact-equality exit test therefore never succeeds. Counting with integers, or testing with < and a tolerance, gives a reliable exit.

890. **B** — PEP 8's section 'A Foolish Consistency is the Hobgoblin of Little Minds' says consistency within a project matters more than PEP 8, and consistency within one module or function matters most. It lists matching surrounding code as a good reason to break a PEP 8 rule. A wholesale rename belongs in its own change.

891. **D** — The code applies 15% (x 0.85) while the comment says 10%. Either could be the defect, so the requirement decides. A misleading comment is worse than none, so the fix must leave them consistent. Comments are still useful for explaining why.

892. **C** — Penguin inherits fly() but rejects it, so code that works for any Bird breaks for a Penguin, which also breaks the Liskov substitution principle. Fowler calls a subclass that does not want what it inherits 'refused bequest'. The fix is to remodel the hierarchy (for example a FlyingBird subtype) or use delegation.

893. **A** — Industry review studies, such as the SmartBear and Cisco study, found that defect detection drops sharply once reviewers go faster than a few hundred lines per hour, or keep going for much over an hour. The data show the fast reviews are skimming. Smaller changes and capped review rates restore effectiveness.

894. **C** — Reviews are static: they read the code, so they find problems that never produce a wrong output, such as dead code, misleading comments and standards breaches. Tests are dynamic and only reveal defects that cause failures. That is why Sommerville recommends both.

895. **A** — A 16-bit int holds at most 32,767, while a typical host int is 32 bits, so code that passes on the host can overflow on the target. Sommerville's host-target issue covers exactly this. Fixed-width types and testing on the target platform or a simulator expose the difference early.

896. **A** — In TDD a new test must fail first (red); seeing it fail proves it can detect the missing behaviour. A test that passes immediately may assert nothing useful, test the wrong thing, or duplicate existing behaviour.

897. **C** — Technical debt hurts when the code is touched. P is changed constantly, so every change pays for its duplication with extra effort and risk; cleaning it pays back quickly. Q's mess costs little while it stays stable.

898. **D** — ISTQB separates three ideas: a human action that produces a wrong result is an error (mistake); the wrong code it leaves behind is a defect (fault or bug); and a failure is the visible deviation when that defect is executed. The misreading is the human action, so it is the error.

899. **A** — Black-box (specification-based) techniques such as equivalence partitioning, boundary value analysis, decision tables and state transition testing treat the system as a box whose inside you cannot see. Tests come from the specification. Code, branches and control-flow graphs are the basis of white-box techniques.

900. **B** — Branch testing designs tests to exercise the outcomes of decisions in the code, so it needs knowledge of the internal structure. Equivalence partitioning, boundary value analysis and state transition testing are black-box techniques that work from the specification.

901. **B** — Passing tests only show that the tested cases behave correctly. Untested inputs and paths can still hide defects, so testing lowers the probability of undiscovered defects but is never proof of correctness. The other options contradict this principle, 'exhaustive testing is impossible' and 'early testing'.

902. **B** — Top-down integration starts from the top module, so the modules it calls may not exist yet. A stub is a simple stand-in that the module under test calls and that returns canned values. A driver is the opposite: it calls the module under test, and it is needed in bottom-up integration.

903. **D** — Alpha and beta testing are forms of acceptance testing for products released to many users. Alpha testing happens at the developer's site; beta testing is carried out by real users at their own locations, so it exposes the software to real environments and usage before the final release.

904. **C** — Confirmation testing (re-testing) checks that the original defect has been fixed by re-running the failed test. Regression testing is a separate activity: it re-runs other tests to make sure the change has not broken something that worked before.

905. **C** — Static testing examines work products without executing them: reviews, walkthroughs, inspections and static analysis by tools. A code review reads the code, so it is static. Unit tests, load tests and beta tests all execute the software, so they are dynamic testing.

906. **D** — Every executable statement lies on some branch outcome, so exercising every branch outcome also executes every statement. The reverse fails: an if with no else can have all its statements run by the True outcome while the False outcome is never taken. Branch coverage subsumes statement coverage.

907. **A** — If the same tests are repeated, they eventually stop finding new defects, just as pests become resistant to the same pesticide. CTFL v4.0 calls this 'tests wear out'. The remedy is to review and update the tests and add new ones. Repeating them still has value as regression checks.

908. **C** — A test oracle tells you what the correct output should be, so you can decide pass or fail. It can be the specification, a trusted older system, a calculation by hand or a domain expert. Without an oracle, a test can run but cannot be judged.

909. **C** — CTFL v4.0 lists component, component integration, system, system integration and acceptance testing. System integration testing focuses on interfaces between the system and other systems or external services, such as a payment gateway. Component integration testing checks interfaces between components inside one system.

910. **A** — Defect clustering says a small number of components usually contain most of the defects, an instance of the Pareto principle. Testers use it in risk-based testing: put more test effort into the modules where defects have clustered before.

911. **B** — The V-model pairs each development phase with a test level: user requirements with acceptance testing, system specification with system testing, architectural design with integration testing, and detailed module design with unit testing. Integration tests check that the components defined in the architecture work together through their interfaces.

912. **B** — Functional testing checks what the system does (does login accept a valid password?). Non-functional testing checks how well it does it: performance, load, usability, security, reliability. A response-time target under load is a performance requirement, so this is non-functional testing.

913. **B** — Mutation testing makes small changes (mutants) to the code and checks whether tests detect them. An equivalent mutant changes the syntax but not the behaviour, for example replacing i < n with i != n in a loop where i starts at 0, rises by 1 and n is never negative. No test can tell it apart, so it is excluded from the mutation score.

914. **D** — There is one valid partition (18 to 60) and two invalid partitions: values below 18 and values above 60. Each partition is assumed to behave the same throughout, so one representative test per partition is enough for EP. 43 is the number of valid values, not partitions.
915. **A** — 2-value BVA tests each boundary value and its nearest neighbour in the adjacent partition: 17 and 18 at the lower edge, 60 and 61 at the upper edge. 17, 18, 19, 59, 60, 61 is the 3-value version, and 18 and 60 alone miss the invalid side.
916. **A** — 3-value BVA takes each boundary plus both neighbours: 0, 1, 2 at the lower boundary and 99, 100, 101 at the upper boundary, giving 6 values. 2-value BVA would need 4 (0, 1, 100, 101).
917. **C** — There are three valid partitions (the three tax bands) and one invalid partition (negative income), so four tests, one per partition. Three forgets the invalid partition; six is the number of 2-value boundary values for the bands.
918. **A** — Each boundary between neighbouring partitions gives a pair: -1 and 0, 50,000 and 50,001, and 100,000 and 100,001. That is six values. Leaving out 50,001 misses the first value of the 5% band; mid-range values belong to EP, not BVA.
919. **D** — There are 6 executable statements: the first if, return "invalid", result = "fail", the second if, result = "pass" and return result. With m = 70 the first condition is false, so only return "invalid" is skipped: 5/6 = 83.3%. We confirmed this by tracing the function in Python.
920. **A** — There are 2 decisions, so 4 outcomes. m = 70 takes the False outcome of the first if and the True outcome of the second if: 2 of 4 = 50%. Note that branch coverage (50%) is lower than statement coverage (83.3%) for the same test.
921. **A** — The outcomes needed are: first if True (e.g. m = -5), first if False with second if True (m = 70) and first if False with second if False (m = 30). An invalid input returns immediately, so it cannot also cover the second if. Brute-force search over candidate inputs confirms no pair reaches 100%, but three do.
922. **A** — V(G) = number of binary decisions + 1. There are two: the while condition and the if condition, so V(G) = 3. The control-flow graph gives the same answer: 7 edges - 6 nodes + 2 = 3. So 3 independent paths form a basis set.
923. **A** — McCabe's formula is V(G) = E - N + 2P, where P is the number of connected components (1 here). So V(G) = 11 - 9 + 2 = 4. Using E - N alone (2) or E - N + 1 (3) are common slips.
924. **D** — Each Boolean condition doubles the number of combinations, so a full table has 2^3 = 8 rules. The table can later be collapsed when some conditions do not affect the action, but the complete table starts with 8.
925. **C** — Valid-transition coverage requires every valid transition to be exercised at least once. The model lists six transitions. 4 is the number of states (Idle, CardIn, Authenticated, Blocked), which is the target for state coverage, a weaker criterion.
926. **D** — Mutation score = killed mutants / (total mutants - equivalent mutants) = 36 / (50 - 5) = 36/45 = 80%. Dividing by all 50 gives 72%, which unfairly counts mutants that no test could ever kill.
927. **B** — When A is tested first, the modules it calls directly (B and C) are replaced by stubs. D is called only by B, so it is not needed until B is integrated. Bottom-up integration would need drivers instead of stubs.
928. **B** — The existing test takes the True outcome of both ifs, so 2 of 4 outcomes are covered. The missing outcomes are weight > 10 False and express False. Only (5, False) supplies both at once. Running each option with the existing test in a coverage tracer gives 100%, 50%, 75% and 75%.
929. **B** — With total = 1500 every statement runs, including rate = 10, so statement coverage is 100%. The if has no else, so its False outcome (total below 1000) is never taken: branch coverage is 50%. A defect in the no-discount path would go unnoticed.
930. **C** — With n = 2 the loop runs twice: 2 is even (if True), 1 is odd (if False), then n reaches 0 and the while condition is False. All four outcomes are covered. Tracing each candidate in Python gives 100% for n = 2, 25% for n = 0 and n = -1, and 75% for n = 1.
931. **D** — 2-value BVA needs 0, 1, 100 and 101. These four values also fall in all three partitions: below range (0), valid (1 and 100) and above range (101). So one set of four values meets both criteria at once.
932. **D** — The failure is the observable deviation from the expected behaviour at run time: the wrong fee. The misreading is the error (mistake), and the wrong operator in the code is the defect (fault). A bug report is a record of the failure, not the failure itself.

933. **C** — Bottom-up integration starts with the lowest-level modules and uses drivers to call them, so a risky low-level module is exercised first. Top-down reaches it last and stubs it in the meantime. Big-bang tests everything together, which makes faults hard to locate.

934. **B** — Regression testing re-runs existing tests on unchanged areas to detect side effects of a change. The fix itself was confirmed, so confirmation testing was done. The unexpected breakage elsewhere is exactly what a regression suite is designed to catch.

935. **C** — The decisions are the for loop, if p > 100, elif p > 50 and if member: V(G) = 4 + 1 = 5. The control-flow graph agrees: 13 edges - 10 nodes + 2 = 5. V(G) is the number of independent paths in a basis set, and an upper bound on the tests needed for branch coverage.

936. **D** — One test with items [120, 70, 20] and member True covers all three price branches, the loop's enter and exit outcomes, and member True. A second test with member False covers the last outcome. Brute-force search in a coverage tracer confirms that 2 tests suffice and 1 does not. V(G) = 5 is only an upper bound.

937. **C** — Testing shows the presence of defects, not their absence. Even a function with two 16-bit integer inputs has 2^32 combinations, about 72 minutes at a million tests per second, and real systems are far larger, so exhaustive testing is impossible. More tests reduce risk, but no count proves correctness.

938. **C** — Testing every combination of inputs and preconditions is infeasible except for tiny programs: even two 16-bit inputs give over four billion cases. ISTQB's principle is that test techniques, risk and priorities should be used to focus the effort instead.

939. **C** — A defect in a requirement that is caught in a review costs little to fix. The same defect found after design, coding and release must be fixed in many places, and possibly in the field. Starting static and dynamic testing early, often called 'shift left', avoids that cost.

940. **C** — Alpha testing is done at the developing organisation's site, by potential users or an independent test team, before the software goes out. Beta testing is done by real users in their own environments. Both are forms of acceptance testing.

941. **C** — A smoke test is a short, broad check that the build is stable enough to be worth testing further, for example that the app starts, users can log in and the main screens open. If it fails, the build is rejected straight away. Regression testing is broader and checks for side effects of changes.

942. **C** — Boehm's classic pair: validation asks 'are we building the right product?' (does it meet the users' real needs), and verification asks 'are we building the product right?' (does it conform to its specification). Acceptance testing is mainly validation.

943. **B** — Severity measures the technical impact, such as a crash or data loss versus a cosmetic flaw. Priority is a business decision about fix order. They can differ: a crash in a rarely used admin report may be high severity but low priority.

944. **A** — The test basis is whatever the tests are derived from, such as requirements, user stories, design documents or the code itself. For black-box tests it is usually the specification; for white-box tests it is the structure of the code.

945. **D** — Each V-model test level matches a development phase on the left of the V. Acceptance testing checks the system against the user requirements, system testing against the system specification, integration testing against the architectural design and unit testing against the detailed design.

946. **C** — Testing shows that failures or defects exist. Debugging is the development activity of locating the cause of a failure, fixing it and checking the fix. The other three options are test activities: dynamic execution, test design and static testing.

947. **A** — Component testing checks a single unit, such as a function or class, in isolation, often with stubs or mocks replacing the parts it depends on. System testing covers the whole system, and acceptance and system integration testing work at much higher levels.

948. **B** — The context-dependence principle says there is no single right way to test: safety-critical, commercial, mobile and other systems need different techniques, rigour and documentation. The other principles describe defect distribution, test-suite ageing and the timing of testing.

949. **D** — The absence-of-defects fallacy warns that verification alone is not enough: a system can meet its specification with no defects and still fail users if the specification is wrong. Validation, for example through user involvement and acceptance testing, addresses this.

950. **C** — Big-bang integration joins all the modules together and tests them at once. It needs few stubs or drivers, but a failure could come from any of many interfaces, so fault localisation is slow. Incremental approaches add modules one at a time to avoid this.
951. **D** — Sandwich integration tests the upper layers top-down with stubs and the lower layers bottom-up with drivers at the same time, converging on a middle target layer. It gets an early view of the top-level control flow while the risky low-level modules are also tested early.
952. **D** — Exploratory testing is an experience-based technique in which test design, execution and learning happen together, often in time-boxed sessions guided by a charter. It is useful when specifications are poor or time is short, and it complements scripted tests.
953. **B** — Error guessing is an experience-based technique: tests target errors, defects and failures the tester expects from past experience, often from a list of likely faults (a fault attack). EP and BVA are derived from the specification, and statement testing from the code.
954. **B** — Both are performance testing. Load testing checks response times and throughput under expected and peak usage, such as 500 users. Stress testing pushes past the specified limits, or reduces resources, to see whether the system degrades gracefully and recovers.
955. **C** — The test pyramid, popularised by Mike Cohn, puts many fast, cheap, isolated unit tests at the base, fewer integration or service tests in the middle, and only a few slow, brittle end-to-end UI tests at the top.
956. **D** — ISTQB ranks independence from none (authors testing their own work) through peers in the same team and separate test teams, up to testers from outside the organisation. More independent testers bring different assumptions, so they are more likely to find defects the author is blind to.
957. **B** — Reviews and static analysis examine work products such as requirements and code without running them, so they point straight at the defect. Dynamic testing runs the software and observes failures; debugging is then needed to find the defect behind each one.
958. **B** — Exit criteria (part of the Definition of Done in Agile teams) define when testing at a level can stop, for example 100% of planned tests run, 90% branch coverage and no open high-severity defects. Entry criteria define when testing can start.
959. **D** — ISTQB groups maintenance-testing triggers as modification (fixes, enhancements, patches), migration (moving to another platform, or converting data) and retirement (for example testing data archiving and restore). Each needs testing of the change plus regression testing.
960. **C** — Both stand in for collaborators. A stub supplies fixed responses so the code under test can run (state verification). A mock is set up with expectations and lets the test check how it was called, for example that send_email was called once with a given address (behaviour verification).
961. **A** — MC/DC (required for the most critical avionics software under DO-178C) needs every condition to take both values, every decision to take both outcomes, and each condition to be shown to flip the outcome while the others stay fixed. It sits between decision coverage and full multiple-condition coverage.
962. **D** — Bottom-up integration starts from the modules that call nothing else, which here are B, D and E. Drivers stand in for their callers. Then C is integrated with D and E, and finally A joins. Starting from A with stubs is top-down.
963. **B** — Each 16-bit input has 2^16 = 65,536 values, and every value of one can pair with every value of the other: 65,536 × 65,536 = 2^32, about 4.3 billion. Adding instead of multiplying gives 131,072. This is why exhaustive testing is impossible in practice.
964. **C** — The only boundary is between the invalid partition (17 and below) and the valid one (18 and above). 2-value BVA takes the boundary value and its nearest neighbour across the boundary: 18 and 17. 19 is the inner neighbour used only by 3-value BVA.
965. **A** — When behaviour depends on the current state plus an event, state transition testing models the states, events and transitions and derives tests to cover them, including invalid events in a given state. Statement testing is white-box, and EP or pairwise testing ignore the history of earlier events.
966. **C** — The two assignments are on opposite sides of an if/else, so no single input can run both. One non-negative and one negative input, for example 5 and -3, execute all 4 statements. A line tracer confirms that any single input reaches only 75%.
967. **B** — With two true/false conditions there are 2^2 = 4 rules: (Y, Y), (Y, N), (N, Y) and (N, N). An or-rule is true in every column except (N, N), so 3 rules give free delivery. Reading 'or' as 'and' would give only 1.

968. **B** — The smallest step is 0.01, so the nearest neighbours outside the valid range are 0.00 and 1000.00. 2-value BVA tests each boundary plus its outside neighbour. Whole-number steps ignore the precision, the third set has no invalid values, and the six-value set is 3-value BVA.

969. **D** — A full state table has one cell for every state-event pair: 4 × 3 = 12. Five are valid transitions, so 12 - 5 = 7 cells are invalid. Testing them checks that the system correctly ignores or rejects events that should not happen in that state.

970. **C** — The 8 statements are price = 1000, if age < 12, price = 500, elif student, price = 700, if age >= 60, price -= 200 and return. For (30, True) only price = 500 and price -= 200 are skipped, so 6 of 8 = 75%. A Python line tracer gives the same result.

971. **A** — There are 3 decisions, so 6 outcomes. (5, False) takes age < 12 True; (30, True) takes age < 12 False and student True. Both take age >= 60 False. That covers 4 outcomes; student False and age >= 60 True are missed: 4/6 = 66.7%.

972. **B** — DDP = defects found by the test stage / (defects found by that stage + defects found later) = 160 / (160 + 40) = 80%. 25% is 40/160, 75% wrongly subtracts, and 20% is the share that escaped.

973. **A** — Each if can be taken or skipped independently, so there are 2 × 2 × 2 = 8 paths. Tracing all eight input combinations gives eight different paths. 4 is the cyclomatic complexity (3 decisions + 1), which counts a basis set of paths, not all of them.

974. **D** — For a simple loop with at most n passes, Pressman tests: skipping the loop entirely, one pass, two passes, m passes where m < n, and n - 1, n and n + 1 passes. This targets the off-by-one and boundary faults that loops typically hide.

975. **D** — With P = 2 connected components, V(G) = 20 - 18 + 2 × 2 = 6. Using P = 1 by habit gives 4. The formula adds 2 for each separately analysed component.

976. **B** — 0.1 + 0.2 is 0.30000000000000004 in binary floating point, so assertEqual fails. assertAlmostEqual rounds the difference to 7 decimal places by default, so it passes. Float results should be checked with a tolerance.

977. **C** — Each test may hold only one invalid value, so the 4 invalid partitions need 4 tests. In each of them the other field is valid, so both valid partitions are covered along the way, and a brute-force search confirms 4 is the minimum. A separate all-valid test (5) is common practice but not needed for coverage.

978. **A** — Seeding assumes real defects are found at the same rate as seeded ones: 15/20 = 75%. Estimated real total = 45 / 0.75 = 60, so about 60 - 45 = 15 remain. 60 is the estimated total, not the remainder, and 5 is the number of seeded defects still unfound.

979. **C** — Full combination testing needs 2^3 = 8 tests, but pairwise testing needs every value pair of every two parameters only. Each parameter pair has 4 value pairs, so at least 4 tests are needed, and 4 well-chosen rows cover all 12 pairs. A brute-force search confirms 4 is the minimum.

980. **C** — Each condition needs a pair of tests in which only it changes and the outcome changes. (T, T) gives True; (F, T) gives False, showing a's effect; (T, F) gives False, showing b's effect. These 3 tests suffice, and a search over all subsets confirms that 2 cannot. All 4 combinations is multiple-condition coverage.

981. **A** — Each condition takes both True and False across the two tests, so condition coverage is 100%. But True or False and False or True are both True, so the decision never takes its False outcome: 50%. This is why condition coverage does not imply decision coverage.

982. **D** — A test kills a mutant when the mutant gives a different result from the original. Only at exactly 2.5 do >= and > disagree: the original returns True and the mutant returns False, so the assertion fails on the mutant. This is why boundary tests are good at killing relational-operator mutants.

983. **A** — (25, 'remote') takes all three True outcomes; (2, 'local') takes all three False outcomes. Two tests reach 6 of 6 outcomes, and a coverage tracer confirms it. One test can never cover both outcomes of an if, so 2 is the minimum.

984. **D** — Coverage only shows that code ran. Without an assertion such as assert cart.total() == 1500, the test can fail only if total() raises an exception, so a wrong result goes unnoticed. Mutation testing would expose this: almost every mutant would survive.

985. **B** — The code uses > 85 where the spec says 85 and above. At 85 it returns B instead of A. 84, 70 and 69 all give the specified answer, which we confirmed by running each value against the spec. Off-by-one operator errors show up only exactly on the boundary.

986. **A** — Decision tables list every combination of conditions with the action each should produce, so missing or contradictory rules become visible. BVA tests the edges of single ranges, state transition testing suits event-driven behaviour, and statement testing is white-box.
987. **D** — Beta testing is acceptance testing by real users in their own environments before general release. It exposes the variety of devices, networks and usage that in-house testing cannot reproduce. Alpha testing happens at the developer's site.
988. **D** — Top-down integration starts from the top control modules, such as the UI and menus, with stubs standing in for the modules below. An early skeleton of the system is visible and can be demonstrated. Bottom-up gives a working top level only at the end.
989. **B** — This is a flaky test caused by shared state and test-order dependence. Each test should create the data it needs (fixtures, setUp/tearDown or transactions that roll back) so it gives the same result in any order. Skipping or re-running hides the problem.
990. **A** — The mutant returns 550, which is still below 1000, so the loose assertion cannot tell it from the original's 450. The mutant is not equivalent, since outputs differ. A surviving non-equivalent mutant points to a gap in the test's checking, which a precise expected value closes.
991. **D** — The defect has almost no technical impact, so severity is low. But it is visible to every user and damages the brand, so the business wants it fixed quickly: high priority. Severity and priority are set separately.
992. **A** — If the age check rejects the form first, the email validation may never run, so a defect in it is hidden. This is defect masking. EP practice is to test each invalid partition with all other inputs valid. Valid partitions, by contrast, can be combined.
993. **A** — (90, 6) takes both True outcomes and (50, 1) both False outcomes, so branch coverage is 100%, as the tracer confirms. Only 2 of 4 paths run. rate = 15 overwrites rather than adds, so (50, 6) returns 15 against a specified 5. Path-oriented tests catch it.
994. **B** — Regression test selection uses impact analysis to find what the change can affect, here the function and its callers, and tests those first, then spends the remaining time on other high-risk areas. Confirmation alone misses side effects, and running tests in file order ignores risk.
995. **A** — i starts at 0 and rises by exactly 1, so it reaches len(items) exactly and stops; != behaves the same as < for every input. No test can kill this mutant, which is why equivalent mutants are excluded from the mutation score. We ran all four mutants against 300 random lists to confirm.
996. **B** — When the same person writes both code and tests from the same misunderstanding, the tests check the wrong expectation and pass. Independent testing from the requirements, or reviewing the requirement itself, brings a different view that can expose the error.
997. **A** — The team has an inverted pyramid (the 'ice-cream cone'). UI tests are slow and brittle, so feedback is late and noisy. Following the test pyramid, most logic is checked by fast, isolated unit tests, with fewer integration tests and a thin layer of end-to-end tests.
998. **D** — Corrective maintenance is the reactive modification of software after delivery to correct problems that have been discovered, as defined in ISO/IEC 14764. A crash reported by a user is a discovered failure, so fixing it is corrective.
999. **C** — Adaptive maintenance keeps a software product usable in a changed or changing environment, such as a new operating system, database version, hardware or external interface. Nothing was broken in the software itself; the environment changed around it. Sommerville calls this environmental adaptation.
1000. **A** — Perfective maintenance provides enhancements for users and improves attributes such as performance or maintainability. The report was not failing and the environment did not change, so speeding it up is perfective.
1001. **B** — Preventive maintenance detects and corrects latent faults before they become operational failures. The overflow has not caused a failure yet, so the fix is preventive. Fixing it after users had seen wrong dates would have been corrective.
1002. **C** — Sommerville describes legacy systems as older systems that rely on languages and technology no longer used for new development, yet remain essential to the business. Replacing them is risky because business rules are buried in the code and documentation is often missing.
1003. **A** — ISO/IEC 14764, also summarised in SWEBOK, puts corrective and preventive maintenance under correction, because both deal with faults (found or latent). Adaptive and perfective maintenance form enhancement, because both change the software to meet new needs or environments.

1004. **A** — Lehman's law of continuing change says a program used in a real-world environment must keep changing, because the environment changes. If it is not adapted, it becomes progressively less useful. This is why maintenance is unavoidable for real systems.

1005. **D** — Each change tends to add special cases and erode the original structure. Lehman's second law says complexity increases unless extra effort, such as refactoring and restructuring, is spent to control it. That effort does not add features, so it is often skipped, which makes later changes costlier.

1006. **D** — Lehman classified programs as S-type (formally specified, correct or not against the spec), P-type (practical problems whose solutions approximate reality) and E-type (part of a real-world environment they also change). The evolution laws describe E-type systems, such as banking or hospital software.

1007. **B** — The law of conservation of familiarity says the incremental change in each release is approximately constant, because everyone involved must keep understanding the system. Very large releases tend to bring many faults and are often followed by a smaller, corrective release.

1008. **C** — Reverse engineering is examination only: it recovers designs, models or documentation from an existing system, such as class diagrams from code. It does not alter the system. Forward engineering is the traditional process from high-level abstractions down to implementation.

1009. **A** — Re-engineering combines reverse engineering (understanding the old system) with alteration, such as restructuring, and forward engineering to implement the improved form. Its aim is to make a valuable system easier to maintain without the cost and risk of a full rewrite.

1010. **D** — Restructuring transforms one representation into another at the same level of abstraction while preserving external behaviour. Code restructuring makes programs easier to read and change. Refactoring is the object-oriented, small-step form of the same idea.

1011. **B** — Law VII, added in the 1990s, says quality appears to decline unless the system is actively maintained and adapted. Even without new bugs, a system that does not keep up with its environment looks worse to its users over time.

1012. **B** — The feedback system law says evolution is driven by feedback loops among users, developers, managers and the market. To improve the process significantly, these loops must be recognised and managed, not just the coding activity.

1013. **D** — Key factors include team stability, contractual responsibility (developers may have no incentive to build for maintainability), staff skills and program age and structure. When a new team takes over, much effort goes on understanding code written by others.

1014. **D** — The app's code was correct for the old platform; the environment changed. Modifying software to stay usable in a changed environment is adaptive maintenance.

1015. **B** — Adding a requested enhancement to working software is perfective maintenance in ISO/IEC 14764 and SWEBOK; Sommerville calls it functionality addition. Nothing failed and the environment did not change.

1016. **A** — ACT = (KLOC added + KLOC modified) / total KLOC = (9 + 6) / 120 = 0.125. In COCOMO the annual maintenance effort = ACT x development effort = 0.125 x 600 = 75 PM. Counting only added code gives 45 PM.

1017. **B** — ACT is the fraction of the code base touched in a year: (12 + 8) / 200 = 20/200 = 0.10. Using only the added code gives 0.06, and only the changed code gives 0.04.

1018. **C** — Maintenance over 10 years = 0.3 x 10 = 3.0 million. Lifetime cost = 2.0 + 3.0 = 5.0 million, so the share is 3.0/5.0 = 60%. 150% compares maintenance with development only, not with the lifetime total.

1019. **A** — Enhancement = adaptive + perfective = 4 + 8 = 12 of 20 = 60%. Correction = corrective + preventive = 6 + 2 = 8 = 40%. The latent-fault fixes are preventive, so they belong to correction, not enhancement.

1020. **A** — Follow the uses-links backwards from Tax: Pricing uses Tax; Orders and Billing use Pricing; UI uses Orders. Reports uses only DB, so it does not depend on Tax. A reverse-reachability search over the graph gives exactly {Pricing, Orders, Billing, UI}.

1021. **C** — Modules that directly or indirectly use Payment are Checkout (uses Payment) and Email (uses Checkout). Add Payment itself: 3 modules. Auth is used by Payment but does not depend on it, so a change in Payment cannot ripple into Auth, Cart or Admin.

1022. **D** — Low quality with high value means the system matters to the business but is expensive to maintain. Sommerville recommends re-engineering it, or replacing it if a suitable off-the-shelf system exists. Scrapping suits low-value systems, and normal maintenance suits high-quality, high-value ones.

1023. **C** — A system that is both expensive to keep (low quality) and of little use to the business (low value) should be scrapped. Re-engineering is spent on systems whose business value justifies the cost.

1024. **B** — Mean = (4 + 10 + 6 + 12 + 8) / 5 = 40/5 = 8 days. Mean time to change is a common process measure of maintainability: if it rises release after release, the code is getting harder to modify.
1025. **C** — Recovering design representations from existing code without modifying it is reverse engineering (design recovery). Refactoring and re-engineering change the code; forward engineering goes from design to implementation.
1026. **D** — Data re-engineering analyses and reorganises a program's data structures and values: cleaning, standardising formats and migrating to a new database. Source code translation converts code to another language, and modularisation regroups program parts.
1027. **C** — Pressman's cycle runs inventory analysis, document restructuring, reverse engineering, code restructuring, data restructuring and forward engineering. Inventory analysis comes first: it lists applications with size, age, business criticality and maintainability so the team can pick re-engineering candidates.
1028. **C** — A latent fault fixed before it causes a failure is preventive, not corrective. Corrective maintenance reacts to a discovered failure. The other three labels are right: a reported crash is corrective, a runtime platform change is adaptive and a requested feature is perfective.
1029. **D** — The average increment is about 32 modules, so 90 is nearly three times the norm. Conservation of familiarity says the change per release stays roughly constant because staff and users must keep understanding the system. Large jumps tend to introduce many faults and are usually followed by a corrective release.
1030. **A** — High value with low quality calls for re-engineering or replacement, but the no-outage constraint rules out a big-bang switch. An incremental approach replaces parts gradually behind stable interfaces (often called the strangler fig pattern), so risk and downtime stay small.
1031. **B** — Ledger and Export use Rates directly. Report uses Ledger, and Audit uses Report, so both depend on Rates indirectly. The Ledger-Rates cycle loops back to Rates and adds nothing new. Users is isolated. A reverse-reachability search confirms {Ledger, Report, Audit, Export}.
1032. **A** — The yearly saving is 100 - 40 = 60 PM. Break-even = 300 / 60 = 5 years. If the system is expected to be retired sooner than that, re-engineering does not pay off on cost alone.
1033. **D** — Refactoring improves internal structure in small, behaviour-preserving steps. Re-engineering is a larger effort that rebuilds a system in a new form, often with reverse engineering and data or architecture changes. Nothing in the environment changed, so it is not adaptive maintenance.
1034. **B** — Lehman's second law says complexity grows unless work is done to reduce it. Rising complexity with flat features is exactly that decay. The remedy is deliberate effort, such as refactoring and restructuring, that adds no features but restores structure.
1035. **C** — Program comprehension dominates maintenance on undocumented legacy code: the developer must find where a change goes and what it affects. Reverse engineering tools and redocumentation recover that knowledge and cut the cost of every later change.
1036. **A** — ISO/IEC 14764 defines emergency maintenance as an unscheduled modification performed to temporarily keep a system operational pending corrective maintenance. It reacts to an actual failure, so it belongs with corrective maintenance, not preventive.
1037. **B** — Impact analysis identifies everything a proposed change could affect, by tracing dependencies, so effort can be estimated and ripple effects tested. Inventory analysis ranks whole applications for re-engineering, not single changes.
1038. **A** — Lehman's sixth law, continuing growth, says functionality must keep growing because users' expectations grow. Continuing change is about adapting to the environment; self-regulation and conservation of familiarity describe the regular, bounded pattern of releases.
1039. **C** — S-type (specification) programs are fully defined by a formal specification, and correct means meeting that specification. E-type programs are embedded in a changing real world; P-type programs solve a precise problem whose solution can only be approximated.
1040. **C** — P-type (problem) programs tackle a precisely formulated problem where only an approximate solution is practical, and their acceptability depends on how good the approximation is. GCD and a regex-defined lexer are S-type; a hospital system embedded in real work is E-type.
1041. **B** — Forward engineering is the traditional direction of development: from requirements and design to implementation. Recovering designs is reverse engineering, same-level transformation is restructuring, and adding documents is redocumentation.

1042. **D** — Redocumentation produces alternative views at the same abstraction level, such as data-flow or control-flow diagrams and pretty-printed listings, without changing the code. It is regarded as the simplest and oldest form of reverse engineering.

1043. **C** — Chikofsky and Cross define design recovery as a subset of reverse engineering in which domain knowledge, external information and reasoning are added to what can be observed in the code, so that higher-level abstractions are recovered. Like all reverse engineering, it does not change the system.

1044. **A** — Shotgun surgery: a single change scatters edits across many places, so it is easy to miss one. Its mirror image is divergent change (one class changed for many reasons). Using another class's data is feature envy; a shared long parameter list suggests a parameter object.

1045. **A** — Parallel running lowers risk by keeping the trusted old system live while the new one proves it gives the same results. It costs extra effort, but a big-bang switch has no safety net.

1046. **C** — Wrapping is black-box modernisation: the legacy component keeps running unchanged, and a wrapper translates between it and new clients. It is quick and low-risk, but the old code's problems remain inside.

1047. **D** — Maintenance changes can have unexpected side effects elsewhere in the system. Regression testing re-runs earlier tests to confirm that existing behaviour still works. It complements impact analysis, which decides what is likely to be affected.

1048. **C** — The stages are initial development, evolution, servicing, phase-out and close-down. In servicing, the architecture has degraded or expertise has left, so only small patches are made. Major functional change belongs to the evolution stage.

1049. **D** — ISO/IEC 14764 lists process implementation, problem and modification analysis, modification implementation, maintenance review and acceptance, migration and retirement. Migration moves software to a new environment, while retirement takes it out of service.

1050. **C** — A change to one module can affect modules that depend on it, directly or indirectly. Tracing this ripple effect is the purpose of change impact analysis, and missing it is a common cause of new faults.

1051. **D** — Repeated studies find that most maintenance effort goes on enhancing the system for new needs and environments, not on fixing bugs. This is why the idea that maintenance is 'mostly bug fixing' is a classic trap.

1052. **B** — Lehman's third law says system attributes such as size, time between releases and number of reported errors show near-normal, predictable trends across releases. It rests on the feedback within the organisation and its environment.

1053. **D** — Lehman's fourth law says the effective global activity rate is largely invariant: adding many more staff does not raise output proportionally, because of communication overhead and the system's own complexity. Constant change per release is conservation of familiarity.

1054. **C** — Parnas called these 'lack of movement' (the software is not updated, so it fits its environment less well) and 'ignorant surgery' (changes that break the original design concept). Software does not wear out physically, but it ages in these ways.

1055. **A** — Sommerville's legacy system layers are system hardware, support software, application software, application data, business processes, and business policies and rules. Changing one layer often forces changes in the others, which is why legacy change is hard.

1056. **D** — In a big-bang replacement, the whole new system goes live on a single date. If something hidden in the old system was missed, there is no gradual path or parallel system to fall back on. Incremental replacement and parallel running exist to reduce this risk.

1057. **D** — Brooks's top-down model has the maintainer form domain-based hypotheses and verify them in the code. Shneiderman and Mayer's bottom-up model builds understanding by chunking statements into larger meanings. Maintainers usually mix the two.

1058. **B** — A traceability matrix links each requirement to the artefacts that implement and test it. When a requirement changes, maintainers can follow the links to find what to change and retest, which is the core of impact analysis.

1059. **A** — By definition, refactoring changes structure without changing external behaviour. Running the tests after each small step gives evidence that behaviour was preserved, and pinpoints the step that broke something.

1060. **B** — ISO/IEC 25010 splits maintainability into modularity, reusability, analysability, modifiability and testability. Fault tolerance belongs to reliability, learnability to usability, and time behaviour to performance efficiency.

1061. **C** — The trigger is a change in the system's environment (an external API it depends on), not a failure or a user request for improvement. Changes that keep software usable in a changed environment are adaptive.

1062. **A** — SWEBOK, following ISO/IEC 14764, lists improving maintainability under perfective maintenance. Pressman and many lecture notes would call the same work preventive, which is why the stem names the convention. Nothing failed, so it is not corrective, and the environment did not change, so it is not adaptive.

1063. **B** — ACT = (KLOC added + KLOC modified) / total KLOC = (20 + 10) / 250 = 0.12. Using only the added code gives 0.08, and only the modified code gives 0.04.

1064. **D** — ACT = (8 + 4.8) / 160 = 0.08. Annual maintenance effort = ACT x development effort = 0.08 x 800 = 64 PM. Counting only added code gives 0.05 x 800 = 40 PM.

1065. **A** — ACT = (18 + 12) / 300 = 0.10. Annual effort = 0.10 x 1,200 = 120 PM. One full-time person supplies 12 person-months a year, so 120 / 12 = 10 people. 120 is the effort in PM, not the head count.

1066. **A** — Maintenance = 0.25 x 12 = 3.0 million. Lifetime cost = 1.5 + 3.0 = 4.5 million, so maintenance is 3.0 / 4.5 = 66.7%. This matches the textbook point that maintenance usually exceeds development cost for long-lived systems.

1067. **B** — Impact analysis follows dependants transitively: Checkout and Invoice use Currency directly, Cart uses Checkout, Report uses Invoice, and Dashboard uses Report. Login and Session do not depend on Currency at all, so they are outside the impact set.

1068. **C** — Modules that depend on C, directly or indirectly: B and E (direct), then A (via B) and F (via E). With C itself that is 5: A, B, C, E, F. D is used by C but does not depend on it, and G and H reach D, not C.

1069. **D** — Sommerville's re-engineering process includes source code translation, reverse engineering, program structure improvement, program modularisation and data re-engineering. Converting code from one language to another is source code translation; reverse engineering only analyses the program.

1070. **B** — A method more interested in another class's data than its own shows feature envy. Fowler's remedy is Move Method, putting the behaviour next to the data it uses, which improves cohesion and lowers coupling.

1071. **B** — The function has a long parameter list, and four of its parameters (street, city, postcode, country) always travel together as a data clump. Introduce Parameter Object bundles them into an Address object, shortening every call and giving the concept a name.

1072. **B** — The unexplained literal 1.17 became a named constant GST_RATE, which is Replace Magic Number with Symbolic Constant. Behaviour is unchanged: round(100 x 1.17, 2) and round(100 x (1 + 0.17), 2) both print 117.0.

1073. **C** — High quality means maintenance is cheap, but low value means it is not worth investing in. Sommerville suggests replacing it with a COTS system, scrapping it, or keeping it going cheaply. Re-engineering suits high value with low quality.

1074. **A** — High business value with high system quality is the best quadrant: the system is valuable and cheap to change, so it stays in service with normal maintenance. Re-engineering is aimed at high-value systems of low quality.

1075. **B** — 5.2 ln(1000) = 35.92; 0.23 x 10 = 2.30; 16.2 ln(200) = 85.83. MI = 171 - 35.92 - 2.30 - 85.83 = 46.95, about 47. On the common scale, values below about 65 suggest the module is hard to maintain. 133 forgets the LOC term.

1076. **B** — Comprehension = 50% of 2,400 = 1,200 hours. A 30% cut saves 0.30 x 1,200 = 360 hours. Applying 30% to the whole 2,400 gives 720 hours, which overstates the saving.

1077. **D** — Keep: 120 x 5 = 600k. Replace: 500 + 30 x 5 = 650k. Keeping is 50k cheaper. Replacement saves 90k a year, so it only breaks even after 500 / 90 = about 5.6 years, longer than the remaining life.

1078. **D** — ACT = annual effort / development effort = 90 / 600 = 0.15. Since ACT = (added + modified) / total, the changed code is 0.15 x 150 = 22.5 KLOC a year. 0.15 is the ratio itself, not a size.

1079. **C** — The correction class is corrective plus preventive. Emergency patches are a temporary form of corrective maintenance, so correction = 10 + 3 + 5 = 18, and 18 / 40 = 45%. Leaving out the emergency patches gives 37.5%; leaving out preventive gives 32.5%; 55% is the enhancement class.

1080. **A** — Fowler's strangler fig pattern grows the new system around the old one and gradually takes over its functions, so the business never depends on a single risky cut-over. Big-bang replacement does the opposite.

1081. **B** — Any n greater than 20 is also greater than 10, so the first branch always catches it and the elif can never run. The function only ever returns "big" or "small". If "huge" was intended, the fix is to test n > 20 first, a behaviour change that needs a change request.

1082. **C** — The team recovered the design (reverse engineering), altered it, and implemented a new form (forward engineering). That combination is re-engineering in Chikofsky and Cross's taxonomy.

1083. **C** — The original guards the empty list and returns 0; the new one divides by len([]) = 0 and raises ZeroDivisionError. Refactoring must keep behaviour for every input, so one matching case is not proof. A test for the empty list would have caught it.

1084. **A** — An E-type system's quality is judged against a changing real world. If the system does not adapt, the first law (continuing change) and the seventh (declining quality) predict that it becomes less satisfactory, even with frozen code.

1085. **D** — The fourth law says the effective work rate on an evolving system is roughly invariant to the resources applied. Extra people add communication and coordination overhead, and the system's complexity limits how fast it can change safely.

1086. **A** — A: 25 x 5 - 60 = 65 PM. B: 70 x 5 - 240 = 110 PM. A pays back sooner (2.4 against 3.4 years), but over the full 5 years B saves more. Payback time and total net benefit are different questions.

1087. **D** — Alerts -> Report -> Stats -> Parser, so Alerts and Report are transitive dependants. Impact analysis has to follow the ripple effect through every level; retesting only direct users missed two modules.

1088. **B** — ISO/IEC 14764 defines preventive maintenance as detecting and correcting latent faults before they become operational faults. Nothing has failed yet, so it is not corrective. The flaw is in the product, not a change in the environment.

1089. **A** — Sommerville separates environment assessment from application assessment. Here the application is good and the environment is poor, so the problem is the platform. Migrating (adaptive maintenance) keeps the valuable, sound code and removes the risk.

1090. **C** — Drag = 15% x 40 = 6 points a sprint. Removing two-thirds saves 4 points a sprint. Payback = 30 / 4 = 7.5 sprints. Dividing by the full 6 points gives 5, which wrongly assumes all the drag disappears.

1091. **A** — Correction (corrective + preventive) is only 25%; enhancement (adaptive + perfective) is 75%. Better testing helps, but it addresses the smaller part. This matches the long-standing finding that enhancements dominate maintenance effort.

1092. **D** — In Chikofsky and Cross's terms, reverse engineering examines the subject system and produces representations of it without changing it. Delivering a new codebase adds alteration and forward engineering, which together make re-engineering. Mixing up the terms is a common exam trap.

1093. **B** — min(100.0, 50) returns the int 50, so member_price gives 1000 - 50 = 950; staff_price still takes 10%, giving 1000 - 100.0 = 900.0. Duplicated code means every fix must be found and repeated in each copy. Extracting one shared function removes that risk.

1094. **A** — ACT scales effort by how much code changes, but not by how hard that code is to understand. Missing documents and lost expertise raise comprehension effort, so Q will likely cost more. Boehm's intermediate COCOMO applies effort multipliers for such factors.

1095. **C** — High value with low quality (W) is the re-engineering case; low/low (X) is scrapped; high/high (Y) stays under normal maintenance; low value with high quality (Z) can be replaced, scrapped or kept while it is cheap.

1096. **D** — In the staged model, a system moves from evolution to servicing when major change becomes too difficult. Returning to evolution is hard: the knowledge and architectural integrity needed for big changes are gone, so a large change often justifies re-engineering or replacement.

1097. **C** — QA is process-oriented and preventive: it defines and checks the processes, standards and audits that should produce good software. QC is product-oriented and detective: testing, inspecting and measuring the actual product to find defects. Setting and auditing standards is QA.

1098. **A** — Verification checks that the product conforms to its specification (built right). Validation checks that it meets the user's real needs (the right product). Reviews and spec-based tests mainly verify; acceptance testing with users mainly validates.

1099. **C** — ISO 9001 sets requirements for a quality management system (QMS) and is generic: factories, hospitals and software firms can all use it. It does not prescribe code metrics. The five maturity levels belong to CMMI, not ISO 9001.

1100. **B** — CMMI's maturity levels are 1 Initial, 2 Managed, 3 Defined, 4 Quantitatively Managed and 5 Optimizing. 'Repeatable' was level 2 in the older SW-CMM, which CMMI replaced; it is a common trap.

1101. **A** — The moderator is a trained, neutral leader who plans the inspection, runs the meeting and checks the follow-up. The author wrote the work product, the reader paraphrases it during the meeting and the recorder (scribe) logs the defects found.

1102. **A** — The seven principles are customer focus, leadership, engagement of people, process approach, improvement, evidence-based decision making and relationship management. ISO 9001 does not demand zero defects or any particular development model.

1103. **A** — ISO 9001:2015 combines the process approach with the Plan-Do-Check-Act cycle and risk-based thinking. Red-Green-Refactor is the test-driven development loop and Build-Measure-Learn comes from Lean Startup.

1104. **B** — At level 3, processes are well characterised and understood across the organisation and described in standards, procedures and tools; projects tailor them. Level 2 manages processes per project, level 4 adds quantitative control, and level 1 is ad hoc.

1105. **D** — At level 4 the organisation sets quantitative objectives for quality and process performance and controls processes with statistical techniques. Level 5 builds on this with continuous improvement and causal analysis of defects.

1106. **C** — In IEEE 1028 terms, a walkthrough is typically led by the author, who guides the team through the work product to share understanding and find issues. An inspection is the most formal review: defined roles, entry and exit criteria, checklists, defect logging and metrics. Neither executes code.

1107. **C** — Appraisal costs pay for finding defects: inspections, reviews, testing and audits. Prevention costs stop defects being made (training, planning, standards). Internal failure costs are rework before release, and external failure costs arise after release.

1108. **C** — ISO/IEC 25010:2011 defines eight product quality characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. The 2023 revision adds safety and renames some, but maintainability remains. Profitability, velocity and LOC are not quality characteristics.

1109. **A** — McCall's triangle has product operation (correctness, reliability, efficiency, integrity, usability), product revision (maintainability, flexibility, testability) and product transition (portability, reusability, interoperability). Maintainability concerns how easily the product can be changed, so it is under revision.

1110. **D** — The staged representation groups process areas into maturity levels 1 to 5 and rates the organisation as a whole. The continuous representation lets an organisation choose process areas and rate each with a capability level (0 to 3 in CMMI v1.3), improving them in the order it prefers.

1111. **C** — ISO writes and publishes standards but does not certify anyone. Certification is done by independent certification bodies, usually accredited by a national accreditation body, after auditing the QMS. Self-declaration is possible but is not certification, and the CMMI Institute handles CMMI appraisals.

1112. **C** — IEEE 730 covers software quality assurance processes and plans. IEEE 1028 covers software reviews and audits, IEEE 829 covered test documentation (now largely replaced by ISO/IEC/IEEE 29119-3) and IEEE 830 covered SRS documents (replaced by ISO/IEC/IEEE 29148).

1113. **A** — Defect density = defects / size = 45 / 15 KLOC = 3 defects per KLOC. Inverting the ratio gives 0.33, which is KLOC per defect.

1114. **B** — DRE = E / (E + D), where E is defects found before release and D is defects found after. DRE = 90 / (90 + 10) = 0.90, so 90% of the defects were removed before users saw them.

1115. **C** — Availability = MTTF / (MTTF + MTTR) x 100 = 98 / 100 x 100 = 98%. The service is usable 98 hours out of every 100.

1116. **D** — Pressman defines MTBF = MTTF + MTTR = 450 + 50 = 500 hours: one complete cycle of running then being repaired. Availability here is 450/500 = 90%. Some reliability texts use MTBF to mean uptime between failures, so always check which definition a question uses.

1117. **C** — Availability = uptime / (uptime + repair time) = 480 / (480 + 20) = 480/500 = 96%. 95.8% comes from (480 - 20)/480, which wrongly subtracts repair time from uptime.

1118. **B** — Allowed downtime = (1 - 0.999) x 8,760 = 0.001 x 8,760 = 8.76 hours a year. 87.6 hours corresponds to 99%, and 0.876 hours to 99.99%.

1119. **A** — With a constant failure rate, lambda = 1/MTTF = 0.001 per hour, and R(100) = e^(-0.001 x 100) = e^(-0.1) = 0.905. Taking 1 - 0.1 = 0.900 is a linear approximation, not the exponential model.

1120. **D** — DPMO = defects / (units x opportunities) x 1,000,000 = 18 / 10,000 x 1,000,000 = 1,800. Ignoring the 5 opportunities gives 9,000. Six Sigma quality is 3.4 DPMO.

1121. **D** — Total cost of quality = 20 + 30 + 25 + 45 = 120k. Failure cost = internal + external = 25 + 45 = 70k, and 70/120 = 58.3%. 37.5% is external failure alone, and 41.7% is prevention plus appraisal.
1122. **A** — Densities: A = 12/4 = 3.0, B = 15/6 = 2.5, C = 9/2 = 4.5 defects per KLOC. B has the most defects in absolute terms but is the largest module, so its density is the lowest. C is the hotspot.
1123. **A** — Each inspector needs 1,200 / 200 = 6 hours. Four inspectors make 4 x 6 = 24 person-hours. 6 hours is the time for one inspector only.
1124. **B** — Using statistical process control, with control limits and quantitative objectives, is the hallmark of level 4. Level 3 has standard processes but does not require statistical management. Level 5 would add systematic causal analysis and continuous improvement.
1125. **D** — Project-level planning, requirements management and configuration management are level 2 practices. Without organisation-wide standard processes the firm has not reached level 3. It is above level 1 because projects are planned and controlled, not ad hoc.
1126. **D** — When testing starts, the defects still present are those testing finds plus those that escape to users: 50 + 10 = 60. Testing DRE = 50/60 = 83.3%. 93.3% is the overall pre-release DRE (140/150), not testing's share.
1127. **B** — Absolute counts ignore size. Normalising by KLOC gives X = 30/10 = 3 and Y = 10/2 = 5 defects per KLOC, so Y's code is more defect-dense. Density still depends on how hard each team tested, so treat it as a signal, not a verdict.
1128. **D** — QC examines the product to detect defects, and testing a build is product inspection. Writing a quality plan, training staff and auditing processes are QA activities, because they shape and check the process that produces the product.
1129. **B** — CMMI appraises process capability. Level 5 (Optimizing) means processes are quantitatively managed and continuously improved through causal analysis and innovation. That makes good outcomes more likely and predictable, but it does not certify any single product as defect-free.
1130. **D** — Availability = MTTF / (MTTF + MTTR). A = 1000/1010 = 99.01%; B = 500/502 = 99.60%. B fails twice as often but recovers five times faster, so it is down less overall. Reliability (time to failure) and availability are different measures.
1131. **A** — Inspections are the most formal review type in IEEE 1028: a trained moderator, defined roles, individual preparation with checklists, defect logging, follow-up and metrics. They find more defects than informal reviews, which suits safety-critical code.
1132. **B** — Before: 10 + 40 + 60 + 90 = 200. After: 30 + 40 + 40 + 40 = 150. The fall is 50/200 = 25%. Spending 20 more on prevention removed 70 in failure costs, the classic argument that prevention is cheaper than failure.
1133. **B** — ISO 9001 is cross-industry, and certificates come from accredited certification bodies, which makes it widely recognised. CMMI, from the CMMI Institute, rates process maturity through an appraisal (SCAMPI in v1.3) and is development-specific. Many firms hold both.
1134. **B** — DRE = in-house / (in-house + escaped). Release 1: 180/200 = 0.90. Release 2: 160/200 = 0.80. Both releases had 200 defects in total, but twice as many reached users in release 2. Finding fewer defects in-house here means weaker removal, not better code.
1135. **C** — Passing spec-based tests means verification succeeded: the product matches its specification. Users rejecting the workflow means the specification itself missed their real needs, a validation failure. Involving users earlier, through prototypes and reviews, catches this sooner.
1136. **D** — DPMO = defects / opportunities x 1,000,000. A = 50/10,000 x 10^6 = 5,000; B = 12/4,000 x 10^6 = 3,000. B has fewer defects per opportunity. Both are far from Six Sigma's 3.4 DPMO.
1137. **D** — Crosby's definition is conformance to requirements: a product is of quality if it meets its stated requirements, so the requirements must be clear. Juran's alternative, fitness for use, centres on the user's purpose.
1138. **A** — Juran defined quality as fitness for use, meaning the product serves the customer's real purpose. Crosby used conformance to requirements, and 'zero defects' was Crosby's performance standard, not a definition of quality.
1139. **C** — Garvin's views are transcendental (recognised but hard to define), user (fitness for purpose), manufacturing (conformance to specification), product (inherent measurable characteristics) and value-based (what the customer is willing to pay).
1140. **B** — ISO/IEC 25010 divides reliability into maturity, availability, fault tolerance and recoverability. Fault tolerance is the degree to which the system keeps operating as intended despite hardware or software faults.

1141. **D** — Usability's sub-characteristics include appropriateness recognisability, learnability, operability, user error protection, user interface aesthetics and accessibility. Learnability is how easily users learn to use the product.
1142. **C** — Quality in use describes outcomes for users in a real context: effectiveness, efficiency, satisfaction, freedom from risk and context coverage. Portability, compatibility and maintainability are product quality characteristics.
1143. **C** — McCall's product transition factors are portability, reusability and interoperability, all about adapting the software to new environments or uses. Correctness and usability are product operation; flexibility is product revision.
1144. **C** — Product operation covers correctness, reliability, efficiency, integrity and usability. Testability and flexibility are product revision factors, and portability is a product transition factor.
1145. **A** — Prevention costs are spent to stop defects being made: training, planning, standards and process improvement. Running tests is appraisal, fixing before release is internal failure, and post-release patches are external failure.
1146. **D** — External failure costs arise after the product reaches customers: complaint handling, help-desk work, patches, returns and lost reputation. Inspections are appraisal, standards are prevention, and rework before release is internal failure.
1147. **A** — At level 5, Optimizing, the organisation continually improves its processes using quantitative understanding, causal analysis and resolution, and organisational innovation. Level 4 controls processes statistically but does not yet focus on continual improvement.
1148. **D** — Level 1 organisations have no stable processes; results depend on talented people, and schedules and budgets are often missed. Planned and tracked projects mark level 2; a tailored standard process marks level 3; statistical control marks level 4.
1149. **B** — SCAMPI A is the most rigorous appraisal and the only class that yields a maturity or capability rating. Classes B and C are lighter and are used to check progress or find gaps. There is no Class D.
1150. **A** — The Fagan stages run planning, overview, individual preparation, inspection meeting, rework and follow-up. In rework the author fixes the logged defects, and in follow-up the moderator checks the fixes.
1151. **B** — The reader leads the team through the work product by paraphrasing it, which exposes misunderstandings. The recorder logs defects, the moderator plans and runs the inspection, and the author fixes defects later, in rework.
1152. **A** — The recorder logs every defect raised in the meeting, which gives the defect list for rework and the data for inspection metrics. The author answers questions and later fixes the defects.
1153. **A** — Pressman's FTR guidelines include: review the product, not the producer; set and keep an agenda; limit debate; point out problems but do not try to solve them all; take written notes; limit participants and insist on advance preparation; and use checklists.
1154. **D** — Pressman lists three outcomes: accept the work product as is, reject it because errors are severe (another review follows correction), or accept it provisionally when minor errors must be fixed without another review.
1155. **B** — Six Sigma allows 3.4 DPMO, which assumes the usual 1.5-sigma long-term shift. 233 DPMO corresponds to about five sigma, and 6,210 to about four sigma.
1156. **C** — DMAIC is Define the problem and goals, Measure the current process, Analyse root causes, Improve by removing them, and Control to keep the gains. DMADV is the variant for designing new processes.
1157. **D** — ISO/IEC/IEEE 90003 interprets ISO 9001's QMS requirements for the acquisition, supply, development, operation and maintenance of software. ISO/IEC 25010 is a product quality model, IEEE 1028 covers reviews and audits, and ISO/IEC 14764 covers maintenance.
1158. **C** — SPICE (Software Process Improvement and Capability dEtermination) assesses individual processes on capability levels 0 to 5, much like CMMI's continuous representation. Its role has passed to the ISO/IEC 33000 family.
1159. **D** — An audit is an independent examination of compliance. It checks whether the process the project said it would follow is actually being followed, and records any noncompliance. Testing and fixing defects are QC and development activities.
1160. **B** — Pressman's statistical SQA collects defect data, traces each defect to its cause, and uses the Pareto principle: roughly 80% of defects come from about 20% of causes. Fixing those vital few gives the biggest gain.
1161. **A** — Defect density = defects / KLOC = 36 / 24 = 1.5 defects per KLOC. Dividing KLOC by defects gives 0.67, which is the inverse.

1162. **C** — Allowed downtime = 0.0001 x 8,760 hours = 0.876 hours = about 52.6 minutes. 8.76 hours corresponds to 99.9%, and 5.3 minutes to 99.999%.
1163. **B** — Rework on defects found before the product reaches the customer is internal failure cost. The original test run is appraisal; the same defect found by a customer would make the fix an external failure cost.
1164. **A** — Phase DRE = E_i / (E_i + E_i+1), where E_i+1 counts errors from phase i found later. DRE = 24 / (24 + 6) = 0.80. 0.25 is 6 / 24, and 4.0 is 24 / 6.
1165. **D** — In series every component must work, so R = 0.95 x 0.90 x 0.98 = 0.8379. A series system is always less reliable than its weakest part. 0.943 is the average, and 0.999 is what three parallel parts would give.
1166. **B** — Parallel: the system fails only if both fail. P(both fail) = 0.10 x 0.10 = 0.01, so R = 1 - 0.01 = 0.99. 0.81 is the series result (0.9 x 0.9).
1167. **C** — R(200) = e^(-200/400) = e^(-0.5) = 0.607. A common mistake is to take 1 - 200/400 = 0.50; another is to report the failure probability 1 - 0.607 = 0.39.
1168. **D** — POFOD = failed demands / total demands = 4 / 2,000 = 0.002. Sommerville uses POFOD for systems that are called on occasionally, such as protection systems; 500 is the inverse ratio.
1169. **B** — ROCOF = 6 / 1,500 = 0.004 failures per hour = 4 per 1,000 hours. 250 is the mean hours between failures (1,500 / 6), not a rate.
1170. **A** — Estimated total N = 24 x 18 / 12 = 36. Distinct defects found = 24 + 18 - 12 = 30. Remaining = 36 - 30 = 6. 36 is the estimated total, not the remainder, and 30 is what was found.
1171. **D** — Cost after release = 10 x 100 = 1,000; cost in review = 10 x 1 = 10. Extra cost = 990 units. This escalation is the economic case for early reviews.
1172. **A** — Meeting time = 1,800 / 150 = 12 hours. With a 2-hour cap, that needs 12 / 2 = 6 meetings. 12 is the hours, not the meetings. Keeping meetings short avoids fatigue, which lowers defect detection.
1173. **D** — Total effort = 4 x (3 + 2) = 20 person-hours. Effort per defect = 20 / 25 = 0.8 person-hours. Counting only one inspector gives 0.2; dividing defects by effort gives 1.25 defects per hour.
1174. **A** — DPMO = 58 / 250,000 x 1,000,000 = 232, which matches about 5 sigma. Wrong options come from slipping a power of ten in the DPMO calculation.
1175. **B** — Total = 15 + 35 + 30 + 70 = 150k. Conformance = 15 + 35 = 50k, so 50 / 150 = 33.3%. The remaining 66.7% is the cost of nonconformance (failure costs).
1176. **C** — Failure rate = 1 / MTTF = 1 / 250 per hour. Expected failures in 1,000 hours = 1,000 / 250 = 4. 0.25 is the inverse, 1 / 4.
1177. **C** — An organisational set of standard processes, tailored by each project, is the mark of level 3. Level 2 has managed projects but no common standard; level 4 needs statistical process control, which is missing here.
1178. **C** — DRE = E / (E + D), so E + D = 200 / 0.95 = 210.5 and D = 10.5, about 11. A common mistake is to take 5% of 200 = 10, but the 5% applies to the total, not to the found defects.
1179. **B** — 995 / (995 + MTTR) = 0.995 gives 995 + MTTR = 1,000, so MTTR = 5 hours. Taking 0.5% of 995 gives 4.975, which is close but uses the wrong base.
1180. **D** — Total = 100. The top two give 45 + 30 = 75 = 75%, so two causes are enough. Both are requirements problems, so a Pareto analysis would point improvement at requirements gathering and review.
1181. **B** — Pressman's SQA group ensures that deviations from the process are documented and handled, and records any noncompliance and reports it to senior management. Silent fixes hide the process problem, and weakening the plan removes the control.
1182. **B** — The model's central idea is that money spent on prevention and appraisal reduces the larger internal and external failure costs. Removing prevention because failures are low mistakes the effect for a reason to drop the cause.
1183. **C** — Inspection effectiveness (yield) = defects found / defects present. A: 40 / 50 = 80%. B: 30 / 32 = 93.75%. Raw counts reflect how many defects the module contained, not how good the inspection was.
1184. **A** — Availability: X = 100 / 100.1 = 99.90%, Y = 2,000 / 2,010 = 99.50%. Reliability over 1 hour: X = e^(-1/100) = 0.990, Y = e^(-1/2000) = 0.9995. When a failure during use is the danger, reliability (time to failure) matters more than quick repair.
1185. **D** — A (series) = 0.99 x 0.99 = 0.9801. B (parallel) = 1 - (0.03 x 0.03) = 0.9991. Series structures multiply reliabilities down, while redundancy multiplies failure probabilities down.

1186. **D** — The levels build on each other. Level 2 makes projects managed and level 3 makes processes defined and consistent; only then is the data stable enough for the statistical control expected at level 4. A dashboard on chaotic processes measures noise.

1187. **A** — When a measure becomes a target, people change the measurement instead of the reality. Review findings are a sign the QA process is working; hiding them loses the data that statistical SQA and process improvement depend on.

1188. **C** — Defect density per KLOC depends on language verbosity. Normalising by delivered functionality (function points) gives 30 / 200 = 0.15 defects per FP for both, so their quality is the same on that measure.

1189. **B** — N = 10 x 12 / 2 = 60. Distinct found = 10 + 12 - 2 = 20, leaving about 40. Low overlap between independent inspectors signals a large undiscovered population, which calls for re-inspection.

1190. **A** — Inspection studies show detection drops sharply as the review rate rises. Review 1 found 2 / 3 = 0.67 defects per KLOC and Review 2 found 12 / 0.6 = 20 per KLOC. With similar code, a tenfold difference in rate is the likely cause.

1191. **C** — Phase DRE = found / (found + escaped). Requirements: 20 / 50 = 0.40; design: 34 / 40 = 0.85; code: 40 / 50 = 0.80. Requirements errors also cost the most once they escape, so that review is the priority.

1192. **B** — Before: 20 + 50 + 80 + 150 = 300, failure 230 (76.7%). After: 50 + 60 + 50 + 60 = 220, failure 110 (50%). Spending 40 more on conformance saved 120 on failures.

1193. **D** — A: 50 / (1,000 x 10) x 10^6 = 5,000 DPMO, and 0.05 defects per unit. B: 60 / (400 x 50) x 10^6 = 3,000 DPMO, and 0.15 defects per unit. DPMO normalises for complexity (opportunities); defects per unit does not, so the two measures can disagree.

1194. **C** — Checking against users' real needs (are we building the right product?) is validation, so acceptance testing with real users is mislabelled. Checking work products against their specifications is verification.

1195. **A** — Any point outside the control limits signals a special cause that must be investigated. In review data, an unusually low defect density often means a rushed or poorly prepared review, so the cause must be found before drawing conclusions.

1196. **B** — Reviews range from desk checks to formal inspections, and formality should match the product's risk and value. A low-risk, tiny internal script does not justify five people's preparation and meeting time, while a safety-critical module would.

1197. **D** — A direct measure is counted straight from the product or process with no model in between, such as LOC, execution speed or defects found. Pressman treats quality attributes such as maintainability, reliability and usability as indirect measures, derived from other counts.

1198. **D** — VAF = 0.65 + 0.01 × ΣFi, where the 14 general system characteristics Fi are each rated 0 to 5. ΣFi therefore runs from 0 to 70, so VAF runs from 0.65 to 1.35, adjusting the unadjusted count by up to ±35%.

1199. **B** — Function point analysis counts external inputs (EI), external outputs (EO), external inquiries (EQ), internal logical files (ILF) and external interface files (EIF). It measures functionality from the user's view, so design-level counts such as classes are not part of it.

1200. **B** — V(G) counts linearly independent paths, which is the size of a basis set. It equals E − N + 2P for the control flow graph, the number of binary decisions plus 1, or the number of regions in a planar graph. In basis path testing it gives the number of test cases needed to cover a basis set.

1201. **B** — McCabe proposed 10 as a practical upper limit for a module's cyclomatic complexity. Above it, the module has many independent paths, is harder to test fully and is more likely to contain defects, so it is a candidate for refactoring. Some organisations allow up to 15 with justification.

1202. **C** — The categories have a meaningful order (Critical is worse than High) but the gaps between them are not equal or measurable, which defines an ordinal scale. A nominal scale has no order, while interval and ratio scales need equal-sized units.

1203. **B** — GQM works top-down: state a measurement goal, derive questions that show whether the goal is being met, then choose metrics that answer those questions. Starting from metrics leads to collecting data nobody uses.

1204. **B** — Defect density normalises the defect count by product size, usually per KLOC or per function point. This lets you compare modules or projects of different sizes. Dividing by developers, test cases or time measures something else.

1205. **D** — DIT (depth of inheritance tree) is the length of the path from a class to the root of its hierarchy. A deep class inherits more behaviour, which aids reuse but makes it harder to predict. NOC counts direct subclasses, CBO counts coupled classes and WMC sums method complexity.

1206. **A** — Both are user-recognisable groups of data. The difference is who maintains them: the application being counted adds, changes or deletes data in its ILFs, while an EIF is read from another application that maintains it. Storage medium and data type do not decide the classification.

1207. **A** — An external inquiry retrieves and presents data without derived data, calculations or updating an ILF. If the process calculated totals or other derived values, it would be an external output. An EI brings data into the boundary, and an ILF is a data function, not a process.

1208. **C** — Function points measure delivered functionality from the user's view, so you can count them from a requirements specification. LOC is only known after coding and changes with the language and coding style. Function points do not measure quality, and counting them needs a trained person.

1209. **D** — Product metrics describe the software itself: size, complexity, coupling, cohesion. Cyclomatic complexity is measured on the code. Fix time describes the process, while sprint effort and approved changes describe how the project is running.

1210. **D** — Halstead defines n1 and n2 as distinct operators and distinct operands, so vocabulary n = n1 + n2. The program length N = N1 + N2 uses total occurrences instead. Volume is then V = N × log2(n).

1211. **A** — LCOM grows when pairs of methods do not use common attributes. High LCOM therefore signals low cohesion, a hint that the class could be split into smaller classes with one responsibility each. Coupling to other classes is measured by CBO, and subclasses by NOC.

1212. **B** — POFOD is the likelihood that a request for service fails. It suits systems that are demanded rarely but must work when they are, such as safety shutdown systems. Continuously running systems are better described by ROCOF (rate of occurrence of failures) or availability.

1213. **B** — Defect density = defects / KLOC = 36 / 12 = 3 defects per thousand lines. Dividing by 12,000 instead of 12 or inverting the ratio gives the wrong figures.

1214. **D** — Effort = 8 people × 6 months = 48 person-months. Productivity = 24,000 / 48 = 500 LOC per person-month. Dividing by months only (4,000) or by people only (3,000) ignores half the effort.

1215. **A** — V(G) = E − N + 2P = 11 − 9 + 2 × 1 = 4, where P = 1 because the graph is one connected component. Forgetting the + 2 gives 2.

1216. **C** — UFP = 5×4 + 4×5 + 3×4 + 2×10 + 1×7 = 20 + 20 + 12 + 20 + 7 = 79. Fifteen is just the number of functions, unweighted.

1217. **A** — VAF = 0.65 + 0.01 × 42 = 1.07. FP = UFP × VAF = 79 × 1.07 = 84.53. Adding 42 to 79 or using 0.65 as the multiplier misreads the formula.

1218. **A** — There are three binary decisions: the if, the elif and the second if. The else adds no decision of its own. V(G) = decisions + 1 = 3 + 1 = 4, so 4 test cases cover a basis set.

1219. **C** — DRE = E / (E + D), where E is defects found before delivery and D after. DRE = 90 / (90 + 10) = 0.90, so 90%. Using 80/90 or 10/90 confuses the denominator.

1220. **A** — Availability = MTTF / (MTTF + MTTR) = 98 / 100 = 0.98, or 98%. MTBF here is MTTF + MTTR = 100 hours.

1221. **B** — Vocabulary n = 10 + 6 = 16 and length N = 30 + 18 = 48. V = N × log2(n) = 48 × 4 = 192 bits. Using log2(48) or multiplying 48 × 16 gives wrong values.

1222. **C** — Effort = 450 / 9 = 50 person-months. Cost = 50 × 200,000 = Rs 10,000,000. Multiplying 450 by 9 inverts the productivity rate.

1223. **C** — For graphs whose decisions are all binary, V(G) = number of predicate nodes + 1 = 4 + 1 = 5. A node with k outgoing edges contributes k − 1 instead.

1224. **B** — UFP = 4×3 + 2×6 + 3×5 + 2×3 + 1×15 + 2×5 = 12 + 12 + 15 + 6 + 15 + 10 = 70. VAF = 0.65 + 0.35 = 1.00, so FP = 70. A VAF of exactly 1 is a neat check that the adjustment is neutral.

1225. **D** — (fan-in × fan-out)² = (3 × 2)² = 36, and 20 × 36 = 720. Forgetting to square gives 120, adding fan-in and fan-out gives 20 × 25 = 500, and squaring the length as well gives 14,400.

1226. **C** — Decisions: the for loop (1), the two simple conditions in the if (2) and the while loop (1), giving 4. V(G) = 4 + 1 = 5. A tool that counts the whole if as one decision would report 4, which is why the stem fixes the rule.

1227. **C** — Team A does reach 600 LOC/PM against Team B's 300, but both delivered the same functionality and Team B used half the effort. LOC productivity penalises high-level languages, which is why Pressman and others recommend function-based measures for cross-language comparison.

1228. **C** — Raw counts favour the bigger module, but normalising by size shows Module B has 12 / 2 = 6 defects per KLOC against 30 / 10 = 3 for A. Module B is the more error-prone code per line, so reviewing it first finds more defects per hour of effort.

1229. **A** — When a measure becomes a target, people optimise the measure (Goodhart's law). LOC rewards long code, copy-paste and avoiding libraries. Sommerville warns against using metrics to judge individuals because it distorts behaviour and damages trust.

1230. **C** — V(G) is the number of independent paths in a basis set, and covering a basis set needs V(G) = 12 test cases (fewer only if some paths are infeasible). Branch coverage is a weaker criterion: each edge executed at least once, often with fewer tests. Basis path coverage implies branch coverage, not the other way round.

1231. **A** — Before: V(G) = 14 + 1 = 15. After: the original keeps 8 decisions, so V(G) = 9; the helper has 6, so V(G) = 7. Each function is now under McCabe's limit of 10, though the total (16) rises by one, because the call itself adds no decision but each function counts its own + 1.

1232. **B** — FP before = 200 × (0.65 + 0.30) = 190. After = 200 × (0.65 + 0.50) = 230. The increase is 40 FP, about 21%. The unadjusted count stays the same, but VAF scales it.

1233. **D** — Overall DRE = E / (E + D) where E is every defect removed before delivery: 120 + 60 = 180. DRE = 180 / 200 = 0.90. The manager's ratio is closer to a phase-level DRE for reviews, and even then the defects that escaped reviews should include the 20 found later.

1234. **A** — Cost per LOC: X = 6,000,000 / 12,000 = 500 and Y = 9,000,000 / 30,000 = 300. Cost per FP: X = 30,000 and Y = 45,000. Both delivered the same functionality, so cost per FP is the fair comparison, and X wins. Cost per LOC flatters the more verbose language.

1235. **D** — Decisions: the outer for (1), the inner for (1), the if (1) and the elif (1), giving 4. V(G) = 4 + 1 = 5. Return and break end paths early, but they do not create new branches in the control flow graph.

1236. **D** — Availability = MTTF / (MTTF + MTTR). A: 500 / 505 = 99.01%. B: 200 / 201 = 99.50%. B fails more often but recovers five times faster, so it is down less in total. If the requirement were fewest failures (reliability), A would win; the stem asks for availability.

1237. **B** — Pressman builds a ladder: a measure is a raw number, a metric relates measures (such as errors per review hour), and an indicator is a metric or set of metrics that gives a manager or engineer insight they can act on.

1238. **B** — The categories are just names with no order: a logic defect is not 'more' than an interface defect. That is a nominal scale, where you can only count and compare for equality. Severity levels, by contrast, are ordinal.

1239. **D** — Calendar dates have meaningful differences (10 days apart) but no true zero, so you cannot say one date is 'twice' another. That is an interval scale. Counts, effort and LOC all have a true zero, so they are ratio measures.

1240. **A** — 'Twice as large' needs a true zero, which only a ratio scale has. Category labels are nominal, severity ranking is ordinal, and a difference between dates is interval: none of those support ratios.

1241. **C** — Pressman lists LOC, execution speed, memory size and defects reported as direct measures, because you count them straight off the product. Maintainability, reliability and quality cannot be counted directly; they are inferred from other measures.

1242. **D** — Process metrics are collected across projects over long periods to improve the way the organisation works, and DRE is Pressman's standard example. Complexity, class counts and LOC all describe one product.

1243. **C** — WMC adds up a complexity weight (often cyclomatic complexity) for each method in the class. A high WMC means more effort to build, test and maintain the class. Direct subclasses are NOC, coupling is CBO and depth is DIT.

1244. **C** — NOC counts only the direct subclasses. A class with many children has a lot of reuse riding on it, so changes to it need careful testing. Counting all descendants is a common mistake.

1245. **A** — CBO counts the classes a class is coupled to. High coupling makes a class sensitive to changes in others, harder to reuse and harder to test in isolation. Subclasses are NOC, cohesion is LCOM and depth is DIT.

1246. **A** — RFC is the size of the response set: every method that can run when an object of the class receives a message, which is its own methods plus the methods they invoke directly. A larger RFC means more testing and debugging effort.

1247. **A** — Halstead length counts every occurrence: N = N1 + N2. The distinct counts give the vocabulary n = n1 + n2, and multiplying N by log2 n gives the volume V, not the length.

1248. **D** — Fan-in counts the modules that call (flow into) a module; fan-out counts the modules it calls. High fan-in usually signals useful reuse. Both feed the Henry-Kafura information flow metric.

1249. **C** — Pressman describes feature points as function points plus an algorithms parameter. Ordinary FP undercount systems whose complexity lies in computation rather than inputs, outputs and files, such as embedded or scientific code.

1250. **D** — Object points weight each screen, report and 3GL module by complexity. Despite the name they have nothing to do with object-oriented classes. Inputs, outputs, inquiries and files are the function point types.

1251. **D** — ROCOF is a frequency of failures over time, suited to systems that run continuously and handle many requests. The probability per demand is POFOD, the percentage of uptime is availability, and average repair time is MTTR.

1252. **A** — The time between two failures is the time the system runs (MTTF) plus the time it is down for repair (MTTR), so MTBF = MTTF + MTTR. The last option is the formula for availability.

1253. **C** — A static metric is taken from a representation of the system, such as the code or design, without executing it. Dynamic metrics, such as response time or failures observed, need the program to run.

1254. **B** — GQM goes Goal, then Questions that tell you whether the goal is being met, then Metrics that answer each question. The failure-rate question links the goal to the failures-per-1,000 metric. A tool purchase is an action, not part of GQM.

1255. **D** — The report contains derived data (totals and percentages), so it is an external output. An external inquiry retrieves data with no calculation. EI brings data into the boundary, and EIF is a data file, not a process.

1256. **A** — Function-oriented metrics normalise by function points; size-oriented metrics normalise by LOC or KLOC. Pressman uses errors per KLOC, cost per LOC and pages per KLOC as his size-oriented examples.

1257. **C** — Sommerville's predictor metrics are product measures used to predict qualities such as maintainability, for example cyclomatic complexity or identifier length. Control metrics support process management: effort, elapsed time and defects repaired.

1258. **B** — MTTC is the time to analyse a change request, design the change, implement, test and distribute it. Short MTTC means the program is easy to change, which is maintainability.

1259. **A** — V(G) = E − N + 2 for one connected component, which equals predicate nodes + 1 and the number of regions. Swapping edges and nodes gives a different number: for 11 edges and 9 nodes it gives 0 instead of 4.

1260. **D** — Data functions (ILF and EIF) are rated with DETs and RETs. Transactions (EI, EO, EQ) are rated with DETs and FTRs, which is the common mix-up. Physical size never enters the count.

1261. **C** — Defect density = defects ÷ size = 18 ÷ 120 = 0.15 defects per FP. Dividing the other way (120 ÷ 18 = 6.67) gives FP per defect, which is the inverse.

1262. **B** — Effort = 4 × 5 = 20 person-months, so productivity = 600 ÷ 20 = 30 FP per person-month. Dividing by people only (150), months only (120) or people plus months (66.7) ignores how effort is measured.

1263. **C** — Cost per LOC = 5,400,000 ÷ 18,000 = Rs 300. The other values come from slipping a decimal place or dividing 18,000 into 6 million.

1264. **B** — With P = 2, V(G) = 20 − 17 + 2 × 2 = 7. Treating it as one component gives 5, using + P gives 4, and leaving out the term gives 3.

1265. **B** — Decisions: the if, the two elifs and the while loop, so 4 decisions and V(G) = 4 + 1 = 5. Forgetting the + 1 gives 4; counting the else as a decision gives 6.

1266. **C** — The switch has 3 case labels, so 3 decisions; the if adds 1, giving 4 decisions and V(G) = 5. Check with the graph: 8 nodes and 11 edges, so 11 − 8 + 2 = 5. Counting the default as an extra decision gives 6.

1267. **A** — Length counts every occurrence: N = N1 + N2 = 40 + 25 = 65. The distinct counts give the vocabulary (10 + 12 = 22), which is a different measure.

1268. **A** — D = (12 ÷ 2) × (40 ÷ 8) = 6 × 5 = 30. Using only one factor gives 6 or 5, and forgetting to halve n1 gives 60.

1269. **B** — n = 16 and N = 64, so V = 64 × log2 16 = 64 × 4 = 256. D = (8 ÷ 2) × (24 ÷ 8) = 12. E = 12 × 256 = 3,072. 256 is only the volume, 12 only the difficulty, and 6,144 comes from not halving n1.

1270. **B** — VAF = 0.65 + 0.01 × 0 = 0.65, so FP = 120 × 0.65 = 78. A zero total does not leave the count unchanged (that needs ΣFi = 35) and does not wipe it out. 162 uses the maximum VAF of 1.35.

1271. **D** — 3 × 7 + 2 × 10 + 4 × 3 = 21 + 20 + 12 = 53. Using average weights for everything gives 51, rating the EQs average gives 57, and rating the ILFs complex gives 63.
1272. **C** — VAF runs from 0.65 (all 14 characteristics rated 0) to 1.35 (all rated 5), so FP runs from 100 × 0.65 = 65 to 100 × 1.35 = 135.
1273. **C** — Availability = MTTF ÷ (MTTF + MTTR). Solve 495 ÷ (495 + R) = 0.99, so 495 + R = 500 and R = 5 hours. 4.95 hours is 1% of MTTF, which uses the wrong denominator.
1274. **B** — ROCOF = failures ÷ operating time = 6 ÷ 1,200 = 0.005 per hour, which is 5 per 1,000 hours. 200 comes from dividing hours by failures.
1275. **D** — POFOD = failed demands ÷ total demands = 3 ÷ 1,500 = 0.002, meaning about 2 failures in every 1,000 demands. 500 is demands per failure, the inverse.
1276. **D** — Phase DRE = errors found in the phase ÷ (those + errors from that phase found later) = 40 ÷ (40 + 10) = 80%. 20% is the share that escaped, and 25% is 10 ÷ 40.
1277. **B** — SMI = (500 − (10 + 25 + 5)) ÷ 500 = 460 ÷ 500 = 0.92. As SMI approaches 1 the product is stabilising. 0.93 forgets the deleted modules and 0.95 counts only changed ones.
1278. **A** — $(2 \times 4)^2 = 64$, and 50 × 64 = 3,200. Forgetting to square gives 400; adding fan-in and fan-out before squaring gives 1,800; leaving out length gives 64.
1279. **B** — VAF = 250 ÷ 200 = 1.25. From VAF = 0.65 + 0.01 × ΣFi, ΣFi = (1.25 − 0.65) ÷ 0.01 = 60. 50 is the FP difference, a different quantity.
1280. **C** — Effort = 36,000 ÷ 600 = 60 person-months. With 6 people, duration ≈ 60 ÷ 6 = 10 months, ignoring communication overhead. 60 is the effort, not the duration.
1281. **A** — Pairs: M1–M2 share b (Q = 1); M1–M3 and M2–M3 share nothing (P = 2). LCOM = 2 − 1 = 1. M3 shares no data with the others, hinting the class may hold an unrelated responsibility.
1282. **A** — Testability follows control-flow complexity, not raw length. A needs 25 independent paths and breaks McCabe's threshold of 10; B is long but has only 8 paths. Splitting A gives the biggest gain.
1283. **B** — More defects found in-house plus fewer escaping to customers means a higher DRE. Defect density measured before release reflects how hard you look as well as how many defects exist, so it must be read with post-release data.
1284. **C** — GQM is top-down: define a goal, ask questions that show progress towards it, then pick metrics to answer those questions. Collecting whatever a tool offers produces data with no decision attached to it.
1285. **D** — Ordinal data supports order (Critical > High) but not arithmetic, because the step from High to Critical need not equal Low to Medium. Use the median, mode or counts per level, for example 'Critical defects fell from 6 to 2'.
1286. **A** — NOC counts direct subclasses. Chidamber and Kemerer note that more children means more reuse but also more potential for misused abstraction and more testing, because each child depends on the parent's behaviour.
1287. **B** — Once a measure becomes a target, people manage the measure. Linking rewards to defect counts encourages under-reporting, which hides problems. Pressman advises never using metrics to judge or reward individuals.
1288. **D** — Per KLOC favours the verbose language, because COBOL needs far more lines for the same features. Normalised by delivered function, J has half the defects per FP. Use FP when comparing across languages.
1289. **D** — Sommerville recommends availability for continuously running systems where users care about downtime and repair time. ROCOF treats a one-second and a one-hour outage the same, and POFOD suits rarely triggered protection systems.
1290. **A** — The if/elif chain has 7 decisions, so V(G) = 8. The table version has no decisions, so V(G) = 1. Cyclomatic complexity measures control flow only; the 7 data entries still need validating, but not as separate paths.
1291. **C** — IFPUG counts elementary processes: the smallest unit of activity meaningful to the user. Submitting the form is one EI. Its fields are DETs, which, with files referenced, decide whether it is simple, average or complex. Counting fields inflates FP heavily.
1292. **D** — Counting copied lines as new work makes LOC productivity look 2.5 times better than it was. Report 20,000 new LOC ÷ 10 PM = 2,000 LOC/PM, and record reuse separately, since reuse is valuable but different work.

1293. **A** — X: 400 ÷ 404 ≈ 99.01%. Y: 200 ÷ 202 ≈ 99.01%. Doubling MTTF or halving MTTR changes the ratio MTTF:MTTR by the same factor, so availability is identical. The choice then rests on cost and on whether failures themselves do harm.

1294. **C** — Availability = MTTF ÷ (MTTF + MTTR) = MTTF ÷ MTBF. Since MTBF = MTTF + MTTR, MTTF = 96 h and availability = 96 ÷ 100 = 96%. The student treated MTBF as MTTF and counted the repair time twice.

1295. **B** — A valid metric must reflect the attribute it claims to measure. Commit size varies hugely, so commit count tracks activity rather than output, and once it is rewarded people will split work to raise it.

1296. **D** — Physical-line counts include comments and blanks, so the same code gives a bigger KLOC and a smaller defect density. Without a common counting rule the denominators differ, and the comparison means nothing.

1297. **B** — Pressman defines an SCI as information created during the software process that is put under configuration control: documents, models, source code, test cases, even tools. Each item is identified, versioned and changed only through the change process.

1298. **C** — IEEE 610.12 describes a baseline as a specification or product that has been formally reviewed and agreed on, serves as the basis for further development, and can be changed only through formal change control. Baselines are milestones, such as the approved SRS or a release build.

1299. **B** — The CCB (also called the change control authority) reviews each change request with its impact analysis, cost and risk, then approves, rejects or defers it. Developers implement approved changes; the CCB makes the decision.

1300. **A** — git add records a snapshot of the file in the index, the staging area. Nothing enters history until git commit, and nothing reaches a remote until git push.

1301. **D** — git clone copies every commit, branch and tag of a remote repository and sets it up as origin. git init creates a new empty repository, and git fetch only updates an existing clone.

1302. **C** — In Git each developer has a complete repository, so commits, branches and log queries are local and fast; the shared server is a convention. In SVN, commits and history queries go to the one central server.

1303. **D** — Semantic versioning raises MAJOR for incompatible API changes, MINOR for backward-compatible new features, and PATCH for backward-compatible bug fixes. Raising MAJOR resets MINOR and PATCH to 0.

1304. **C** — git pull fetches from the remote and then integrates the upstream branch, by merging unless pull.rebase is configured or --rebase is given. git fetch alone downloads commits but leaves your branch where it is.

1305. **C** — .gitignore stops untracked files matching its patterns from showing as untracked and from being added by git add .. It has no effect on files that are already tracked; those must be removed from the index with git rm --cached.

1306. **A** — The classic SCM activities are identification, change control, status accounting and auditing (Pressman adds version control). Unit testing is a verification activity; SCM controls the items it produces but does not perform it.

1307. **D** — A functional configuration audit (FCA) checks that the product performs as its requirements say. A physical configuration audit (PCA) checks that the as-built product, with its components and versions, matches its documentation.

1308. **A** — Sommerville distinguishes a version (an instance of a configuration item), a variant (versions that coexist and differ for different settings, such as Windows and Linux builds) and a release (a version distributed to customers).

1309. **B** — Git lets everyone edit their own copy and resolves overlaps at merge time, asking a person only when changes conflict. Lock-modify-unlock, used by older tools and for binary assets, stops two people editing a file at once.

1310. **B** — git tag -a creates a tag object with its own metadata (and optional signature); git cat-file -t on it reports tag. A lightweight tag is a ref pointing straight at a commit, so cat-file reports commit. Releases usually use annotated tags.

1311. **B** — Revert keeps history append-only: teammates who already pulled the bad commit simply pull the reverting commit too. Resetting and force-pushing rewrites history that others have built on, which causes lost work and confusing merges.

1312. **C** — Trunk-based development keeps branches short-lived and merges often, so integration problems appear early and continuous integration stays meaningful. Long-lived feature branches drift from main and cause painful merges.

1313. **A** — A backward-compatible new feature raises MINOR and resets PATCH: 2.4.1 becomes 2.5.0. PATCH alone is for bug fixes, and MAJOR is for breaking changes.
1314. **C** — Breaking the public API requires a MAJOR increase, and MINOR and PATCH reset to 0, so 2.4.1 becomes 3.0.0. Keeping 4.1 after the major bump is not allowed.
1315. **B** — git diff shows working tree versus index. After git add the two match, so it prints nothing. Use git diff --staged (or --cached) to compare the index with HEAD and see the staged change.
1316. **D** — main has no commits that feature lacks, so Git just moves the main pointer forward to B. No merge commit is created unless you pass --no-ff. Conflicts need changes on both sides.
1317. **B** — The branches have diverged, so Git performs a three-way merge using A as the common ancestor and creates a merge commit M with parents C and B. History: A, B, C and M, four commits.
1318. **D** — --soft moves only the branch pointer. The index and working tree keep the changes, so git status shows f as staged and you can recommit straight away. --mixed (the default) would also reset the index, and --hard would also reset the working tree.
1319. **C** — --hard resets the branch, the index and the working tree to HEAD~1, so the change disappears from all three. It can often still be recovered through git reflog for a while, but not from the working tree.
1320. **D** — Revert creates a new commit that applies the inverse of B. History becomes A, B, C, and f's content is as it was after A. Nothing is deleted, which is why revert is safe on shared branches.
1321. **A** — -a stages modifications and deletions of tracked files before committing, but it never adds untracked files. new.txt still shows as ?? in git status until you git add it.
1322. **A** — --cached acts only on the index. The file stays in your working tree; git status shows a staged deletion plus the file as untracked. Past commits still contain it, so secrets need history rewriting and rotation.
1323. **A** — SemVer compares MAJOR, MINOR and PATCH numerically, left to right. MAJOR is equal (1), and MINOR 10 > 9, so 1.10.0 is newer. Sorting version strings as text gets this wrong, a common bug in scripts.
1324. **A** — --squash applies the combined changes to the index and working tree but does not create a commit or record a merge. Git prints "Squash commit -- not updating HEAD". You then run git commit to create a single commit.
1325. **D** — The section between <<<<<<< HEAD and ======= is the version in the current branch (HEAD, here main). The section between ======= and >>>>>>> ft is the incoming branch's version. You edit the file, remove the markers, git add it and commit.
1326. **C** — The remote main has commits your local main lacks, so a push would not be a fast-forward. Integrating them first keeps everyone's work. Force-pushing would discard a teammate's commits on the shared branch.
1327. **D** — .gitignore only applies to untracked files. Because secret.env is already tracked, Git keeps reporting changes. Removing it from the index stops tracking, but the old commits still contain the secret, so it must be treated as leaked and rotated (and history cleaned if needed).
1328. **B** — Teammates now have a commit that no longer exists on the remote. Their next pull can reintroduce it or cause confusing merges. git revert keeps history linear and shared, which is the standard practice for public branches.
1329. **A** — Once an item is baselined, changes go through formal change control so that cost, schedule and side effects are assessed and the baseline and traceability are updated. An informal change leaves the SRS, design, tests and plan out of step with the code.
1330. **B** — Rebase replays B on top of C, creating a new commit B' with a new parent and therefore a new hash. History is linear with no merge commit. If teammates had B, their history now disagrees with yours, which is why you avoid rebasing public branches.
1331. **B** — Cherry-pick re-applies one commit's diff on top of the current branch, creating a new commit object with a different parent, so a different hash. The original H stays on feature. Later merging feature may show the change twice in history, though Git usually merges it cleanly.
1332. **D** — Checking out a commit hash gives a detached HEAD: new commits belong to no branch. When you switch away, nothing points at them except the reflog, and they are pruned eventually. Git warns you and suggests creating a branch to keep them.
1333. **D** — By default git stash saves changes to tracked files (staged and unstaged) and leaves untracked files alone. They follow you to the new branch because Git never touches untracked files on a switch. git stash -u would have stashed u.txt too.

1334. **C** — Each bisect step halves the range of suspects: 1,024 → 512 → … → 1. That takes log2(1,024) = 10 steps. Testing every commit would take up to 1,024 builds, and testing half of them still leaves the answer unknown.
1335. **C** — The software behaved correctly, so a functional audit passed. The mismatch is between what was built and what was documented, which a physical configuration audit checks. Status accounting records would also have shown the wrong version.
1336. **A** — Copy-modify-merge works because text diffs can be merged. Binary files cannot be merged line by line, so the safer model for them is locking: an artist locks the file, edits and unlocks it. Git LFS offers file locking for this case.
1337. **C** — Status accounting keeps the records that answer what changed, who changed it, when, and what is affected. Auditing checks those records and the product against reality; identification names the items in the first place.
1338. **C** — Sommerville separates a version (any distinct instance), a variant (versions that exist side by side, such as per-platform builds) and a release (a version handed to customers, with its documentation and data).
1339. **C** — git init sets up the repository structure in .git. Nothing is tracked or committed yet; you still need git add and git commit. Copying an existing repository is git clone.
1340. **B** — git branch only creates the new branch pointer at the current commit. You stay on main until you run git switch feature (or git checkout feature). git switch -c feature creates and switches in one step.
1341. **D** — Impact analysis tells the change control board what a change will really cost and what else it touches, so it can approve, reject or defer it on evidence. Testing, recording and merging come after a decision.
1342. **B** — Identification decides which work products become configuration items and how they are named, numbered and related. Without it, nothing else in SCM can refer to items reliably.
1343. **D** — clone records the source as a remote called origin, which is why you push with git push origin main and see remote-tracking branches such as origin/main. You can rename it or add more remotes.
1344. **D** — Derived items are reproduced from controlled sources plus a controlled build, so the sources and build scripts are what you version. Git can store binaries, and release builds may be archived as configuration items elsewhere.
1345. **D** — Git is distributed: a commit is written to your own repository, which is why you can commit offline. Sharing it needs git push. The staging area is what you prepare before committing.
1346. **B** — SemVer allows a hyphen and pre-release label, and a pre-release version has lower precedence than the associated normal version. So 2.0.0-beta comes before the final 2.0.0.
1347. **A** — SemVer says major version zero is for initial development. The stability promises of MINOR and PATCH only start at 1.0.0, when the public API is declared.
1348. **C** — A blob holds file content, a tree holds a directory listing, a commit points to a tree and its parents, and an annotated tag points to an object. Branches are references (pointers), not objects.
1349. **C** — A commit's ID is a hash of its content: tree, parents, author, committer and message. Changing the message makes a new commit object with a new hash, though the tree hash stays the same.
1350. **B** — Hotfixes repair production, so they branch from master, get tagged on release, and are merged into develop too so the fix is not lost in the next release.
1351. **A** — The reflog logs every move of HEAD and branch tips in your own repository, so a commit left unreferenced after reset --hard can still be found by its hash or by HEAD@{n} until it expires.
1352. **D** — A codeline is a sequence of versions of a component or system. A mainline is a sequence of system versions developed from an original baseline, and branches create new codelines.
1353. **A** — --no-ff forces a merge commit even when the target has no new commits. Teams use it to keep a visible record that a set of commits arrived together from a feature branch.
1354. **C** — --ff-only allows the merge only if main can simply move forward to feature's tip. With diverged branches it stops with 'Not possible to fast-forward, aborting' and changes nothing.
1355. **D** — Each commit points to a tree that names every file's blob. Unchanged files reuse the same blob, so snapshots are cheap. Git may delta-compress objects in packfiles, but that is a storage detail, not its model of history.
1356. **D** — Comparing each tip with the merge base shows which side changed which lines. A line changed on only one side is taken from that side; a line changed differently on both sides is a conflict.

1357. **D** — Plain reset is --mixed: it moves the branch and resets the index, but leaves the working tree alone. git status shows f as modified and unstaged. --soft would keep the changes staged and --hard would discard them.

1358. **A** — Without --source or --staged, git restore copies the file from the index into the working tree. The index holds v2, so f becomes v2 and the staged change remains. To go back to v1 you would use git restore --source=HEAD --staged --worktree f.

1359. **C** — git diff --staged compares the index with the last commit, and the index holds 'two' but not 'three'. Plain git diff, which compares the working tree with the index, would show only 'three'.

1360. **D** — git restore --staged unstages just the named file, leaving the working copy alone. The commit takes only a.txt, and git status then lists b.txt as untracked.

1361. **C** — -d is the safe delete: it checks that the branch's commits are reachable from the current branch or its upstream. -D deletes anyway, leaving the commit reachable only through the reflog.

1362. **B** — Stashes form a stack, and pop applies and removes the newest one (stash@{0}), which holds change 2. The stash holding change 1 is left in the list.

1363. **D** — A fast-forward was possible, but --no-ff forces a merge commit M with parents A and B. History is A, B and M: three commits. A plain merge would have fast-forwarded to two.

1364. **A** — Pushing a branch does not push tags by default. Push the tag by name, or use --tags (or --follow-tags for annotated tags). Checked with git ls-remote on a test remote.

1365. **A** — *.log ignores every .log file, and the later !keep.log negates that rule for keep.log, so it shows up again. a.log stays ignored. Order matters: a later matching line wins.

1366. **B** — A leading slash anchors the pattern to the directory that holds the .gitignore, so /temp matches only the top-level temp. The nested src/temp is not matched; plain temp without the slash would ignore both.

1367. **B** — Rebase replays B on top of C as a new commit B'. feature is now directly ahead of main, so the merge only moves main forward: three commits, no merge commit.

1368. **C** — A feature bumps MINOR and resets PATCH (1.5.0). A fix bumps PATCH (1.5.1). A breaking change bumps MAJOR and resets the others (2.0.0). Forgetting to reset PATCH is a common slip.

1369. **B** — Pre-releases come before the normal version. Identifiers are compared part by part: 'beta' < 'rc' in ASCII order, and numeric parts are compared as numbers, so 2 < 11. A plain text sort wrongly puts beta.11 before beta.2.

1370. **B** — Git protects uncommitted work: when switching would overwrite a modified file, it aborts. Commit, stash, or use git switch --merge if you deliberately want a merge into the new branch.

1371. **A** — Staging the file tells Git the conflict is resolved; then git commit (or git merge --continue) completes the merge. git merge --abort throws the merge away, and git resolve is not a Git command.

1372. **A** — main..ft means commits reachable from ft but not from main. A is reachable from both, so only B and C remain (log lists newest first). ft..main would list only D.

1373. **A** — ~n follows the first parent n times: HEAD~1 is C and HEAD~2 is B. HEAD^ is the same as HEAD~1; HEAD^2 would mean the second parent of a merge.

1374. **A** — git clean -f removes untracked files in the current tree but skips untracked directories unless you add -d, and skips ignored files unless you add -x. So only u.txt goes. -n is a dry run, not a requirement.

1375. **C** — git rm deletes the file and stages the deletion in one step; the next commit records it. Keeping the file on disk needs --cached. Past commits are never changed.

1376. **D** — git mv moves the file and stages both sides, and git status shows the pair as a rename. Git detects renames from content similarity rather than storing them, but status still reports it as R.

1377. **D** — fetch updates origin/main only, so status compares your branch with it and reports 'behind by 2 commits, and can be fast-forwarded'. Your files change only when you merge or pull.

1378. **D** — Reverting a merge undoes its changes but keeps the commits in history. A second merge sees nothing new to bring in. The fix is to revert the revert commit, then merge the new fixes.

1379. **A** — The pain comes from integrating late. Frequent small merges keep each difference small, and automated checks catch breakage the same day. Locking or squashing does not reduce how far the branches drift.

1380. **B** — A hotfix branch from the release tag contains only what is in production plus the fix, so 2.3.1 ships nothing unfinished. Bringing the fix into main stops it from coming back in 2.4.

1381. **C** — --amend makes a new commit with a new hash in place of A. origin/main still has A, so your branch is not a descendant and the push is rejected. Force-pushing would rewrite history teammates may already have.
1382. **D** — A build is reproducible only if every input is identified and versioned, including third-party libraries. Floating versions mean two builds of the same commit can pull different code.
1383. **C** — Pressman notes that before a baseline is set, changes can be made quickly and informally. Once the item is formally reviewed and baselined, changes must go through the formal process.
1384. **C** — Git merges by line regions, not whole files. Conflicts arise only where both sides changed the same or adjacent lines. Checked on a test repository: the merge completed with both edits kept.
1385. **A** — The textual changes never overlap, so the merge is clean, but the result is broken. Checked on a test repository: the merge succeeded and running the new file failed with an import error. This is why CI builds and tests the merged result.
1386. **C** — Every earlier commit still contains the key, and anyone may have cloned it already. The only safe step is to treat the key as compromised and replace it; history rewriting with tools such as git filter-repo is secondary.
1387. **B** — SemVer says deprecating public API is a MINOR bump, because nothing breaks yet. The function is removed later in a MAJOR release. Adding the replacement is also a backward-compatible feature.
1388. **D** — Impact analysis traces a change to everything that depends on the changed item. Here it stopped at the requesting screen, so the board approved without knowing the real scope.
1389. **A** — Rebasing unpushed local commits is safe, since no one else has them, and gives a linear history. Force-pushing would erase teammates' commits, and reset --hard would throw away your own.
1390. **C** — The audit that passed was physical: it checked the as-built items against documentation. Checking that the product does what the requirements say is the functional configuration audit.
1391. **B** — reset moved the branch but B still exists as an unreferenced object. The reflog records the move, so HEAD@{1} points to B. Checked on a test repository: B was restored.
1392. **A** — A clone copies all history, and the old snapshots still reference the video blobs. Only rewriting those commits removes them, which needs coordination since hashes change. LFS stores pointers instead of the media itself.
1393. **B** — Pressman says a risk always involves uncertainty, because it may not happen, and loss, because unwanted consequences follow if it does. A certain event is a constraint or a problem, not a risk.
1394. **B** — RE = P × C, where P is the probability that the risk occurs and C is the cost or loss if it does. It gives an expected loss, which lets you rank risks and size contingency budgets.
1395. **C** — Proactive management starts early: identify risks, analyse probability and impact, rank them and plan mitigation and contingencies. Reactive management, the "fire-fighting" approach Pressman criticises, waits until a risk becomes a problem.
1396. **D** — Product risks affect the quality or performance of the software being built. Project risks affect the schedule or resources (such as staff leaving), and business risks affect the organisation (such as a competitor launching first). Pressman calls product risks technical risks.
1397. **B** — Boehm's spiral model makes each loop pass through objective setting, risk assessment and reduction, development and planning. The risks found decide what the next loop does, such as building a prototype.
1398. **A** — The risk register (Pressman's risk table plays a similar role) is the living record of risk management. It is updated as risks are added, re-assessed, closed or turn into problems.
1399. **B** — Transferring a risk shifts its consequences to a third party, through insurance, contracts or outsourcing. The risk still exists, but someone else bears the loss.
1400. **C** — Avoidance changes the plan so the risk cannot occur, here by removing the dependency. Mitigation would keep the feature but reduce probability or impact, for example by building a hardware simulator.
1401. **D** — Sommerville lists avoidance, minimisation and contingency strategies. A contingency plan accepts that a risk may happen and prepares the response in advance, such as a backup supplier if the main one fails.
1402. **D** — The RMMM plan records, for each risk, how to mitigate it (lower its probability or impact), how to monitor it (watch indicators) and how to manage it if it happens (contingency). It may be a separate document or part of a risk information sheet.
1403. **C** — Pressman's impact categories are 1 = catastrophic, 2 = critical, 3 = marginal and 4 = negligible. Many students assume 4 is the worst because higher numbers usually mean more, so this is a common trap.

1404. **B** — Predictable risks are extrapolated from past project experience, such as turnover or poor communication with the customer. Known risks are found by careful evaluation of the plan and environment, such as an unrealistic delivery date. Unpredictable risks can occur but are very hard to foresee.

1405. **B** — Triggers turn monitoring into action. For example, two failed builds in a row or rising overtime can signal schedule risk. When a trigger fires, the team re-assesses the risk or starts the contingency plan.

1406. **A** — Qualitative analysis is fast and uses scales or a probability-impact matrix to prioritise. Quantitative analysis puts numbers on probability and impact, such as expected monetary value or simulation, usually for the highest-ranked risks.

1407. **D** — Boehm's 1991 list starts with personnel shortfalls, followed by unrealistic schedules and budgets and developing the wrong functions. The others are on the list but lower down.

1408. **A** — A secondary risk is created by the response itself, such as vendor risk after outsourcing a module. Residual risk is what remains of the original risk after the response. Sommerville and Pressman do not use these terms, but PMBOK-based questions do.

1409. **A** — RE = P × C = 0.3 × 2,000,000 = Rs 600,000. That is the expected loss you might hold in reserve. 1,400,000 is 70% of the cost, which uses the wrong probability.

1410. **C** — Score = 4 × 3 = 12 out of a maximum of 25. Adding the ratings (7) is a different and weaker scoring method; this matrix multiplies them.

1411. **B** — RE = 0.25 × 40 = 10 person-days. Exposure can be expressed in any unit of loss: money, days or effort.

1412. **A** — Components to build = 60 × 0.3 = 18. Cost = 18 × 100 × 14 = Rs 25,200. RE = 0.7 × 25,200 = Rs 17,640. Rs 25,200 is the cost alone, without the probability.

1413. **C** — REs: 0.2 × 500,000 = 100,000; 0.5 × 200,000 = 100,000; 0.1 × 1,500,000 = 150,000. Total = Rs 350,000. Summing raw costs (2,200,000) assumes every risk will happen.

1414. **B** — REs: R1 = 60,000, R2 = 90,000, R3 = 75,000, R4 = 40,000. Highest first gives R2, R3, R1, R4. Ranking by probability alone would put R4 first, which is the reverse.

1415. **A** — RRL = (RE before − RE after) / cost of reduction = (400,000 − 100,000) / 60,000 = 5. A value above 1 means the reduction is worth more than it costs.

1416. **B** — RRL(M1) = (300,000 − 150,000) / 50,000 = 3. RRL(M2) = (300,000 − 90,000) / 150,000 = 1.4. M1 buys more reduction per rupee, and it also leaves the lower total expected cost (200,000 against 240,000).

1417. **C** — Staff turnover affects resources and schedule, so Sommerville classes it as a project risk. If it were about the module failing in use, it would be a product risk.

1418. **B** — Business risks affect the organisation that develops or buys the software, such as market competition or a change of management. The software and schedule may be fine, but the product could still fail commercially.

1419. **A** — The risk still exists, because she may still leave, but cross-training and documentation lower the damage if she does. Lowering probability or impact is mitigation, which Sommerville calls minimisation.

1420. **D** — Expected cost (EMV) = Σ probability × outcome. Build = 0.6 × 1.0 + 0.4 × 1.8 = 1.32 million. Buy = 0.9 × 1.2 + 0.1 × 2.0 = 1.28 million. Buy is lower, even though build's best case is cheaper.

1421. **D** — Without mitigation, RE = 0.5 × 800,000 = 400,000. With mitigation, the cost is 150,000 + 0.1 × 800,000 = 230,000. Mitigating saves 170,000 in expected terms.

1422. **C** — Expected delay = 0.3 × 1 + 0.2 × 3 = 0.9 months, so RE = 0.9 × 400,000 = Rs 360,000. Adding the delays without weights (4 months) gives 1,600,000, and averaging the three outcomes equally gives about 533,333.

1423. **D** — Third-party dependency is a standard checklist item. A proactive team would have estimated it, chosen a mitigation (such as portable deployment scripts) and set a trigger (price notices). Reacting only after impact is what Pressman calls fire-fighting.

1424. **D** — RE gives an average, but organisations are not risk-neutral about catastrophes. A single occurrence of A could sink the project, and cheap safeguards such as tested backups cut its impact sharply. B can simply be accepted and absorbed by the schedule. Pressman does advise against spending much management time on very unlikely risks, but low cost, high value safeguards for an unrecoverable loss are standard practice.

1425. **A** — Risks were identified and analysed at the start, so those steps happened. Sommerville's process ends with monitoring: check each risk regularly, update probability and impact, and act when indicators change. Without it the register goes stale.

1426. **A** — RE falls from 100,000 to 50,000, a 50,000 reduction that costs 80,000. RRL = 0.625. On expected cost alone it is not justified; it may still be chosen for non-monetary reasons such as safety or reputation, which the stem excludes.

1427. **A** — Scores: A = 10, B = 10, C = 12, D = 8. C is first. A and B tie, and many organisations break ties in favour of higher impact, which would put B above A. Neither likelihood nor impact alone decides the ranking.

1428. **C** — RE(X) = 0.05 × 10,000,000 = 500,000. RE(Y) = 0.5 × 800,000 = 400,000. X ranks higher, and its large single impact adds to the case. Ranking on probability alone is a classic error.

1429. **D** — Both main risks (unclear requirements and unfamiliar technology) shrink through early, cheap experiments. The spiral model puts risk analysis and prototyping at the start of every loop. Freezing vague requirements just moves the risk to the end.

1430. **C** — Monitoring watches factors that signal the risk is rising. For turnover, Pressman suggests watching attitude and pressure on the team, interpersonal relations and the availability of jobs elsewhere. Code output and bug counts can change for many unrelated reasons.

1431. **D** — Every response can create new risks. Outsourcing transfers part of the work, but it brings vendor performance, communication and integration risks. PMI calls these secondary risks; they need their own analysis and response.

1432. **C** — P(at least one) = 1 − P(neither) = 1 − 0.7 × 0.7 = 1 − 0.49 = 0.51. Adding gives 0.6, which double-counts the case where both occur. 0.09 is the chance that both occur.

1433. **B** — A risk may or may not happen, so it is managed with probability and planned responses. An issue has already occurred and needs action now. When a monitored risk materialises, it becomes an issue and the contingency plan starts.

1434. **D** — Residual risk is what is left of the original risk once the response is in place, because few responses remove a risk completely. A new risk caused by the response itself is a secondary risk, which is a different term.

1435. **A** — Sommerville's process first identifies risks, then analyses their probability and effects, then plans how to avoid or minimise them, and finally monitors them throughout the project. Monitoring feeds back into the earlier stages as risks change.

1436. **D** — Identification produces the list of risks, using techniques such as checklists, brainstorming, interviews and lessons from past projects. Rating probability, calculating exposure and ranking all happen afterwards, in analysis.

1437. **D** — Each significant risk is assigned to a named owner who watches its triggers and acts on the planned response. Without an owner, risks tend to be listed but never managed, because everyone assumes someone else is handling them.

1438. **C** — Accepting a risk means acknowledging it and choosing not to act in advance, usually because the cost of a response exceeds the likely loss. Avoidance would remove the risk, transfer would shift it to another party, and mitigation would reduce it.

1439. **B** — Sommerville's risk types are technology, people, organisational, requirements, tools and estimation. Restructuring, financial problems and management changes come from the organisation's environment, so they are organisational risks.

1440. **A** — Pressman says technical risks threaten the quality and timeliness of the software, covering design, implementation, interface, verification and maintenance problems. Project risks threaten the plan, and business risks threaten the product's viability.

1441. **C** — The spiral's second quadrant evaluates alternatives and resolves the main risks, using prototypes, simulations or benchmarks. Development and verification follow, then planning for the next loop. This is what makes the spiral risk-driven.

1442. **C** — PMI defines risk as an uncertain event with a positive or negative effect on objectives. Negative risks are threats and positive ones are opportunities, which have their own responses such as exploit, enhance and share.

1443. **B** — Pressman's business risks are market risk (no one wants the product), strategic risk (it no longer fits the company's strategy), sales risk (the sales force cannot sell it), management risk (losing senior support) and budget risk (losing budget or staff).

1444. **A** — This is Pressman's management risk. Strategic risk is about the product no longer fitting the overall business strategy, and market risk is about nobody wanting the product. Project risk is a separate top-level category.

1445. **A** — Pressman's components are performance (will it meet requirements), cost (will the budget hold), support (will it be easy to correct, adapt and enhance) and schedule (will it be delivered on time).

1446. **A** — Refinement breaks a broad risk into more detailed ones written as condition-transition-consequence: the current condition, how it could turn into a problem, and what would follow. Detailed statements are easier to mitigate and monitor.

1447. **D** — The table is sorted by probability and then impact, and a cut-off line is drawn. Risks above it get RMMM planning; those below are re-evaluated later as lower priority, not deleted.

1448. **C** — Generic risks are potential threats to every project. Product-specific risks come from the particular technology, people and environment of the software being built, so they need people with detailed knowledge of the project to spot them.

1449. **C** — Sommerville's bands are very low (under 10%), low (10 to 25%), moderate (25 to 50%), high (50 to 75%) and very high (over 75%). Students often confuse high with very high.

1450. **A** — Sommerville uses catastrophic, serious, tolerable and insignificant. Catastrophic, critical, marginal and negligible is Pressman's impact scale, rated 1 to 4, which is the usual source of confusion.

1451. **D** — The contingency reserve is sized from the identified risks, often from their total expected value, and the project manager can use it. The management reserve is extra budget for risks nobody identified and usually needs senior approval to release.

1452. **B** — Risk-based testing uses likelihood and impact of failure to prioritise testing, so critical and fragile areas get deeper and earlier tests. Low-risk areas still get tested, just less intensively.

1453. **B** — Each failure mode is rated, typically 1 to 10, for severity, likelihood of occurrence and difficulty of detection, and the three are multiplied. A high detection score means the failure is hard to detect, which raises the RPN.

1454. **A** — The SEI principles include maintaining a global perspective, taking a forward-looking view, encouraging open communication, integrating risk management, emphasising a continuous process, a shared product vision and teamwork.

1455. **B** — Insurance, warranties and fixed-price contracts move the loss to someone else, at the cost of a premium or a higher price. The event can still happen and still disrupt the project, so transfer usually leaves residual risk and may add vendor risk.

1456. **D** — Monte Carlo runs the schedule thousands of times, each time drawing task durations from their ranges. The result shows, for example, the chance of finishing by a given date, which a single-point estimate cannot give.

1457. **B** — RE = P × C = 0.15 × 800,000 = Rs 120,000. Using the full cost ignores probability, 680,000 subtracts instead of multiplying, and 53,333 divides the cost by 15.

1458. **C** — The score is 3 × 3 = 9, which falls in the 8 to 14 amber band. A score of 6 comes from adding instead of multiplying.

1459. **C** — Sommerville gives inefficient generated code as his example of a tools risk, which covers CASE and other support tools. Technology risks concern the hardware and software the system itself runs on, such as a database that is too slow.

1460. **C** — Risks about the people on the team, such as recruitment failure, illness or missing skills, are people risks. Organisational risks come from the wider organisation, for example restructuring or financial problems.

1461. **C** — Exploit tries to make sure a positive risk definitely happens. Enhance only increases its probability or impact, share gives part of it to a partner who can capture it better, and transfer is a threat response.

1462. **B** — RE = 0.4 × 15 = 6 days. Exposure can be expressed in time as well as money. 9 days is 0.6 × 15, the chance it does not happen, and 37.5 divides instead of multiplying.

1463. **A** — Sommerville pairs each risk type with indicators. Requirements risks show up as many change requests and customer complaints. Poor morale signals people risks, late hardware signals technology risks and reluctance to use tools signals tools risks.

1464. **C** — 450,000 ÷ 9,000,000 = 0.05, which is 5%. Pressman compares overall risk exposure with the project estimate to judge whether the plan needs a contingency or a rethink.

1465. **B** — Expected = (4 + 4 × 6 + 14) ÷ 6 = 42 ÷ 6 = 7 weeks. Standard deviation = (14 − 4) ÷ 6 ≈ 1.67 weeks. The long pessimistic tail pulls the expected value above the most likely estimate.

1466. **D** — RPN = severity × occurrence × detection = 8 × 4 × 6 = 192. Adding gives 18, and multiplying only two factors gives 32 or 48.

1467. **B** — A: 9 × 2 × 2 = 36. B: 5 × 6 × 5 = 150. B is more likely and harder to detect, so it ranks higher by RPN. Many teams still review any severity-9 mode separately, but by RPN alone B comes first.

1468. **A** — RE before = 0.5 × 600,000 = 300,000. RE after = 0.2 × 600,000 = 120,000. RRL = (300,000 − 120,000) ÷ 40,000 = 4.5. 7.5 ignores the remaining exposure, 3 uses the after value, and 0.22 inverts the ratio.

1469. **D** — Exposures are 200,000 + 100,000 + 90,000 + 300,000 = Rs 690,000. 8,550,000 sums raw costs, which assumes every risk happens.

1470. **C** — R1 is likely but only marginal (3). R3 is catastrophic but below 40%. R2 and R4 meet both conditions. Reading impact 3 as serious is the usual mistake, because on Pressman's scale a lower number is worse.

1471. **D** — Sommerville marks specification delays, requirements change and size underestimate as both project and product risks, because they hit the schedule and the quality of what is built. Staff turnover is project only, and technology change is business.

1472. **A** — Expected delays are 0.3 × 10 = 3, 0.5 × 4 = 2 and 0.1 × 30 = 3, which sum to 8 days. 44 days assumes all three happen, and 3 days takes only one risk.

1473. **A** — Sharing gives part of an opportunity to a third party better able to capture it, such as a partner or joint venture. Exploit means making sure the opportunity happens through your own action, and mitigate is a threat response.

1474. **A** — Doing nothing: 0.3 × 2,000,000 = 600,000. X: 250,000 + 0.15 × 2,000,000 = 550,000. Y: 450,000 + 0.3 × 500,000 = 600,000. X is the cheapest. Y's 450,000 forgets the remaining exposure.

1475. **B** — RE before = 0.4 × 1,200,000 = 480,000. RE after = 0.25 × 800,000 = 200,000. RRL = (480,000 − 200,000) ÷ 140,000 = 2.0. 1.29 changes only P, 3.43 ignores the remaining exposure, and 0.5 inverts the ratio.

1476. **A** — Before: 8 × 5 × 7 = 280. After: 8 × 3 × 2 = 48. The fall is (280 − 48) ÷ 280 ≈ 82.9%. Severity is unchanged, because the actions reduce how often the failure happens and how hard it is to spot, not how bad it is.

1477. **A** — Mean = (20 + 4 × 26 + 44) ÷ 6 = 28. Standard deviation = (44 − 20) ÷ 6 = 4. About 84% of a normal distribution lies below mean + 1σ, so 32 days. Using the pessimistic value as the deadline wastes buffer.

1478. **D** — Lateness is a consequence, not a risk you can mitigate. Refinement asks what condition could cause it, such as "the payment vendor's API is unfinished, so integration may slip three weeks". A specific risk can be given triggers, an owner and a response.

1479. **C** — Shared ownership in practice means no ownership. Naming one person per risk makes someone responsible for watching triggers and starting the response. The rating scale or analysis method would not explain why visible triggers were ignored.

1480. **B** — Transfer moves money, not consequences. Customers still lose data, work is still delayed and reputation still suffers. Mitigation (backups) and transfer (insurance) work together; each covers a different part of the loss.

1481. **C** — Short feedback loops do not stop misunderstandings, but they surface them within one sprint, which cuts the cost of rework. That reduces impact, which is mitigation. The risk still exists and should stay in the register.

1482. **D** — Sommerville lists failure to meet the agreed schedule and failure to clear reported defects as indicators of estimation risk: the work was underestimated. Tools risks show as complaints about tools, and organisational risks as gossip or management inaction.

1483. **D** — Acceptance is valid, but passive acceptance means simply dealing with the consequences when they come. Active acceptance prepares a contingency, such as a documented rebuild procedure or a spare machine, so the impact stays small.

1484. **D** — RE before = 600,000. M1: RE 400,000, RRL = 200,000 ÷ 50,000 = 4, total 450,000. M2: RE 50,000, RRL = 550,000 ÷ 300,000 ≈ 1.83, total 350,000. Leverage measures efficiency per rupee; total expected cost measures the outcome, so they can disagree.

1485. **C** — Total exposure = 5 × 0.2 × 500,000 = 500,000. All five together has probability 0.2^5 = 0.00032. Holding the full sum ties up money for a near-impossible case; a reserve near the total exposure, with management reserve for surprises, is the usual practice.

1486. **A** — P(both) = P(leaves) × P(late | leaves) = 0.3 × 0.9 = 0.27, more than double 0.12. Multiplying the plain probabilities is valid only for independent events. Linked risks should be analysed together, or the register understates the danger.

1487. **B** — Convert to a common unit first. R1 = 0.5 × 20 = 10 days × 40,000 = Rs 400,000. R2 = 0.2 × 900,000 = Rs 180,000. Ranking a mixed-unit register without conversion gives meaningless orders.

1488. **B** — Gold-plating is the team adding unrequested features. Boehm's suggested techniques include requirements scrubbing, prototyping and cost-benefit analysis. A stream of requirement changes comes from the customer, not from the developers' own additions.

1489. **B** — RRL = (200,000 − 120,000) ÷ 120,000 ≈ 0.67. Below 1 means each rupee spent removes less than a rupee of expected loss. That can still be right when the impact is a safety or legal matter that money figures do not capture.

1490. **D** — Residual risk is what remains of the original risk, poor quality, after the response. Secondary risk is a new risk created by the response, here dependence on a vendor that could fail. Both belong in the register with their own owners.

1491. **D** — Falling exposure is the usual sign that responses work. A rise after an initial fall means new risks were added or existing ones got worse. That is exactly what monitoring is for: it should prompt a review of responses and escalation if needed.

1492. **C** — Numbers invented without data look rigorous but are guesses. Bands such as low/moderate/high are honest about the uncertainty and fast to apply. Quantitative work is worth it for the few risks that matter most, once there is evidence to estimate from.

1493. **B** — STRIDE pairs each threat with a property: Spoofing–authentication, Tampering–integrity, Repudiation–non-repudiation, Information disclosure–confidentiality, Denial of service–availability, Elevation of privilege–authorisation. Tampering means unauthorised change to data or code.

1494. **D** — Flaws found in requirements or design are far cheaper to fix than those found after release. A secure SDLC adds security requirements, threat modelling, secure coding and automated checks from the start.

1495. **B** — SAST is white-box: it inspects code paths, for example tracing user input into a SQL string. It runs early, even in the IDE or CI pipeline, but can report false positives because it does not see runtime behaviour.

1496. **C** — DAST is black-box: it probes the deployed app much as an attacker would, so it finds runtime and configuration problems. It cannot test paths it never reaches and does not point to the exact line of code.

1497. **A** — Least privilege gives each user, process or service only the rights it needs for its task. If the reporting service is compromised, the attacker cannot change or delete data through it.

1498. **B** — Parameterised queries send the SQL text and the user values separately, so input is always treated as data, never as SQL. Hiding errors and encryption help in other ways but do not stop an injected query from running. OWASP lists parameterisation as the first defence.

1499. **C** — Passwords should never be recoverable. A unique salt per user defeats precomputed rainbow tables, and a deliberately slow function makes brute force expensive. Fast hashes such as MD5 or SHA-256 alone are too quick to guess against.

1500. **D** — Defence in depth layers independent controls so that if one fails, others still stand. No single layer is assumed to be perfect.

1501. **B** — Broken Access Control moved to first place in 2021, up from fifth in 2017, and it stays first in the 2025 edition. Injection, which now includes XSS, fell to third in 2021. Rankings change between editions, so always quote the year.

1502. **C** — Repudiation means a user can deny an action because there is no trustworthy proof. Non-repudiation controls such as signed transactions and protected audit logs supply that proof. Rate limiting addresses denial of service, and encryption addresses information disclosure.

1503. **A** — A trust boundary marks a change in privilege or control, such as browser to server or app to third-party API. Data crossing it must be authenticated, validated and protected, so STRIDE analysis focuses there.

1504. **A** — Base access decisions on permission rather than exclusion: if a rule is missing or a check fails, the default outcome is deny. An allow-by-default design fails open, which is far more dangerous.

1505. **A** — Complete mediation means checking authority on every access. Caching a permission check, or checking only on the first page of a workflow, breaks this principle and leads to bugs such as insecure direct object references.

1506. **C** — Misuse cases model hostile actors and their goals alongside normal use cases, such as "attacker guesses passwords". Each one suggests a countermeasure requirement, such as account lockout or rate limiting.

1507. **C** — Fuzzers mutate or generate input automatically and watch for crashes, hangs or sanitiser errors. It is very effective at finding memory-safety and parsing bugs that hand-written tests miss.
1508. **A** — SCA builds an inventory of dependencies, including transitive ones, and matches them against vulnerability databases. It addresses the OWASP 2021 category Vulnerable and Outdated Components.
1509. **D** — The attacker modifies data in transit to the server, which is tampering with integrity. The fix is to never trust client-supplied prices: look them up on the server.
1510. **C** — Without trustworthy records, the user can deny the action. Non-repudiation controls such as audit logs with user identity and timestamps, or signed transactions, counter this threat.
1511. **D** — Scores: T1 = 8, T2 = 10, T3 = 9. T2 has the highest risk score even though it is the least likely, because its impact is severe.
1512. **D** — The f-string builds SELECT name FROM users WHERE name = 'x' OR '1'='1'. The OR condition is true for every row, so all 3 users are returned. This is classic SQL injection.
1513. **A** — With a placeholder, the whole string x' OR '1'='1 is treated as one name value. No user has that name, so 0 rows return. The injected quote never becomes SQL syntax.
1514. **C** — html.escape converts &, <, > and, by default, both quote characters into entities. The browser then shows the text instead of running it, which defeats stored XSS in an HTML body context. With quote=False, the quotes would be left alone.
1515. **C** — The server trusts the id without checking ownership. Every request must verify authorisation for that specific object (complete mediation). Unguessable IDs help a little, but the server-side check is the real fix.
1516. **B** — This is cross-site request forgery (CSRF): the browser sends the victim's cookies automatically. A per-session or per-request token that the attacker's page cannot read, plus SameSite cookies, lets the server reject forged requests. HTTPS alone does not stop it.
1517. **B** — Regenerating the session identifier on authentication means the ID the attacker planted is no longer valid. Session IDs should also be random, sent only over HTTPS and kept in cookies marked Secure and HttpOnly.
1518. **B** — DREAD score = (8 + 6 + 7 + 9 + 5) / 5 = 35 / 5 = 7. Microsoft no longer recommends DREAD because ratings are subjective, but it still appears in exams and older material.
1519. **A** — Stack traces reveal internals an attacker can use to choose exploits. Users should get a generic message with a reference number, while full details go to protected server logs.
1520. **A** — Three ../ segments climb from /srv/app/files to /, so normpath gives /etc/passwd. This is path traversal. The fix is to resolve the final path and check that it still starts with the base directory, or to map IDs to files instead of using raw names.
1521. **A** — Likelihood = (7 + 6 + 8 + 7) / 4 = 7, which is High. Impact = (4 + 5 + 3 + 4) / 4 = 4, which is Medium. This mirrors the OWASP Risk Rating Methodology, which then combines the two levels in a matrix.
1522. **A** — MD5 is fast and unsalted here, so identical passwords share hashes and GPUs can try billions of guesses per second. A slow, salted algorithm fixes both problems. Because the old hashes cannot be reversed, you upgrade each one when the user next logs in.
1523. **D** — This is a design flaw, not a coding slip, so late testing can only find it, not prevent it. A secure SDLC threat-models the design, where the trust problem would have been visible on the DFD and cheap to change.
1524. **C** — Deny-lists always miss some variant; there are many ways to run script in HTML. Encoding output for its context makes any markup display as text, and allow-list validation accepts only what is expected. A Content Security Policy adds another layer.
1525. **D** — Client-side checks are for user experience only. Anyone can disable them or call the API directly with tools such as curl or an intercepting proxy. The server is the trust boundary and must re-validate everything.
1526. **B** — SAST covers code that DAST's crawler never reaches, such as admin pages, and points to exact lines, but it lacks runtime context. DAST sees configuration and runtime behaviour but not unreached code. Mature secure SDLCs run both, triaging SAST findings.
1527. **D** — A data flow can be modified (T), read (I) or blocked (D). Spoofing and repudiation apply to external entities and processes, and elevation of privilege applies to processes. Processes are subject to all six.
1528. **C** — Anyone who discovers the endpoint can call it directly. Authorisation must be enforced on the server for every request (complete mediation). Hiding UI elements is only a convenience.

1529. **B** — An unsalted hash maps the same password to the same value, so duplicates stand out and one rainbow-table lookup or guess breaks every matching account. Per-user salts make each hash unique, and a slow function such as Argon2 makes guessing expensive.
1530. **D** — With admin rights, an injection can read every table, drop data or even run system commands. A restricted account contains the blast radius. This is least privilege used as defence in depth.
1531. **D** — The vulnerable code is not in the team's own source, so reviews and SAST on their code will not see it. SCA inventories every direct and transitive dependency and checks each build against vulnerability databases, so new advisories are caught quickly.
1532. **B** — Scores: T1 = 20, T2 = 10, T3 = 16, T4 = 6. Order: T1, T3, T2, T4. Putting T2 high because it has the highest impact ignores its low likelihood.
1533. **A** — Denial of service makes a system or resource unavailable to legitimate users, for example by flooding an API. Information disclosure breaks confidentiality, repudiation breaks non-repudiation and spoofing breaks authentication.
1534. **B** — Elevation of privilege means gaining capabilities you were not authorised to have, which violates authorisation. Spoofing would be pretending to be another identity, for example by logging in with stolen credentials.
1535. **A** — The attack surface is everything an attacker can reach: endpoints, forms, open ports, file uploads, configuration interfaces. Reducing it, for instance by disabling unused features, cuts the chances of a flaw being reachable.
1536. **C** — CWE (Common Weakness Enumeration) lists weakness classes, such as CWE-89 for SQL injection. CVE (Common Vulnerabilities and Exposures) gives an ID to one concrete flaw in one product and version. Severity scoring is done separately with CVSS.
1537. **A** — DevSecOps puts checks such as SAST, dependency scanning and secret scanning into continuous integration and delivery, so issues are caught on every change. It is a practice and culture, not a single tool or a separate gatekeeping team.
1538. **C** — Open design echoes Kerckhoffs's principle: assume the attacker knows how the system works. Security that relies on hiding the mechanism (security through obscurity) fails as soon as the design leaks.
1539. **D** — White-box testers get full internal knowledge, which allows deep coverage. Black-box testers start with nothing, like an outside attacker, and grey-box testers get partial knowledge such as a user account or some documentation.
1540. **B** — Threat modelling analyses the architecture, data flows and trust boundaries, so it belongs in design, when flaws are cheapest to fix. Verification is for testing such as DAST and fuzzing, and response handles incidents after release.
1541. **D** — An allow-list defines what valid input looks like, such as 3 to 16 letters, digits or underscores, and rejects the rest. A deny-list tries to block known-bad strings and is easy to bypass with variations. Escaping on output is a separate control.
1542. **A** — CVSS v3.x bands are None 0.0, Low 0.1 to 3.9, Medium 4.0 to 6.9, High 7.0 to 8.9 and Critical 9.0 to 10.0. A score of 7.5 is therefore High.
1543. **A** — Simple, small mechanisms are easier to review and test, and leave fewer places for errors. The principle is about design complexity, not about money.
1544. **B** — Separation of privilege requires more than one condition, or more than one person, before an action is allowed, so one compromised account is not enough. Least privilege is about each account holding only the rights it needs.
1545. **D** — Shared resources, such as a common temporary directory or shared cache, can carry information or interference between users. Keeping mechanisms per-user where practical limits this. It is a different idea from least privilege.
1546. **A** — A04:2021 Insecure Design covers missing or weak security controls in the design itself, which perfect code cannot fix. It calls for threat modelling and secure design patterns. Misconfiguration, Injection and Cryptographic Failures existed in older forms.
1547. **D** — Git keeps every earlier version, so anyone with access can read the old commit. Once a secret has been committed it must be treated as leaked: revoke it, issue a new one and store secrets in a secret manager or environment configuration.

1548. **B** — SAMM measures how mature an organisation's practices are in areas such as governance, design, implementation, verification and operations, and helps plan improvement. CVSS scores individual vulnerabilities.

1549. **C** — The compiler inserts a canary value and checks it before the function returns; a changed canary means the stack was overwritten. Non-executable memory (DEP/NX) and address randomisation (ASLR) are separate mitigations.

1550. **D** — An external entity can be impersonated (spoofing) and can deny its actions (repudiation). Data flows and data stores face tampering, information disclosure and denial of service, and processes face all six.

1551. **C** — Between checking a condition, such as "this file is safe" or "this coupon is unused", and using the resource, another process or request can change it. The fix is to make the check and the use one atomic operation.

1552. **D** — token_hex(16) produces 16 random bytes, and each byte becomes two hex characters, so the string is 32 characters long. The secrets module uses a cryptographically secure generator, unlike random.

1553. **D** — Different salts make the hashed input different, so the hashes differ even for the same password. A SHA-256 hex digest is 64 characters. Real systems should also use a slow hash such as bcrypt or Argon2 rather than plain SHA-256.

1554. **D** — Online guessing is a spoofing threat, and it is countered by limiting attempts, ideally combined with MFA. Encryption protects stored data, not the login endpoint, and logs alone do not stop the attack.

1555. **D** — Reading data you should not see breaks confidentiality, which is information disclosure. Encrypting the channel with TLS stops eavesdropping. The attacker did not change the messages, so this is not tampering.

1556. **A** — encodeURIComponent encodes & as %26, = as %3D and the space as %20, so the value cannot break the query string into extra parameters. HTML entity encoding such as & is for HTML contexts, not URLs.

1557. **B** — Anyone with log access, or anyone who steals the logs, can read passwords. Logs should record events, such as a failed login for a user ID, without secrets. Hashing usernames does not address the passwords.

1558. **C** — This is clickjacking. Telling the browser that the page may not be framed by other origins stops it. Output escaping prevents XSS and parameterised queries prevent SQL injection, neither of which is happening here.

1559. **A** — SAST and SCA are fast, automatic and work on code and manifests, so they suit every build. Pen tests and bug bounties are valuable but manual and periodic, and training does not check code.

1560. **C** — fullmatch needs the whole string to be 3 to 16 letters, digits or underscores. ali_99 and Omar2026 pass. ab is too short, sara khan has a space and x'-- has a quote and dashes. The allow-list rejects anything it does not recognise.

1561. **B** — With quote=False, html.escape only converts &, < and >, so the double quotes pass through and close the value attribute early, adding an onmouseover handler. Encoding must match the context; in attributes, quotes must be escaped too.

1562. **C** — 65536 × 65536 = 2^{32}, which does not fit in 32 bits, so unsigned arithmetic wraps to 0. If this value is passed to malloc and the code then writes 65536 records, it overflows the heap. Check for overflow before multiplying, for example with count > SIZE_MAX / size.

1563. **C** — A normal == comparison can stop at the first different character, so response time leaks how much of the guess was right. hmac.compare_digest takes time independent of where the strings differ. The is operator checks object identity, not content.

1564. **A** — SameSite tells the browser not to attach the cookie to cross-site requests (Lax still allows top-level GET navigation). HttpOnly hides the cookie from JavaScript and Secure restricts it to HTTPS; neither stops cross-site sending.

1565. **A** — Mass assignment happens when frameworks bind request data straight onto objects, including fields the user should never set. Defining exactly which fields a user may change, or using a separate input model, stops it.

1566. **C** — The server is tricked into requesting an internal address the attacker cannot reach directly. SSRF is A10 in the OWASP Top 10:2021. CSRF abuses the victim's browser, not the server.

1567. **C** — Letting the token choose its own algorithm means an unsigned token can pass. The server must decide the algorithm and key and always verify. Base64 is encoding, not encryption, and a short expiry does not stop forged tokens.

1568. **B** — Under STRIDE-per-element, data stores face T, I and D, plus R when they hold audit logs. Data flows face T, I and D only, and external entities face S and R. A trust boundary is not an element that is attacked; it marks where privilege changes.

1569. **D** — normpath gives /srv/app/files_backup/keys.txt, which starts with the string /srv/app/files even though it is a sibling folder. A string prefix check is not a path check; compare with os.path.commonpath or check for BASE + "/".

1570. **A** — The query becomes SELECT name FROM users WHERE id = 0 UNION SELECT pw FROM users. No row has id 0, but the UNION adds every password. Numeric contexts are injectable too, because no quote needs to be broken; parameters fix this.

1571. **D** — AND: 0.5 × 0.4 = 0.2. OR: 1 − (1 − 0.2)(1 − 0.1) = 1 − 0.72 = 0.28. Adding 0.2 and 0.1 gives 0.30, which double-counts the chance both happen. 0.02 multiplies all three.

1572. **A** — $62^8 \approx 2.18 \times 10^{14}$. Divided by 10^{10} per second that is about 21,800 s, roughly 6 hours. Divided by 10^4 it is about 2.18×10^{10} s, roughly 690 years. The slow hash multiplies the attacker's work, which is why password hashes must be slow.

1573. **B** — TLS stops eavesdropping and tampering on the wire, but the request is legitimate as far as TLS is concerned. The flaw is broken access control (IDOR). Hiding or encoding IDs only obscures them; the server must check that the record belongs to the user.

1574. **B** — When a security check cannot decide, the safe default is to deny. Returning True on any exception turns an outage, or an attacker-induced error, into full access. Return False, and alert operators.

1575. **D** — A deny-list always misses variants such as other extensions or letter case. Allow-listing types, generating server-side names and storing uploads where they are never executed removes the whole class of attack.

1576. **B** — SAST reasons about patterns, so it can flag safe code. Triage means tracing where the value comes from. If no user input reaches it, it is a false positive, but making the safety explicit, for example mapping inputs to fixed column names, keeps future changes safe.

1577. **C** — Good requirements are specific and testable. Concrete statements, about authentication, logging, encryption or lockout, can be designed, implemented and verified. Standards such as OWASP ASVS give ready-made testable requirements.

1578. **C** — A09:2021 covers insufficient logging, monitoring, alerting and response. Logs that nobody watches cannot catch an attack. The fix is alerting on suspicious events and a tested incident response process.

1579. **B** — A signature proves the request was not altered and came from a key holder, but not that it is fresh. Adding a unique nonce or timestamp, signing it and tracking used ones stops replays.

1580. **A** — Different messages and timings tell an attacker which accounts exist, which helps targeted guessing and phishing. A single message and the same work for every attempt, for example hashing a dummy password, remove the signal.

1581. **D** — This is dependency confusion, a supply-chain attack where the resolver prefers the higher public version. Scoping internal packages to the private registry and pinning exact, hash-verified versions stops it.

1582. **D** — CSP blocks injected inline scripts only if inline scripts are disallowed. Nonces or hashes let the site's own inline scripts run while blocking injected ones. CSP is a second layer; output encoding remains the primary fix.

1583. **B** — Both requests read used = false before either marks it used. A conditional update, a transaction with locking or a unique constraint makes only one request succeed. Speeding up credit() narrows the window but does not close it.

1584. **D** — SCA compares dependencies with vulnerability databases. Missing headers are a runtime configuration issue that DAST sees in HTTP responses. SAST reads source code and traces input into queries. Each tool sees what the others cannot.

1585. **A** — Unpickling untrusted data can call any function the attacker chooses, so this is a full compromise (elevation of privilege). Data formats like JSON cannot run code, and an HMAC signature stops tampering. Base64 is encoding, not encryption.

1586. **C** — Defence in depth means layers, not substitutes. Attackers routinely bypass WAF signatures with alternative syntax and encodings. Fixing the query removes the vulnerability; the WAF then catches some attempts at other flaws.

1587. **B** — Treating the internal network as a trust boundary means one foothold reaches everything. Zero-trust designs authenticate each service, for example with mutual TLS or signed tokens, and authorise each call, which limits lateral movement.

1588. **C** — The V-model pairs each development phase on the left of the V with a test level on the right, and the test plan for each level is produced alongside its development phase. Plain waterfall leaves test planning to the testing phase.

1589. **B** — The spiral model is risk-driven. Each loop starts by setting objectives, then assesses and reduces the main risks (often with a prototype), develops and validates, and plans the next loop. How many loops occur depends on the risks found.

1590. **A** — Pressman presents prototyping and the spiral model as evolutionary models: they produce increasingly complete versions of the software as understanding grows. The waterfall and V-model are his linear (sequential) examples.

1591. **C** — Incremental development grows the product slice by slice (login, then payments, then reports). Iterative development revisits the whole system, refining it each cycle. Most modern processes, including Scrum, combine the two.

1592. **B** — Sommerville defines plan-driven processes as ones where all activities are planned in advance and progress is measured against that plan. In agile processes, planning is incremental, so it is easier to change the process to reflect changing requirements. Plan-driven processes can still deliver incrementally.

1593. **D** — In the waterfall, users usually see working software only after all phases, including system testing, are finished. Incremental, Scrum and evolutionary prototyping all deliver something usable or reviewable early and repeatedly.

1594. **D** — Volatile requirements, a small co-located team, an available product owner and a need for frequent feedback are the conditions agile methods were designed for. Scrum's two-week sprints give a usable increment and a review each cycle.

1595. **A** — The spiral model builds risk assessment into every loop and ends each loop with a review and a decision on whether to continue. That suits large, costly projects with major technical unknowns.

1596. **B** — The prototype exists only to clarify requirements and is then discarded, since the real system is built separately on another stack. In evolutionary prototyping the prototype itself grows into the product.

1597. **D** — The V-model links every development phase to a matching test level and produces test plans early, which gives the traceability that certification demands. The other options either produce little formal documentation or none.

1598. **A** — Stable, well-understood requirements, an experienced team and a fixed-scope contract are the conditions where a sequential waterfall works well. Its documentation and clear phases suit fixed-price contracts.

1599. **D** — The system is delivered in planned increments, each usable on its own, with feedback from earlier increments shaping later ones. A single waterfall or V-model delivery would put nothing live until the end.

1600. **A** — Sommerville's three general models are the waterfall, incremental development, and integration and configuration. The last is built around finding reusable components or systems, configuring them and integrating them, as here.

1601. **C** — Waterfall delivers at 2 + 2 + 5 + 3 = month 12. Incremental delivers its first increment at 2 + 3 = month 5. 12 − 5 = 7 months earlier.

1602. **A** — Frequent requirement change is the main reason to avoid waterfall, where change after sign-off is expensive. Documentation is not exclusive to plan-driven models; an incremental or agile process can produce what the ministry needs.

1603. **C** — System tests check the system against the specification (verification). The users' complaint is a validation failure: the specification itself was wrong. Because the V-model involves users mainly at the start and the end, such errors surface late. Prototypes or early increments put the system in front of users sooner.

1604. **C** — Incremental: 0.4 × 4 + 0.35 × 4 + 0.25 × 4 = 4.00. Scrum: 2.0 + 0.70 + 1.25 = 3.95. V-model: 0.4 + 1.75 + 0.25 = 2.40. Waterfall: 2.05. The unweighted sums tie at 12, but the weights favour Incremental's stronger traceability.

1605. **B** — Upfront complete specification, frozen requirements and one big delivery are the defining traits of the waterfall. The label does not change the process; agile requires short iterations, frequent delivery and acceptance of change.

1606. **B** — Sommerville lists large size, distributed teams, safety-critical systems and external regulation as factors that favour plan-driven approaches, while noting that most real projects mix plan-driven and agile practices. A plan-driven backbone with incremental delivery fits all four factors.

1607. **D** — Incremental releases at months 5 and 8 have happened by month 9, so 4 + 4 = 8 features are live. The waterfall delivers nothing until month 12. Early value and early feedback are the main advantage of incremental delivery.

1608. **B** — UML visibility markers are + public, − private, # protected and ~ package. A protected feature is visible to the class and its subclasses.

1609. **B** — A dashed line with a hollow triangle is interface realisation: Wallet provides the operations Payable specifies. Generalisation uses a solid line with a hollow triangle, and dependency uses a dashed line with an open arrowhead.

1610. **C** — UML draws synchronous messages with a filled arrowhead and asynchronous messages with an open arrowhead, both on solid lines. Replies use a dashed line, and a creation message is a dashed line with an open arrowhead pointing at the new lifeline's head.

1611. **D** — When a token reaches an activity final node, the entire activity stops, including other concurrent flows. A flow final node simply consumes the token that reaches it, and other flows carry on.

1612. **C** — UML underlines class-scope (static) features. An abstract class or operation is shown in italics (or with {abstract}), a derived attribute has a leading '/', and private visibility is '−'.

1613. **D** — A deployment diagram models the run-time hardware or execution environments (nodes) and the artifacts deployed to them. A component diagram shows logical components and their interfaces, without the hardware.

1614. **D** — Multiplicities are read at the far end: a hospital has 2 to 6 wards, and each ward has 10 to 30 beds. The maximum is 6 × 30 = 180 (and the minimum is 2 × 10 = 20).

1615. **B** — Invoice creates its lines itself and never shares them, so the lines exist only as parts of that invoice: composition, drawn with a filled diamond at the whole (Invoice) end. Aggregation would suit parts created elsewhere that can outlive the whole.

1616. **C** — Report keeps no reference to a Printer; it only receives one as a method parameter and uses it for the duration of the call. That temporary 'uses' link is a dependency, drawn as a dashed line with an open arrowhead towards Printer.

1617. **C** — An «include» arrow points from the base use case to the use case it always includes. An «extend» arrow points from the optional, conditional use case to the base use case it extends. So Checkout includes Validate Payment, and Apply Coupon extends Checkout.

1618. **B** — checkout() is 1. The loop runs 3 times with 2 messages each, giving 6. The opt guard (6,200 > 5,000) is true, adding 2. charge, receipt and confirmation add 3. Total 1 + 6 + 2 + 3 = 12. Skipping the opt fragment would give 10.

1619. **D** — Calls: checkout() 1, reserve() 4 times (one per item), verify() 0 times because 3,500 > 5,000 is false, and charge() 1. Total 1 + 4 + 0 + 1 = 6. The dashed arrows are replies and are not counted.

1620. **D** — Draft → submit → Submitted → review [42 < 50] → Rejected → edit → Draft → submit → Submitted → review [71 ≥ 50] → Approved. Approved has no edit transition, so the final edit event is ignored and the paper stays Approved.

1621. **A** — There are 2 ways to reach 'Check eligibility' (direct, or after requesting documents once). From there, 'not eligible' gives 1 path to the end, and 'eligible' gives 2 (with or without the scholarship assessment), so 3 paths. Total 2 × 3 = 6.

1622. **B** — In composition a part belongs to at most one composite at a time, so the multiplicity at the composite end must be 1 or 0..1. A wall shared by 2 rooms cannot be a composite part of both; model it as an association or aggregation instead.

1623. **A** — A fork allows the branches to run in any interleaving, but each branch keeps its own order (email before SMS). 'Update roster' can go before, between or after the two messages: 3 orders. The join waits for both branches before continuing.

1624. **A** — A state machine diagram models the states of one object and the event-triggered transitions between them (trigger [guard] / effect), which is exactly an order's life cycle.

1625. **A** — The base use case includes the behaviour, so the arrow starts at 'Register for Course' and points to 'Authenticate'. (It is «extend» whose arrow points towards the base use case.)

1626. **A** — Payable is an abstract base with only abstract operations, acting as an interface, so Team realises it (dashed line, hollow triangle). Players are created elsewhere and survive the team's deletion, so the whole-part link is aggregation, not composition. (If you modelled Payable as an abstract class, the first link would be generalisation instead, but the Player link would still be aggregation.)

1627. **C** — UML distinguishes message kinds by line and arrowhead: a solid line with a filled arrowhead is a synchronous call. A reply must be drawn with a dashed line. Direction alone does not make an arrow a reply.