Introduction to Problem Solving MCQs for NSCT

1. In George Pólya's four-step problem-solving method, what is the first step? Understanding
   A. Devise a plan
   B. Understand the problem
   C. Carry out the plan
   D. Look back at the solution
2. In the IPO model used to describe a problem, what does the letter P stand for? Understanding
   A. Processing
   B. Program
   C. Procedure
   D. Pseudocode
3. Which property of an algorithm requires it to stop after a finite number of steps for every valid input? Understanding
   A. Definiteness
   B. Finiteness
   C. Effectiveness
   D. Input
4. A recipe-style algorithm contains the step "add a little more sugar until it tastes right". Which algorithm property does this step break most directly? Understanding
   A. Finiteness
   B. Output
   C. Definiteness
   D. Input
5. In a standard flowchart, which symbol represents a decision such as "Is marks >= 50?" Understanding
   A. Parallelogram
   B. Diamond
   C. Rectangle
   D. Oval (terminator)
6. In a flowchart, a parallelogram is used for which kind of step? Understanding
   A. A decision with Yes/No branches
   B. Input or output, such as reading a number or printing a result
   C. The start or end of the algorithm
   D. A calculation such as total = total + x
7. Which statement best describes the difference between an algorithm and a program? Understanding
   A. An algorithm is a language-independent sequence of steps; a program is that algorithm written in a specific programming language
   B. An algorithm must run on a computer; a program can be done by hand
   C. An algorithm and a program are the same thing with different names
   D. A program is always shorter than the algorithm it implements

8. Which of these makes a problem well-defined? Understanding
    A. A clear starting state, a clear goal and a known set of allowed operations
    B. It can be solved in under one second
    C. It has only one possible input
    D. It has already been solved by someone else

9. You know the final result of a sequence of operations and must find the starting value, so you undo each operation from the end. Which problem-solving heuristic is this? Understanding
    A. Trial and error
    B. Divide and conquer
    C. Working backwards
    D. Brute-force enumeration

10. What is the main purpose of the "look back" step in Pólya's method? Understanding
    A. Check the result against the original problem and see whether the method can be reused or improved
    B. Re-read the question before starting any work
    C. Choose the programming language for the solution
    D. Collect the input data needed for the plan

11. What is a dry run of an algorithm? Understanding
    A. Tracing it by hand with sample values and recording each variable's value step by step
    B. Running the compiled program with no input
    C. Deleting unused variables before compiling
    D. Running the program on the fastest available computer

12. Which approach starts with the whole problem and splits it into smaller sub-tasks, refining each until it is simple enough to code? Understanding
    A. Bottom-up testing
    B. Trial and error
    C. Working backwards
    D. Top-down design (stepwise refinement)

13. Which statement about pseudocode is correct? Understanding
    A. It must be compiled before it can be checked
    B. It is a machine language understood directly by the CPU
    C. It can only describe algorithms that have no loops
    D. It is an informal, language-independent way to describe an algorithm's steps

14. A good problem definition states the inputs, the required outputs and the constraints. Which of these does it normally NOT need to fix? Understanding
    A. The format of the required output
    B. The valid range of each input
    C. The variable names to use in the final code
    D. Any limits on time or memory

15. In Knuth's classic description of an algorithm, which statement about inputs and outputs is correct? Understanding
    A. It has one or more inputs and one or more outputs
    B. It has zero or more inputs and zero or more outputs
    C. It has exactly one input and exactly one output
    D. It has zero or more inputs and one or more outputs

16. Which algorithm property requires every operation to be basic enough that a person could, in principle, do it exactly in a finite time with pencil and paper? Understanding

    A. Effectiveness
    B. Definiteness
    C. Finiteness
    D. Generality

17. Dry-run this loop. What value of total is printed? Application

```
total = 0
for i in range(1, 6):
    total = total + i * i
print(total)
```

    A. 15
    B. 30
    C. 25
    D. 55

18. A program reads a student's marks in three courses and displays the average and a letter grade. What are the outputs? Application

    A. The marks in the three courses
    B. The three marks and the average
    C. The number 3 and the letter grade
    D. The average and the letter grade

19. A number is doubled, then 7 is subtracted, and the result is divided by 3. The final answer is 9. What was the original number? Application

    A. 10
    B. 24
    C. 20
    D. 17

20. Trace x through this loop. How many steps are counted before x reaches 1? Application

```
x = 10
steps = 0
while x != 1:
    if x % 2 == 0:
        x = x // 2
    else:
        x = 3 * x + 1
    steps += 1
print(steps)
```

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

21. Using a trace table for a and b, what does this code print? Application

```
a = 3
b = 5
a = a + b
b = a - b
a = a - b
print(a, b)
```

A. 3 5
B. 8 3
C. 5 3
D. 5 8

22. An algorithm finds the largest of n numbers by keeping a running maximum and comparing it with each remaining number once. How many comparisons does it make for 10 numbers? Application

A. 9
B. 10
C. 45
D. 20

23. A form accepts ages from 18 to 60 inclusive. Which set of test values best checks the boundaries? Application

A. 18, 39, 60
B. 0, 50, 100
C. 17, 18, 60, 61
D. 20, 30, 40, 50

24. How many times does the inner statement run? (What does the code print?) Application

```
count = 0
for i in range(1, 5):
    for j in range(i, 5):
        count += 1
print(count)
```

A. 16
B. 4
C. 6
D. 10

25. The digits of n are processed one at a time. What does this code print? Application

```
n = 1234
s = 0
while n > 0:
    s = s + n % 10
    n = n // 10
print(s)
```

A. 4
B. 1234
C. 10
D. 4321

26. An algorithm is written as: i = 1; WHILE i ≠ 10: i = i + 2. What happens when it runs? Application

    A. It stops after 5 iterations
    B. It stops after 9 iterations
    C. It never stops, because i takes only odd values and never equals 10
    D. It stops immediately because i starts below 10

27. The requirement is "print Pass if marks are 50 or more", but the code uses IF marks > 50. Which single input exposes the bug? Application

    A. 49
    B. 50
    C. 51
    D. 100

28. Put Pólya's steps in the correct order: (P) devise a plan, (L) look back, (U) understand the problem, (C) carry out the plan. Application

    A. U, P, C, L
    B. P, U, C, L
    C. U, C, P, L
    D. P, C, U, L

29. You must guess a secret whole number from 1 to 100. After each guess you are told "higher" or "lower". Using the best strategy (always guess the middle), what is the most guesses you could need? Application

    A. 10
    B. 50
    C. 6
    D. 7

30. A student tries to swap two values with this code. What does it print? Application

```
a = 3
b = 5
a = b
b = a
print(a, b)
```

    A. 5 3
    B. 5 5
    C. 3 5
    D. 3 3

31. This function is meant to return the largest of three numbers. Which call returns a wrong answer? Analysis

```
def largest(a, b, c):
    m = a
    if b > m:
        m = b
    elif c > m:
        m = c
    return m
```

    A. largest(15, 12, 7)
    B. largest(7, 15, 12)
    C. largest(12, 7, 15)
    D. largest(7, 12, 15)

32. A program meant to print the average of the list [4, 6, 8] prints 9 instead of 6. Which cause fits this output exactly? Analysis

    A. It adds the first element twice
    B. It uses integer division
    C. It divides the sum by n − 1 instead of n
    D. It skips the last element

33. A manager gives you the task "Sort the student records." Before planning, which missing detail matters most for defining the problem? Analysis

    A. Which programming language to use
    B. Which field to sort by, and in ascending or descending order
    C. How many lines the code may have
    D. Whether to use a for loop or a while loop

34. The expected answer for "Education" is 5 vowels, but this code prints 4. What is the cause? Analysis

```
def count_vowels(s):
    n = 0
    for ch in s:
        if ch in "aeiou":
            n += 1
    return n

print(count_vowels("Education"))
```

    A. Only lowercase vowels are checked, so the capital E is missed
    B. The loop skips the last character
    C. The counter starts at 1 instead of 0
    D. The letter u is not counted as a vowel

35. This code should print the sum 1 + 2 + … + n. For n = 10 it prints 45 instead of 55. Which change fixes it for every n? Analysis

```
n = 10
total = 0
for i in range(1, n):
    total += i
print(total)
```

    A. Change total = 0 to total = 10
    B. Change range(1, n) to range(0, n)
    C. Print total + 1 instead of total
    D. Change range(1, n) to range(1, n + 1)

36. Algorithm A tests whether n is prime by trying every divisor from 2 to n − 1. Algorithm B tries divisors from 2 to ⌊√n⌋. Neither stops early. How many divisions does B make for n = 97? Analysis

    A. 9
    B. 8
    C. 48
    D. 95

37. n starts as any positive whole number. Which loop is certain to stop for every such n? Analysis

    A. WHILE n ≠ 0: n = n − 2
    B. WHILE n ≠ 10: n = n + 1
    C. WHILE n > 0: n = n − 2
    D. WHILE n > 0: n = n + 1

38. A function to test whether a number is even gives the correct result for inputs 2, 4 and 6. What can you conclude? Analysis
    A. It is correct for all integers, because three tests agree
    B. It is correct for all even numbers, so it is correct overall
    C. Only that it works for those three inputs; odd numbers, zero and negatives are still untested
    D. It must be wrong, because it was not tested with 0

39. Water lilies on a pond double their covered area every day. The pond is fully covered on day 30. On which day was it half covered? Analysis
    A. Day 29
    B. Day 15
    C. Day 28
    D. Day 20

40. Which plan correctly finds the median of any non-empty list of numbers? Analysis
    A. Take the value in the middle position of the list as given
    B. Sort the list; if the length is odd take the middle value, otherwise average the two middle values
    C. Add all values and divide by the count
    D. Sort the list and always take the element at index length ÷ 2 (integer division)

41. In a standard flowchart, which symbol marks the start or the end of an algorithm? Understanding
    A. Diamond
    B. Oval (terminator)
    C. Parallelogram
    D. Rectangle

42. Which flowchart symbol should hold a calculation such as total ← total + price? Understanding
    A. Diamond (decision)
    B. Oval (terminator)
    C. Parallelogram (input/output)
    D. Rectangle (process)

43. In a trace table used for a dry run, what does each column usually record? Understanding
    A. The changing values of one variable (or the output) as the steps run
    B. One line of the source code
    C. The time each step takes to execute
    D. The memory address of each variable

44. According to structured programming, which three control structures are enough to express any algorithm? Understanding
    A. Input, processing and output
    B. Sorting, searching and recursion
    C. Sequence, selection and iteration
    D. Variables, constants and operators

45. A program reads a circle's radius and prints its area using area = 3.14159 × r × r. Which IPO classification is correct? Understanding
    A. Inputs: r and 3.14159; output: area
    B. Input: r; constant used in processing: 3.14159; output: area
    C. Input: area; output: r
    D. Inputs: r and area; no output

46. Which is the best definition of an algorithm? Understanding
    A. A program written in a high-level language
    B. A diagram drawn with boxes and arrows
    C. A finite sequence of precise steps that solves a problem
    D. Any list of instructions, even if some steps are vague

47. A program compiles and runs without crashing, but it prints the wrong total. What kind of error is this most likely to be? Understanding
    A. Syntax error
    B. Compilation error
    C. Logic error
    D. Input device error

48. Trying every combination of a 3-digit lock, from 000 to 999, until it opens is an example of which strategy? Understanding
    A. Brute force (exhaustive search)
    B. Working backwards
    C. Divide and conquer
    D. Top-down design

49. What is the difference between testing and debugging? Understanding
    A. Testing shows that a fault exists; debugging finds and fixes its cause
    B. Debugging shows that a fault exists; testing finds and fixes its cause
    C. They are two names for the same activity
    D. Testing checks only syntax; debugging checks only logic

50. Which control structure repeats a block of steps while a condition holds? Understanding
    A. Iteration
    B. Selection
    C. Sequence
    D. Decomposition

51. How does a heuristic differ from an algorithm? Understanding
    A. A heuristic always finds the optimal answer but takes longer than an algorithm
    B. A heuristic is an algorithm written in pseudocode instead of code
    C. A heuristic is a procedure that takes no input
    D. A heuristic is a rule of thumb that often helps but does not guarantee a correct or optimal answer

52. Which design approach builds and tests small, general components first, then combines them into the full solution? Understanding
    A. Top-down design
    B. Bottom-up design
    C. Working backwards
    D. Pólya's look-back step

53. Which strategy splits a problem into smaller instances of the same problem, solves those, and then combines their results? Understanding
    A. Brute force
    B. Trial and error
    C. Divide and conquer
    D. Working backwards

54. A student guesses an answer, checks it against the problem's conditions, and uses the result to make a better next guess. What is this strategy called? Understanding

    A. Brute force
    B. Trial and error (guess and check)
    C. Working backwards
    D. Top-down design

55. In Pólya's method, the question "Do you know a related problem you have already solved?" belongs to which step? Understanding

    A. Understand the problem
    B. Devise a plan
    C. Carry out the plan
    D. Look back

56. In a flowchart, what does a small circle labelled with a letter, such as A, indicate? Understanding

    A. A connector: the flow continues at another circle with the same label
    B. A decision with more than two outcomes
    C. The start of the algorithm
    D. A loop counter

57. Why is pseudocode often preferred over a flowchart for a long algorithm? Understanding

    A. A computer can execute it directly without translation
    B. It uses an internationally standard set of symbols
    C. A compiler can check it for syntax errors
    D. It is quicker to write and edit, and it maps closely to program code

58. When a problem is described as a search through states, what is the goal state? Understanding

    A. The state the problem starts from
    B. Any state reachable in one move
    C. The list of operations that are allowed
    D. A state that satisfies the problem's requirements

59. Which of these problems is ill-defined? Understanding

    A. Sort 50 roll numbers in ascending order
    B. Find the greatest common divisor of 84 and 36
    C. Convert 25 °C to Fahrenheit
    D. Make the university website better

60. What does a greedy strategy do at each step? Understanding

    A. Tries every possible choice and keeps the best complete solution
    B. Starts from the goal and works back to the start
    C. Takes the choice that looks best right now and never revisits earlier choices
    D. Splits the problem into two equal halves

61. When a problem is solved recursively, what is the role of the base case? Understanding

    A. It splits the problem into smaller sub-problems
    B. It combines the results of the recursive calls
    C. It solves the smallest instance directly, so the recursion stops
    D. It reads the input before the first call

62. A loop reads exam marks until the user enters a special value that marks the end of input. Why must this sentinel value lie outside the range of valid marks? Understanding

    A. So that the loop runs exactly a fixed number of times
    B. Because the sentinel is added to the total like a normal mark
    C. So that the compiler can detect the end of the loop
    D. Otherwise a genuine mark could be mistaken for the end-of-input signal

63. A loop adds list elements one at a time. The statement "after k passes, total equals the sum of the first k elements" stays true before and after every pass. What is such a statement called? Understanding

    A. A postcondition
    B. A precondition
    C. A loop invariant
    D. A sentinel

64. An algorithm step says: "Write out the exact decimal value of √2 and store every digit." The step is precisely stated, yet it is not a valid algorithm step. Which of Knuth's properties does it break? Understanding

    A. Effectiveness
    B. Definiteness
    C. Input
    D. Output

65. Dry-run this loop. What does it print? Application

```
x = 5
for i in range(3):
    x = x * 2 - 1
print(x)
```

    A. 33
    B. 17
    C. 65
    D. 9

66. An ATM program asks for a PIN and an amount, then dispenses cash and prints a receipt. In the IPO model, which item belongs to processing? Application

    A. Checking that the balance is at least the requested amount
    B. The amount the customer types
    C. The printed receipt
    D. The cash that is dispensed

67. Dry-run this pseudocode for marks = 72. What is output? Application

```
INPUT marks
IF marks >= 80 THEN grade ← "A"
ELSE IF marks >= 65 THEN grade ← "B"
ELSE IF marks >= 50 THEN grade ← "C"
ELSE grade ← "F"
OUTPUT grade
```

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

68. Sana spent half of her money on books, then spent 200 more on lunch. She had 300 left. How much did she start with? Application

    A. 800
    B. 500
    C. 1,000
    D. 1,100

69. How many times does this loop body run? (What does the code print?) Application

```
count = 0
for i in range(2, 20, 3):
    count += 1
print(count)
```

    A. 7
    B. 18
    C. 6
    D. 5

70. A student solves 3x + 5 = 20, gets x = 5, then substitutes 5 back into the equation to confirm 3(5) + 5 = 20. Which of Pólya's steps is the student doing? Application

    A. Carry out the plan
    B. Devise a plan
    C. Look back
    D. Understand the problem

71. Trace Euclid's algorithm. What does this code print? Application

```
a, b = 48, 18
count = 0
while b != 0:
    a, b = b, a % b
    count += 1
print(a, count)
```

    A. 6 3
    B. 6 4
    C. 18 3
    D. 3 6

72. 32 teams enter a knockout tournament: each match eliminates exactly one team, and play continues until one champion is left. How many matches are played? Application

    A. 31
    B. 16
    C. 32
    D. 5

73. A bacteria count triples every day. After 4 days the count is 162. What was the starting count? Application

    A. 2
    B. 6
    C. 54
    D. 18

74. Trace this code. What does it print? Application

```
nums = [3, 8, 1, 9, 4]
m = nums[0]
idx = 0
for i in range(1, len(nums)):
    if nums[i] < m:
        m = nums[i]
        idx = i
print(idx)
```

A. 1
B. 2
C. 3
D. 0

75. Trace this code. What does it print? Application

```
s = "NSCT"
r = ""
for ch in s:
    r = ch + r
print(r)
```

A. NSCT
B. TCSN
C. T
D. N

76. To check for duplicate names, an algorithm compares every pair of students in a class of 10 exactly once. How many comparisons does it make? Application

A. 100
B. 90
C. 45
D. 10

77. A shop gives a discount on bills over 5,000 rupees (strictly more). Bills are whole rupees. Which pair of test values checks the boundary most precisely? Application

A. 4,000 and 6,000
B. 1 and 100,000
C. 5,000 and 5,001
D. 4,999 and 5,000

78. This code reads values until the sentinel −1 appears. What does it print? Application

```
values = [4, 7, 2, -1, 5]
total = 0
count = 0
i = 0
while values[i] != -1:
    total += values[i]
    count += 1
    i += 1
print(total, count)
```

A. 12 4
B. 13 3
C. 18 4
D. 13 4

79. Every pair of people at a meeting of 12 shakes hands once. Solving smaller cases first (2 people: 1, 3 people: 3, 4 people: 6) reveals a pattern. How many handshakes are there? Application

    A. 144
    B. 132
    C. 24
    D. 66

80. A train left 20 minutes after its scheduled time, travelled for 2 h 45 min and arrived at 14:10. At what time was it scheduled to leave? Application

    A. 11:25
    B. 11:45
    C. 10:45
    D. 11:05

81. Trace this recursive function. What does it print? Application

```
def f(n):
    if n <= 1:
        return n
    return n + f(n - 2)

print(f(7))
```

    A. 28
    B. 15
    C. 10
    D. 16

82. Each customer buys half of a seller's remaining eggs plus half an egg. After the third customer, no eggs are left, and no egg was ever cut. How many eggs did the seller start with? Application

    A. 15
    B. 6
    C. 8
    D. 7

83. The Tower of Hanoi is solved by decomposition: move n − 1 disks aside, move the largest disk, then move the n − 1 disks back on top. How many moves are needed for 5 disks? Application

    A. 31
    B. 25
    C. 32
    D. 10

84. Trace this code carefully. What does it print? Application

```
a = 0
b = 1
for _ in range(6):
    a = b
    b = a + b
print(b)
```

    A. 13
    B. 21
    C. 32
    D. 64

85. Using the simpler-cases heuristic, count all squares of any size on a 4 × 4 grid of unit squares. Application

    A. 30
    B. 16
    C. 29
    D. 20

86. A flowchart has the decision "marks >= 50?". Its Yes arrow and its No arrow both lead to the same box, "Print Pass". What is wrong? Analysis

    A. The decision has no effect, because every student is reported as Pass; the No branch should print Fail
    B. The decision should be drawn as a rectangle
    C. "Print Pass" should be drawn as an oval
    D. The condition should be marks > 50

87. An algorithm for making tea reads: 1. Put a tea bag in a cup. 2. Pour the water into the cup. 3. Boil the water. 4. Wait 3 minutes. What is the main flaw? Analysis

    A. It breaks finiteness because it never stops
    B. It has no input, so it is not an algorithm
    C. It needs a loop to repeat the pouring step
    D. The steps are in the wrong sequence: the water must be boiled before it is poured

88. The specification says: multiples of 3 print Fizz, multiples of 5 print Buzz, and multiples of both print FizzBuzz. label(15) returns "Fizz". Which change makes the function meet the specification? Analysis

```
def label(n):
    if n % 3 == 0:
        return "Fizz"
    elif n % 5 == 0:
        return "Buzz"
    elif n % 15 == 0:
        return "FizzBuzz"
    return str(n)
```

    A. Change every elif to if
    B. Move the n % 15 == 0 test so it is checked first
    C. Add an else branch that returns "FizzBuzz"
    D. Replace n % 15 == 0 with n % 3 == 0 and n % 5 == 0 in the same position

89. This code should count the negative numbers in the list (expected 3) but prints 1. What causes the wrong output? Analysis

```
nums = [-2, 5, -7, 3, -1]
for x in nums:
    count = 0
    if x < 0:
        count += 1
print(count)
```

    A. The condition should be x <= 0
    B. range() is needed to loop over the list
    C. count is reset to 0 on every pass, so only the last element's result survives
    D. count += 1 adds 1 twice for each negative number

90. With coins of 1, 3 and 4, a greedy plan pays 6 by always taking the largest coin that fits: 4 + 1 + 1, three coins. What does this show? Analysis

    A. Greedy is optimal, because 3 coins is the minimum for 6
    B. The problem has no solution with these coins
    C. Greedy failed because it should take the smallest coin first
    D. Greedy gave a valid but non-optimal answer here, since 3 + 3 uses two coins

91. Which plan correctly reports whether a list of integers contains any duplicate value? Analysis

    A. Compare each element only with the next element in the original order
    B. Sort the list, then compare each element with the one right after it
    C. Check whether the first element appears again later in the list
    D. Report a duplicate if the sum of the list differs from the sum of its distinct values

92. A function meant to return the largest value in a list passes the tests [1, 2, 3], [4, 7, 9] and [5, 5, 8]. Later it turns out to return the last element. Which single extra test would have exposed the bug? Analysis

    A. [2, 4, 6]
    B. [10, 20, 30]
    C. [9, 2, 5]
    D. [7, 7, 7]

93. Which call makes this function return a wrong answer? Analysis

```
def largest(nums):
    m = 0
    for x in nums:
        if x > m:
            m = x
    return m
```

    A. largest([0, 3, 2])
    B. largest([7])
    C. largest([-1, 4, -3])
    D. largest([-5, -2, -9])

94. A leap-year function is coded as return y % 4 == 0. Under the Gregorian rule (divisible by 4, except century years, which must be divisible by 400), which year exposes the bug? Analysis

    A. 2000
    B. 1900
    C. 2024
    D. 2023

95. A price is raised by 20% and then the new price is cut by 20%. The final price is 960. What was the original price? Analysis

    A. 960
    B. 1,000
    C. 1,152
    D. 800

96. count_words should return the number of words in a sentence. Which input shows that the problem definition of a "word" was never pinned down? Analysis

```
def count_words(s):
    return len(s.split(" "))
```

A. "" (the empty string), which returns 1
B. "one two", which returns 2
C. "NSCT", which returns 1
D. "a b c", which returns 3

97. This binary search prints −1 even though 9 is in the list. What is the cause? Analysis

```
def search(lst, t):
    lo, hi = 0, len(lst) - 1
    while lo <= hi:
        mid = (lo + hi) // 2
        if lst[mid] == t:
            return mid
        if lst[mid] < t:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1

print(search([3, 9, 1, 7, 5], 9))
```

A. mid should be rounded up instead of down
B. The list is not sorted, so binary search's precondition is broken
C. The loop condition should be lo < hi
D. Binary search can never find the largest value in a list

98. is_pal is meant to test whether a string is a palindrome. What happens when it is called with a non-empty string such as "madam"? Analysis

```
def is_pal(s):
    for i in range(len(s)):
        if s[i] != s[len(s) - i]:
            return False
    return True
```

A. It returns True for every string
B. It raises an IndexError on the very first comparison
C. It returns False only for strings of even length
D. It works correctly but makes twice as many comparisons as needed

99. Which plan correctly decides, for every positive whole number n, whether n is a power of 2 (1, 2, 4, 8, ...)? Analysis

A. Answer yes if n is even
B. While n is even, divide it by 2; at the end, answer yes only if n equals 1
C. Answer yes if n is divisible by 4 or n equals 2
D. Keep halving n with integer division until n < 2, then answer yes

100. What is true about this function? Analysis

```
def average(marks):
    total = 0
    for m in marks:
        total += m
        avg = total / len(marks)
    return avg
```

A. It returns the first mark divided by the count, so the answer is too small
B. It returns 0 whenever the list has more than one mark
C. It is fully correct, and returns 0 for an empty list
D. It returns the right average for any non-empty list, but an empty list raises an error because avg is never assigned

Problem Understanding & Analysis MCQs for NSCT

101. In problem analysis, what is a constraint? Understanding

A. The final answer the problem asks for
B. A limit or condition that every acceptable solution must satisfy
C. A step in the algorithm that repeats
D. Any data value given in the question

102. Which of these is a functional requirement for an exam-result portal? Understanding

A. Every page shall load within 2 seconds
B. A student shall be able to download their result card as a PDF
C. The system shall be available 99.5% of the time
D. The interface shall work on screens 360 pixels wide

103. "The search page must return results within 2 seconds" is which kind of requirement? Understanding

A. Functional
B. A domain constraint on input values
C. An assumption
D. Non-functional (performance)

104. What does decomposition mean in problem solving? Understanding

A. Breaking a problem into smaller sub-problems that can be solved separately
B. Removing input data that is not needed
C. Rewriting a solution in a faster language
D. Combining many small problems into one large problem

105. While analysing a problem you decide to treat all months as having 30 days because the question does not say otherwise. What is this? Understanding

A. A functional requirement
B. An assumption, which should be stated explicitly
C. A hard constraint given by the problem
D. An output specification

106. Which of these is an implicit constraint, one the problem does not state but that must still hold? Understanding

A. Input n is at most 1,000, as given in the statement
B. The number of students in a class cannot be negative
C. The answer must be printed to two decimal places, as stated
D. The program must finish within 1 second, as stated

107. In an exam timetable, "no student may have two exams at the same time" must be met, while "avoid exams on Friday afternoon" is only preferred. The second is a: Understanding
    A. Hard constraint
    B. Soft constraint
    C. Functional requirement
    D. Precondition

108. Why is the requirement "The system should be fast" a problem during analysis? Understanding
    A. It is a functional requirement in disguise
    B. It cannot be measured or tested, so nobody can say whether it is met
    C. Speed requirements are not allowed in a specification
    D. It forces the team to use a particular language

109. In Pólya's terms, what is "the unknown" in a problem? Understanding
    A. The quantity or result the problem asks you to find
    B. Data in the problem that is irrelevant
    C. A constraint that is not written down
    D. The method you will use to solve it

110. What is an edge case? Understanding
    A. An input at or beyond the extreme of what is allowed, such as an empty list or the maximum value
    B. The most common input a user will enter
    C. An input that always makes the program crash
    D. A test that checks the user interface layout

111. During decomposition you find that sub-problem B needs the output of sub-problem A. What follows? Understanding
    A. A and B can be solved in any order
    B. A must be completed, or at least its output defined, before B can be finished
    C. A and B should be merged into one sub-problem
    D. B must be solved before A

112. When analysing a word problem, what does abstraction help you do? Understanding
    A. Add extra assumptions to make the problem harder
    B. Split the problem into sub-problems
    C. Convert the answer into a different unit
    D. Ignore details that do not affect the answer and keep only what matters

113. Which list contains only non-functional requirements? Understanding
    A. Students can register; response within 2 seconds; 99.5% availability
    B. Admin can upload results; students can download certificates; password reset by email
    C. Students can choose a test city; pages work on mobile; data encrypted at rest
    D. Response within 2 seconds; 99.5% availability; runs on Android and iOS

114. Which requirement is written in a verifiable way? Understanding
    A. The system should be user-friendly
    B. The system should respond quickly under heavy load
    C. 95% of searches return results within 2 seconds with 500 concurrent users
    D. Search results should be accurate and relevant

115. In a problem specification, what is a precondition? Understanding
    A. A condition that is true after the algorithm finishes
    B. The first line of the algorithm
    C. A condition that must be true before the algorithm starts, such as "the list is sorted"
    D. A requirement about speed

116. Analysis shows that no timetable can satisfy every listed constraint at once. What is the sensible next step? Understanding
    A. Ignore one hard constraint at random
    B. Report that the problem is solved
    C. Keep searching, because a solution must exist
    D. Go back to the stakeholders and relax or prioritise one or more soft constraints

117. A bus with 32 passengers leaves at 8:15 am and travels 90 km at a steady 60 km/h. At what time does it arrive? Application
    A. 9:15 am
    B. 9:45 am
    C. 9:30 am
    D. 10:15 am

118. A problem states n ≤ 100,000 and a time limit of about 1 second (roughly 10^8 simple operations). Which running time fits the constraint? Application
    A. O(n^2)
    B. O(n^3)
    C. O(2^n)
    D. O(n log n)

119. How many three-digit numbers are even and have all three digits different? Application
    A. 360
    B. 450
    C. 328
    D. 320

120. Task A takes 2 h. B (3 h) and C (1 h) can start only after A. D (2 h) can start only after both B and C. With enough people to run tasks in parallel, what is the shortest total time? Application
    A. 8 hours
    B. 6 hours
    C. 7 hours
    D. 5 hours

121. Pay rule: the first 40 hours are paid at 500 per hour; hours above 40 are paid at 1.5 times that rate. What is the pay for 46 hours? Application
    A. 23000
    B. 34500
    C. 24500
    D. 27000

122. x and y are non-negative whole numbers with x + y ≤ 5 and x ≥ 2. How many (x, y) pairs satisfy both constraints? Application
    A. 21
    B. 15
    C. 6
    D. 10

123. "Find the area of a rectangle whose perimeter is 20 cm." What is the correct analysis? Application

A. The data are insufficient: many rectangles have perimeter 20 cm with different areas
B. 25 cm^2
C. 100 cm^2
D. 20 cm^2

124. A requirement says usernames must be 6 to 12 characters long. Which input should be rejected? Application

A. "abcdef"
B. "abcdefghijkl"
C. "abcdefghi"
D. "abcdefghijklm"

125. You must write a function that returns the average of a list of numbers. Which input needs special handling according to careful analysis? Application

A. A list of ten numbers
B. A list containing only positive numbers
C. A list whose numbers are already sorted
D. An empty list, because the sum must be divided by zero

126. How old is Ali? (1) In 5 years Ali will be 20. (2) Ali is older than Sara. Which statement(s) are enough to answer? Application

A. Statement (2) alone is sufficient
B. Both together are needed, but neither alone
C. Statement (1) alone is sufficient
D. Even both together are not sufficient

127. Five meetings must be placed in rooms: 9:00–10:00, 9:30–11:00, 10:00–11:30, 10:30–12:00 and 11:00–12:00. A room freed at a given time can be used by a meeting starting at that same time. What is the minimum number of rooms? Application

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

128. A problem is decomposed into 3 independent sub-problems, and each sub-problem has 4 input classes to test. If each sub-problem is tested on its own, how many test cases are needed in total? Application

A. 12
B. 64
C. 7
D. 81

129. A 4-digit PIN must have no repeated digit and must not start with 0. How many PINs meet both constraints? Application

A. 5040
B. 4536
C. 9000
D. 6561

130. A 5-member committee is chosen from 4 CS and 4 EE students. At most 3 members may be from CS. How many committees are possible? Application

A. 56
B. 48
C. 52
D. 28

131. What is the value of x? (1) $x^2 = 9$. (2) $x > 0$. Which statement(s) are enough to answer? Analysis

A. Both together are needed, but neither alone
B. Statement (1) alone is sufficient
C. Statement (2) alone is sufficient
D. Even both together are not sufficient

132. "Our app's sign-ups doubled in the month after the ad campaign, so the campaign caused the increase." Which unstated assumption does this conclusion depend on? Analysis

A. Nothing else that affects sign-ups changed during that month
B. The ad campaign cost money
C. Sign-ups are counted per month
D. The app had some users before the campaign

133. Exam rules: R1 every exam lasts 3 hours. R2 no exam may start before 2:30 pm. R3 every exam must end by 5:00 pm. R4 each hall holds at most 40 students. Which requirements cannot all be satisfied together? Analysis

A. R1 and R4
B. R2 and R4
C. R3 and R4
D. R1, R2 and R3

134. You must check every subset of n items, and the problem states $n \leq 20$. Is a brute-force $O(2^n \times n)$ approach acceptable for a limit of about 10^8 simple operations? Analysis

A. No: any exponential algorithm is always too slow
B. Yes, but only because 20 is an even number
C. No: 2^20 × 20 is about 10^12
D. Yes: 2^20 × 20 is about 2.1 × 10^7, inside the limit

135. Sub-tasks: Requirements (R); Design (D) needs R; UI mock-ups (U) need R; Coding (C) needs D and U; Testing (T) needs C. Which order respects every dependency? Analysis

A. R, U, C, D, T
B. R, D, U, C, T
C. D, R, U, C, T
D. R, D, C, U, T

136. How many whole numbers from 1 to 50 inclusive are divisible by 3 or by 5? Analysis

A. 23
B. 26
C. 27
D. 3

137. The specification says "Return the second largest value in the list." Which input exposes an ambiguity that needs clarifying before coding? Analysis

A. [7, 9, 9, 4]
B. [1, 2, 3, 4]
C. [10, 5, 8, 2]
D. [3, 6]

138. Requirement: "Split a restaurant bill equally among n friends; each share is a whole number." For a bill of 1000 and 3 friends, each pays 333 and the total collected is 999. What does this reveal? Analysis

    A. The arithmetic is wrong; 1000 ÷ 3 is exactly 333
    B. The bill should always be rounded down to a multiple of 10
    C. The requirement does not say how to handle a remainder, so it must be clarified
    D. Three friends is an invalid input

139. A, B and C have different ages. A is not the oldest. C is not the youngest. B is older than C. Who is the oldest? Analysis

    A. A
    B. C
    C. B
    D. It cannot be determined

140. Given up to 10^6 numbers and a target, decide whether any two of them add up to the target, within about 10^8 operations. Which plan meets the constraint? Analysis

    A. One pass with a hash set: for each x, check whether target − x was seen, then add x
    B. Check every pair with two nested loops
    C. Try every subset of the numbers
    D. For each pair, sort the whole list again before comparing

141. In a problem specification, what is a postcondition? Understanding

    A. A condition that must be true before the solution starts
    B. A limit on the memory or time the solution may use
    C. A condition that must be true when the solution finishes, such as "the list is in ascending order"
    D. A fact the analyst assumed because the problem did not state it

142. Who are the stakeholders of a problem? Understanding

    A. Only the programmers who write the code
    B. Only the person who pays for the project
    C. Everyone affected by or interested in the solution, such as users, clients and administrators
    D. Only the testers who check the solution

143. In the problem "Find two numbers whose sum is 30 and whose difference is 6", what is the condition, in Pólya's terms? Understanding

    A. The sum must be 30 and the difference must be 6
    B. The two numbers that have to be found
    C. The values 30 and 6 taken on their own
    D. Adding the two equations to eliminate one number

144. Which of these is an explicit constraint? Understanding

    A. A person's age cannot be negative
    B. "The list has at most 1,000 items", written in the problem statement
    C. The output should look neat
    D. The programmer prefers to use Python

145. Which activity gathers requirements directly from the people who will use a system? Understanding

    A. Interviews and questionnaires
    B. Unit testing
    C. Compiling the code
    D. Code review

146. On a Wednesday, a client says "Deliver it next Friday." Why is this a problem during analysis? Understanding

A. It is a non-functional requirement
B. It is a hard constraint that can never be met
C. It is ambiguous: it could mean this coming Friday or the Friday of the following week
D. It is irrelevant information

147. What does defining the scope of a problem do? Understanding

A. States what the solution will cover and what it will not
B. Chooses the programming language
C. Measures the running time of the solution
D. Lists the variable names to be used

148. Which item must every test case based on a requirement include? Understanding

A. The name of the programmer
B. The expected output or result
C. The line numbers of the source code
D. The time of day the test is run

149. What is the purpose of the "5 Whys" technique? Understanding

A. To generate five different solutions to a problem
B. To rank requirements in order of importance
C. To split a project into five equal phases
D. To trace a problem back to its root cause by repeatedly asking why it happened

150. What is an Ishikawa (fishbone) diagram used for? Understanding

A. Showing the order of project tasks over time
B. Grouping the possible causes of a problem into categories
C. Drawing the step-by-step flow of an algorithm
D. Showing relationships between classes

151. In Boehm's well-known phrasing, validation asks "Are we building the right product?" What does verification ask? Understanding

A. Are we building the product right, that is, does it meet its specification?
B. Is the product cheap enough to build?
C. Did the users ask for this product at all?
D. Is the code easy for other programmers to read?

152. What is the difference between a constraint and an assumption? Understanding

A. An assumption is a condition the solution must meet; a constraint is something taken as true without being stated
B. They mean the same thing
C. Assumptions apply only to input size, constraints only to output format
D. A constraint is a condition the solution must meet; an assumption is something taken as true without being stated

153. What is an acceptance criterion for a requirement? Understanding

A. The name of the stakeholder who requested the feature
B. A testable condition the solution must meet for the requirement to count as done
C. The order in which requirements will be built
D. An estimate of the cost of the requirement

154. What does requirements traceability let you do? Understanding

A. Track the location of each user of the system
B. Record every keystroke programmers make
C. Link each requirement to the design parts and tests that cover it, so gaps and the effects of a change are visible
D. Trace the order in which program statements run

155. Why does counting "A or B" use inclusion–exclusion, $|A \cup B| = |A| + |B| - |A \cap B|$? Understanding

A. Items in the overlap must be left out of the answer entirely
B. It is only needed when A and B have no items in common
C. Items in the overlap are counted twice by $|A| + |B|$, so the overlap is subtracted once
D. It counts items that are in neither A nor B

156. A library rule says each member may borrow at most 3 books at a time. Which borrowing counts satisfy the rule? Understanding

A. Exactly 3
B. 3 or more
C. 0, 1, 2 or 3
D. 1, 2 or 3 only

157. For a whole number to be divisible by 6, being divisible by 2 is: Understanding

A. Necessary but not sufficient
B. Sufficient but not necessary
C. Both necessary and sufficient
D. Neither necessary nor sufficient

158. In top-down analysis, what does a structure (hierarchy) chart show? Understanding

A. The order in which steps run, including decisions and loops
B. How the problem breaks into sub-problems, level by level
C. The calendar dates on which each task will be done
D. The tables and keys of a database

159. In MoSCoW prioritisation, what does "Won't have (this time)" mean for a requirement? Understanding

A. It must never be built under any circumstances
B. It is included but given the lowest priority
C. It was found to be wrong and has been deleted
D. It is agreed to be out of scope for this release, though it may be reconsidered later

160. What does a feasibility study check? Understanding

A. Whether the code compiles without errors
B. Whether the problem can be solved within the available technology, budget and time
C. Whether users like the colour scheme
D. Whether every test case has passed

161. Valid inputs are 1 to 100. A tester divides inputs into three classes (below 1, 1 to 100, above 100) and tests one value from each class. What is this technique called? Understanding

A. Boundary value analysis
B. Regression testing
C. Equivalence partitioning
D. Decomposition

162. Using Sommerville's classification of non-functional requirements, "The system must comply with the national data-protection law" is which kind? Understanding
    A. Product requirement
    B. Organisational requirement
    C. External requirement
    D. Functional requirement

163. Using Sommerville's classification, "The system shall be developed in Java because it is the university's standard language" is which kind of requirement? Understanding
    A. Organisational requirement
    B. Product requirement
    C. External requirement
    D. Functional requirement

164. Why is "The system shall store student records in a linked list" a weak requirement? Understanding
    A. It cannot be verified by any test
    B. It is a non-functional performance requirement
    C. It is ambiguous because "student" is undefined
    D. It prescribes a design choice instead of stating what the user needs

165. A 500-litre tank with a red tap was filled at 6 am. It leaks 20 litres per hour. How many hours after filling will 300 litres remain? Application
    A. 15
    B. 25
    C. 10
    D. 4

166. Classify these two requirements: (i) "Users can reset their password by email." (ii) "The system is available 99.9% of the time." Application
    A. (i) functional, (ii) non-functional
    B. (i) non-functional, (ii) functional
    C. Both functional
    D. Both non-functional

167. A function returns how many days remain until a person's next birthday. Which birth date is the most important edge case to test? Application
    A. 1 January
    B. 15 June
    C. 31 December
    D. 29 February

168. What was the car's average speed? (1) It travelled 120 km. (2) The trip took 2 hours. Which statement(s) are enough to answer? Application
    A. Both together are needed, but neither alone
    B. Statement (1) alone is sufficient
    C. Statement (2) alone is sufficient
    D. Even both together are not sufficient

169. How many 3-letter codes can be made from the letters A to E if no letter may repeat within a code? Application
    A. 125
    B. 60
    C. 10
    D. 15

170. Rice costs Rs 180 per 500 g. What do 2.5 kg cost? Application

A. Rs 450
B. Rs 360
C. Rs 1,800
D. Rs 900

171. A problem states n ≤ 5,000 and allows about 10^8 simple operations. Which is the slowest growth rate in this list that still fits? Application

A. $O(n^2)$
B. $O(n^3)$
C. $O(2^n)$
D. O(n log n)

172. Task A takes 3 h. B (2 h) and C (4 h) each start after A. D (1 h) starts after B. E (2 h) starts after both C and D. With unlimited workers, which task could start 1 hour late without delaying the finish? Application

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

173. Electricity is billed at Rs 10 per unit for the first 100 units, Rs 15 per unit for the next 100, and Rs 20 per unit above 200. What is the bill for 250 units? Application

A. Rs 5,000
B. Rs 3,750
C. Rs 2,500
D. Rs 3,500

174. In a class of 40, 25 students play cricket, 20 play football and 8 play both. How many play neither? Application

A. 0
B. 11
C. 5
D. 3

175. Four people, A, B, C and D, sit in a row of four chairs. A and B must not sit next to each other. How many seating orders are possible? Application

A. 24
B. 6
C. 12
D. 18

176. n is a whole number. Is n even? (1) n^2 is even. (2) n + 3 is odd. Which statement(s) are enough to answer? Application

A. Statement (1) alone is sufficient, but (2) is not
B. Statement (2) alone is sufficient, but (1) is not
C. Each statement alone is sufficient
D. Both together are needed, but neither alone

177. How many 30 cm × 30 cm tiles are needed to cover a floor measuring 4.5 m by 3.6 m, with no cutting or waste? Application

A. 180
B. 54
C. 5,400
D. 18

178. How many two-digit numbers greater than 50 have digits that add up to 9? Application

A. 5
B. 4
C. 9
D. 10

179. A grading specification reads: A 80–100, B 60–79, C 40–60, F below 40. Which mark should be raised with the client before coding? Application

A. 40
B. 60
C. 80
D. 39

180. Two buses leave a stop together at 7:00. One returns every 12 minutes, the other every 18 minutes. When do they next leave together? Application

A. 7:30
B. 7:36
C. 7:54
D. 8:12

181. A 4-digit code uses digits 0–9 and may start with 0. How many codes contain at least one 7? Application

A. 4,000
B. 2,916
C. 1,000
D. 3,439

182. x and y are positive whole numbers with $2x + 3y = 30$. How many (x, y) pairs satisfy this? Application

A. 5
B. 6
C. 3
D. 4

183. Is $x > y$? (1) $x^2 > y^2$. (2) $x > 0$. Which statement(s) are enough to answer? Application

A. Statement (1) alone is sufficient
B. Statement (2) alone is sufficient
C. Even both together are not sufficient
D. Both together are needed, but neither alone

184. Ali, Bilal and Chand share a 30 km taxi ride costing Rs 1,800 at a flat rate per km. Chand gets off after 10 km, Bilal after 20 km, and Ali rides all 30 km. The cost of each km is shared equally by the people in the taxi during it. How much does Ali pay? Application

A. Rs 600
B. Rs 1,100
C. Rs 900
D. Rs 1,200

185. How many whole numbers from 1 to 100 are divisible by none of 2, 3 and 5? Application

A. 74
B. 26
C. 29
D. 33

186. A straight 50 m fence needs a post every 5 m, including one at each end. A student answers 10 posts. What went wrong? Analysis
    A. The student should have divided 50 by 10
    B. The student counted the gaps, not the posts; posts = gaps + 1, so 11 are needed
    C. Nothing: 10 posts is correct
    D. The student should have multiplied 50 by 5

187. A user complains: "The app is slow." What is the best first analysis step? Analysis
    A. Ask which action is slow, how slow it is and on which device, so the problem can be measured
    B. Buy a faster server straight away
    C. Rewrite the whole app in another language
    D. Close the complaint, because "slow" is subjective

188. A ticketing system lists: R1 every user must log in with two-factor authentication; R2 guests can buy a ticket without creating an account; R3 tickets are emailed as a PDF; R4 the payment page loads within 3 seconds. Which pair conflicts? Analysis
    A. R1 and R2
    B. R2 and R3
    C. R3 and R4
    D. R1 and R4

189. "One worker paints a wall in 6 days, so two workers will take 3 days." Which unstated assumption does this conclusion rely on? Analysis
    A. The wall is exactly 6 metres long
    B. Each worker works 8 hours a day
    C. Both workers work at the same rate and the job divides between them without getting in each other's way
    D. Paint costs the same every day

190. After decomposition: A needs C's output, B needs A's output, C needs B's output, and D needs nothing. What can you conclude? Analysis
    A. Start with A, because it comes first alphabetically
    B. A, B and C form a cycle, so no valid order exists for them and the decomposition must be revised
    C. Do D first, then A, B and C in that order
    D. Run A, B and C in parallel, since they depend on each other

191. What is the price of one pen? (1) 3 pens and 2 pencils cost Rs 70. (2) 6 pens and 4 pencils cost Rs 140. Which statement(s) are enough to answer? Analysis
    A. Both together are needed, but neither alone
    B. Statement (1) alone is sufficient
    C. Even both together are not sufficient
    D. Statement (2) alone is sufficient

192. A days_in_month(m) function returns 28 when m = 2, 31 when m is odd and 30 when m is even. Which month number exposes a fault? Analysis
    A. 4
    B. 7
    C. 6
    D. 8

193. Trying every ordering of n items takes about n! × n steps. With about 10^8 operations allowed, which input limit makes this brute force acceptable? Analysis

A. n ≤ 10
B. n ≤ 15
C. n ≤ 20
D. n ≤ 100

194. Ali drives 120 km at 60 km/h and returns the same 120 km at 40 km/h. A friend says his average speed was 50 km/h. What is the correct average, and what was the flaw? Analysis

A. 50 km/h; the friend is right to average 60 and 40
B. 48 km/h; averaging the two speeds wrongly assumes he spent equal time at each
C. 48 km/h; the return trip was shorter
D. 52 km/h; the faster leg counts for more

195. Hospital A: mild cases 380/400 survive, severe 30/100. Hospital B: mild 98/100, severe 160/400. Overall, A's survival rate is 82% and B's is 51.6%. Which conclusion is best supported? Analysis

A. A is the better hospital, because its overall survival rate is higher
B. The data contradicts itself, so it must contain an error
C. Severity does not matter, because both hospitals treated 500 patients
D. B does better in each group; A's higher overall rate comes from treating mostly mild cases

196. Choose a team of exactly 3 from P, Q, R and S. Rules: P and Q cannot both be chosen; R can be chosen only if P is; S can be chosen only if Q is. How many valid teams are there? Analysis

A. 1
B. 2
C. 0, because the rules conflict for every team of 3
D. 4

197. An array has up to 10^5 numbers, and up to 10^5 queries each ask for the sum of elements from position l to r. About 10^8 operations are allowed. Which plan meets the constraints? Analysis

A. For each query, loop from l to r and add the elements
B. Build prefix sums once, then answer each query as prefix[r] − prefix[l − 1]
C. Sort the array once, then add the elements from l to r
D. For each query, total the whole array, then subtract the parts outside l to r

198. For three years running, February's sales total is about 10% lower than January's. A manager concludes customers lose interest in February. What should be checked first? Analysis

A. Sales per day, because February has 28 or 29 days against January's 31
B. Whether the totals were added correctly on a calculator
C. Whether February's advertising should be doubled
D. Whether February should be dropped from the report as an outlier

199. P, Q, R and S sit in seats 1 to 4, left to right. Q sits immediately to the right of P. R is not in an end seat. S is not in seat 1. Who sits in seat 4? Analysis

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

200. What is the smallest number of students that guarantees at least 3 of them share a birth month? Analysis

A. 24
B. 36
C. 25
D. 13

Logical Reasoning Fundamentals MCQs for NSCT

201. Which of these is a proposition (a statement that is either true or false)? Understanding
    A. Lahore is the capital of Punjab.
    B. Close the door.
    C. Is it raining?
    D. x + 2 = 5

202. Which kind of reasoning ensures a true conclusion whenever the argument is valid and its premises are true? Understanding
    A. Inductive reasoning
    B. Reasoning by analogy
    C. Abductive reasoning
    D. Deductive reasoning

203. What is the contrapositive of $p \rightarrow q$? Understanding
    A. $q \rightarrow p$
    B. $\neg q \rightarrow \neg p$
    C. $\neg p \rightarrow \neg q$
    D. $p \wedge \neg q$

204. When is the conjunction $p \wedge q$ true? Understanding
    A. Only when p and q are both true
    B. When at least one of p and q is true
    C. When p and q have different truth values
    D. Only when both are false

205. For which values is the conditional $p \rightarrow q$ false? Understanding
    A. p false and q true
    B. p false and q false
    C. p true and q true
    D. p true and q false

206. What is a tautology? Understanding
    A. A statement that is false in every row, such as $p \wedge \neg p$
    B. A statement that is true in exactly half of the rows
    C. Any statement containing the word "if"
    D. A compound statement that is true for every assignment of truth values, such as $p \vee \neg p$

207. When is a deductive argument called valid? Understanding
    A. When all of its premises are actually true
    B. When its conclusion is actually true
    C. When the conclusion cannot be false while all the premises are true
    D. When most people agree with it

208. A deductive argument is sound when: Understanding
    A. It is valid and all of its premises are true
    B. It is valid, whatever its premises
    C. Its conclusion is true
    D. It has exactly two premises

209. In a categorical syllogism, what is the middle term? Understanding

A. The subject of the conclusion
B. The term that appears in both premises but not in the conclusion
C. The predicate of the conclusion
D. The word "all" or "some"

210. When is the biconditional $p \leftrightarrow q$ true? Understanding

A. Only when p is true
B. When exactly one of p and q is true
C. Only when both are false
D. When p and q have the same truth value

211. What is the converse of "If it rains, the ground gets wet"? Understanding

A. If it does not rain, the ground does not get wet
B. If the ground does not get wet, it does not rain
C. If the ground gets wet, it rains
D. It rains and the ground does not get wet

212. "If it is a weekday, the library is open. The library is open. So it is a weekday." Which error does this argument make? Understanding

A. Affirming the consequent
B. Denying the antecedent
C. Modus tollens
D. Begging the question

213. What is the negation of "All students passed the test"? Understanding

A. At least one student did not pass the test
B. No student passed the test
C. Some students passed the test
D. At least one student passed and at least one failed

214. By De Morgan's law, $\neg(p \wedge q)$ is equivalent to: Understanding

A. $\neg p \vee \neg q$
B. $\neg p \wedge \neg q$
C. $p \vee q$
D. $\neg p \vee q$

215. Which of these is an example of inductive reasoning? Understanding

A. All squares have four sides; this shape is a square; so it has four sides.
B. If $x > 5$ then $x > 3$; $x = 7$; so $x > 3$.
C. Every swan I have seen is white, so all swans are white.
D. All primes greater than 2 are odd; 13 is a prime greater than 2; so 13 is odd.

216. When is the exclusive OR $p \oplus q$ true? Understanding

A. When both p and q are true
B. When exactly one of p and q is true
C. When at least one of p and q is true
D. When both are false

217. How many rows does a full truth table for a statement with 4 different propositional variables have? Application

A. 8
B. 4
C. 16
D. 32

218. Let p be true, q false and r true. What are the values of $X = (p \rightarrow q) \wedge (q \rightarrow r)$ and $Y = (p \vee q) \rightarrow r$? Application

A. X is false, Y is true
B. X is true, Y is true
C. X is false, Y is false
D. X is true, Y is false

219. Premises: "Every number divisible by 6 is divisible by 3. 42 is divisible by 6." What follows? Application

A. Every number divisible by 3 is divisible by 6
B. 42 is divisible by 3
C. 42 is divisible by 9
D. 42 is not divisible by 3

220. "All programmers are problem solvers. All problem solvers are logical thinkers." Which conclusion must follow? Application

A. All logical thinkers are programmers.
B. Some logical thinkers are not programmers.
C. No programmer is a logical thinker.
D. All programmers are logical thinkers.

221. "Some students are athletes. All athletes are disciplined." Which conclusion must follow? Application

A. All students are disciplined.
B. All disciplined people are athletes.
C. Some athletes are not students.
D. Some students are disciplined.

222. In the truth table of $p \rightarrow (q \vee r)$, how many of the 8 rows are false? Application

A. 2
B. 1
C. 3
D. 7

223. Which expression is logically equivalent to $p \rightarrow q$? Application

A. $p \vee \neg q$
B. $\neg p \vee q$
C. $\neg p \wedge q$
D. $q \rightarrow p$

224. What is the contrapositive of "If a student is registered, the student has a roll number slip"? Application

A. If a student has a roll number slip, the student is registered
B. If a student has no roll number slip, the student is not registered
C. If a student is not registered, the student has no roll number slip
D. A student is registered and has no roll number slip

225. What is the negation of "If it rains, the match is cancelled"? Application

A. If it does not rain, the match is not cancelled
B. If it rains, the match is not cancelled
C. It does not rain and the match is cancelled
D. It rains and the match is not cancelled

226. On an island, knights always tell the truth and knaves always lie. A says: "We are both knaves." What are A and B? Application

A. A is a knave and B is a knight
B. A is a knight and B is a knave
C. Both are knaves
D. Both are knights

227. What is the truth value of "If 2 + 2 = 5, then the moon is made of cheese"? Application

A. False
B. It is not a proposition
C. It cannot be decided without more information
D. True

228. In the 8-row truth table of $(p \wedge q) \vee (\neg p \wedge r)$, how many rows are true? Application

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

229. P is taller than Q. R is shorter than Q. S is taller than P. Who is the shortest? Application

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

230. "If the server is up, the page loads. The page did not load." What follows? Application

A. The server is up
B. The page will load later
C. The server is not up
D. Nothing at all can be concluded about the server

231. Which argument is valid? Analysis

A. If the code compiles, the tests run. The tests ran. So the code compiled.
B. If the code compiles, the tests run. The code did not compile. So the tests did not run.
C. If the code compiles, the tests run. The tests did not run. So the code did not compile.
D. If the code compiles, the tests run. So if the tests run, the code compiled.

232. "Some engineers are poets. Some poets are singers." Which conclusion must follow? Analysis

A. Some engineers are singers.
B. Some singers are not engineers.
C. Some poets are engineers.
D. All poets are engineers.

233. Which pair of statements are logically equivalent? Analysis

A. $p \rightarrow q$ and $q \rightarrow p$
B. $p \rightarrow q$ and $\neg p \rightarrow \neg q$
C. $p \rightarrow q$ and $\neg q \rightarrow \neg p$
D. $\neg(p \wedge q)$ and $\neg p \wedge \neg q$

234. Which of these is a tautology? Analysis

A. $(p \rightarrow q) \rightarrow (q \rightarrow p)$
B. $((p \rightarrow q) \wedge p) \rightarrow q$
C. $(p \vee q) \rightarrow p$
D. $(p \rightarrow q) \wedge (q \rightarrow p)$

235. Knights always tell the truth; knaves always lie. A says: "B is a knave." B says: "A and C are the same type." What can you conclude for certain? Analysis

A. C is a knight
B. C is a knave
C. A is a knight
D. B is a knave

236. "You cannot sit the test unless you register." Which statement is logically equivalent? Analysis

A. If you sit the test, you registered.
B. If you register, you can sit the test.
C. If you do not sit the test, you did not register.
D. You register if and only if you sit the test.

237. "Passing the test is necessary for getting the certificate." Which statement follows? Analysis

A. Anyone who passes the test gets the certificate.
B. Anyone with the certificate passed the test.
C. Anyone without the certificate failed the test.
D. Passing the test is enough to get the certificate.

238. Five friends P, Q, R, S, T sit in a row of five seats. Q sits at one end. R sits somewhere to the right of P. T sits in the middle seat. S does not sit next to T. Who sits in the second seat from the left? Analysis

A. P
B. R
C. Q
D. It cannot be determined

239. What is the negation of "Every course has at least one student who failed it"? Analysis

A. No course has a student who failed it
B. Every course has at least one student who passed it
C. Some course has a student who did not fail it
D. Some course has no student who failed it

240. Which set of statements cannot all be true at the same time? Analysis

A. $p \rightarrow q$, $q \rightarrow r$, $\neg p$, r
B. $p \vee q$, $\neg p$, $q \rightarrow r$
C. $p \rightarrow q$, $q \rightarrow r$, p, $\neg r$
D. $p \rightarrow q$, $\neg q \rightarrow \neg p$, q

241. What is the inverse of the conditional $p \rightarrow q$? Understanding

A. $\neg p \rightarrow \neg q$
B. $q \rightarrow p$
C. $\neg q \rightarrow \neg p$
D. $p \wedge \neg q$

242. Why is the sentence "x + 3 = 7" not a proposition? Understanding

A. Its truth value depends on x, which is not given
B. It is a command, not a statement
C. It is always false
D. Propositions cannot contain numbers

243. Which connective gives a false result only when both of its components are false? Understanding

A. AND ($\wedge$)
B. Conditional ($\rightarrow$)
C. Biconditional ($\leftrightarrow$)
D. Inclusive OR ($\vee$)

244. A compound statement is true in some rows of its truth table and false in others. What is it called? Understanding

A. A tautology
B. A contradiction
C. A contingency
D. A fallacy

245. Which of these statements is a contradiction (false for every truth value of p)? Understanding

A. $p \vee \neg p$
B. $p \rightarrow p$
C. $p \wedge \neg p$
D. $p \leftrightarrow p$

246. Which rule of inference lets you conclude $p \rightarrow r$ from the premises $p \rightarrow q$ and $q \rightarrow r$? Understanding

A. Modus ponens
B. Disjunctive syllogism
C. Contraposition
D. Hypothetical syllogism

247. From the premises $p \vee q$ and $\neg p$, you conclude q. Which rule of inference is this? Understanding

A. Modus tollens
B. Disjunctive syllogism
C. Addition
D. Simplification

248. Rewrite the conditional $\neg p \rightarrow q$ without using an arrow. Which expression is equivalent to it? Understanding

A. $\neg p \vee q$
B. $p \wedge q$
C. $p \vee q$
D. $\neg p \wedge \neg q$

249. In symbolic logic, how is "p only if q" written? Understanding

A. $q \rightarrow p$
B. $p \leftrightarrow q$
C. $\neg p \rightarrow q$
D. $p \rightarrow q$

250. Which expression is logically equivalent to the biconditional $p \leftrightarrow q$ (read "p if and only if q")? Understanding

A. $(p \rightarrow q) \vee (q \rightarrow p)$
B. $(p \rightarrow q) \wedge (q \rightarrow p)$
C. $p \wedge q$
D. $(p \vee q) \wedge \neg(p \wedge q)$

251. What is the negation of "Some books in the library are expensive"? Understanding

A. No books in the library are expensive
B. Some books in the library are not expensive
C. All books in the library are expensive
D. Some books in the library are cheap

252. A mechanic hears a squeal when the brakes are applied and concludes that the brake pads are most likely worn, since that best explains the sound. Which kind of reasoning is this? Understanding

A. Deductive reasoning
B. Abductive reasoning (inference to the best explanation)
C. Circular reasoning
D. Reasoning by contraposition

253. In predicate logic, which symbol means "there exists at least one"? Understanding

A. ∀
B. ∃
C. ¬
D. ⊕

254. What is the negation of ∀x P(x)? Understanding

A. ∀x ¬P(x)
B. ∃x ¬P(x)
C. ∃x P(x)
D. ¬∃x P(x)

255. Which statement is equivalent to ¬∃x ∀y P(x, y)? Understanding

A. ∃x ∃y ¬P(x, y)
B. ∀x ∀y ¬P(x, y)
C. ∀x ∃y ¬P(x, y)
D. ∃x ∀y ¬P(x, y)

256. How many premises does a standard categorical syllogism have? Understanding

A. One
B. Three
C. Two
D. Any number

257. "All cats are mammals. All dogs are mammals. So all cats are dogs." What is the error in this argument? Understanding

A. Affirming the consequent
B. Denying the antecedent
C. Circular reasoning
D. The fallacy of the undistributed middle

258. "If it is a public holiday, the office is closed. Today is not a public holiday. So the office is open." Which error does this argument make? Understanding

A. Affirming the consequent
B. Modus tollens
C. Denying the antecedent
D. Undistributed middle

259. Which single connective is functionally complete, meaning every truth function can be built from it alone? Understanding

A. ↔ (biconditional)
B. ⊕ (exclusive OR)
C. ∧ (AND)
D. NAND

260. If "No A are B" is true, which statement must also be true? Understanding

A. No B are A
B. All B are A
C. Some A are B
D. All A are B

261. In the traditional square of opposition, how are "All S are P" and "Some S are not P" related? Understanding

A. They are contraries: both can be false but not both true
B. They are subcontraries: both can be true but not both false
C. They are contradictories: exactly one of them is true
D. They are equivalent: they always have the same truth value

262. An argument has a valid form, but one of its premises is false. What can you say about it? Understanding

A. It is sound
B. It is invalid
C. It is valid but not sound
D. Its conclusion must be false

263. What is enough to show that the claim "All prime numbers are odd" is false? Understanding

A. Checking that many odd primes exist
B. Proving the converse is false
C. One counterexample, such as the prime number 2
D. Showing that some odd numbers are not prime

264. Which law of logic states that $p \vee (p \wedge q) \equiv p$? Understanding

A. Distributive law
B. De Morgan's law
C. Idempotent law
D. Absorption law

265. A logical expression contains the variables p, q, r, s and t, and p appears three times. How many rows does its full truth table have? Application

A. 25
B. 15
C. 32
D. 10

266. Let p be false and q be true. Which of these expressions is false? Application

A. $p \leftrightarrow q$
B. $p \rightarrow q$
C. $p \vee q$
D. $\neg p \wedge q$

267. In the 8-row truth table of $p \oplus q \oplus r$, in how many rows is the expression true? Application

A. 2
B. 4
C. 6
D. 1

268. In the 8-row truth table of $(p \vee q) \rightarrow r$, how many rows are false? Application

A. 3
B. 1
C. 4
D. 5

269. "The report is either saved in Google Drive or on the USB stick. It is not on the USB stick." What follows? Application

A. The report is lost
B. The report is in both places
C. The report is saved in Google Drive
D. Nothing follows about where the report is

270. "If Ali studies, he passes. If Ali passes, he gets the scholarship." Which conclusion follows from these two premises alone? Application

A. If Ali studies, he gets the scholarship
B. Ali passes
C. If Ali gets the scholarship, he studied
D. If Ali does not study, he does not get the scholarship

271. "No reptiles are mammals. All snakes are reptiles." Which conclusion must follow? Application

A. Some mammals are snakes
B. All reptiles are snakes
C. Some snakes are mammals
D. No snakes are mammals

272. "All doctors are graduates. Some graduates are unemployed." Which conclusion must follow? Application

A. Nothing definite follows about whether any doctor is unemployed
B. Some doctors are unemployed
C. No doctors are unemployed
D. Some unemployed people are doctors

273. What is the converse of the contrapositive of $p \rightarrow q$? Application

A. $\neg p \rightarrow \neg q$
B. $q \rightarrow p$
C. $p \rightarrow q$
D. $\neg q \rightarrow \neg p$

274. Let g = "I go to the match" and s = "it is sunny". How is "I will go to the match only if it is sunny" written? Application

A. $s \rightarrow g$
B. $g \rightarrow s$
C. $\neg s \rightarrow g$
D. $g \leftrightarrow s$

275. Let S(x) mean "x studies" and P(x) mean "x passes", over the domain of students. Which formula expresses "Not every student who studies passes"? Application

A. $\forall x\,(S(x) \rightarrow \neg P(x))$
B. $\exists x\,(S(x) \rightarrow \neg P(x))$
C. $\neg\exists x\,(S(x) \wedge P(x))$
D. $\exists x\,(S(x) \wedge \neg P(x))$

276. Over the domain {1, 2, 3, 4}, what are the truth values of A = $\forall x\ (x^2 > x)$ and B = $\exists x\ (x^2 = 2x)$? Application

A. A is true and B is false
B. A is false and B is true
C. Both are true
D. Both are false

277. Over the domain of positive integers, which of these statements is true? A: $\forall x\ \exists y\ (y > x)$. B: $\exists y\ \forall x\ (y > x)$. Application

A. Only B is true
B. Only A is true
C. Both are true
D. Neither is true

278. Knights always tell the truth and knaves always lie. A says: "At least one of us is a knave." What are A and B? Application

A. A is a knight and B is a knave
B. Both are knights
C. A is a knave and B is a knight
D. Both are knaves

279. Knights always tell the truth and knaves always lie. A says: "B is a knight." B says: "A and I are of opposite types." What are A and B? Application

A. Both are knights
B. A is a knight and B is a knave
C. A is a knave and B is a knight
D. Both are knaves

280. In a queue, Sara is ahead of Hina, Hina is ahead of Zara, and Ayesha is behind Zara. Nobody else is in the queue. Who is second? Application

A. Sara
B. Hina
C. Zara
D. Ayesha

281. In a class of 35 students, Bilal is 12th from the top. What is his position from the bottom? Application

A. 24th
B. 23rd
C. 22nd
D. 25th

282. Which expression is equivalent to $\neg(p \rightarrow q) \vee q$? Application

A. q
B. $p \wedge \neg q$
C. $p \vee q$
D. A tautology (always true)

283. Which expression is equivalent to $(p \wedge q) \vee (p \wedge \neg q)$? Application

A. q
B. p
C. $p \wedge q$
D. A tautology (always true)

284. Let p be true, q false and r false. Which expression is true? Application

A. $p \rightarrow (q \vee r)$
B. $(p \wedge \neg q) \rightarrow r$
C. $\neg(p \vee r)$
D. $(p \oplus q) \leftrightarrow \neg r$

285. What is the contrapositive of "If a number is not even, it is not divisible by 4"? Application

A. If a number is even, it is divisible by 4
B. If a number is divisible by 4, it is even
C. If a number is not divisible by 4, it is not even
D. If a number is odd, it is not divisible by 4

286. Which of these arguments commits the fallacy of affirming the consequent? Analysis

A. If a number ends in 0, it is divisible by 5. 25 is divisible by 5. So 25 ends in 0.
B. If a number ends in 0, it is divisible by 5. 130 ends in 0. So 130 is divisible by 5.
C. If a number ends in 0, it is divisible by 5. 27 is not divisible by 5. So 27 does not end in 0.
D. Every number divisible by 5 ends in 0 or 5. 45 is divisible by 5 and does not end in 0. So 45 ends in 5.

287. Is this argument valid? Premises: $p \rightarrow q$, $q \rightarrow r$, $\neg r$. Conclusion: $\neg p$. Analysis

A. Invalid: it affirms the consequent
B. Invalid: the conclusion does not mention r
C. Valid: $\neg r$ gives $\neg q$ by modus tollens, and $\neg q$ gives $\neg p$ by modus tollens again
D. Valid only if q is known to be true

288. "If it rains, the match is postponed. Therefore the stadium is empty." Which pair of extra premises makes this argument valid? Analysis

A. It rains, and if the stadium is empty then the match is postponed
B. If the match is postponed then the stadium is empty
C. It rains
D. It rains, and if the match is postponed then the stadium is empty

289. Which pair of statements is NOT logically equivalent? Analysis

A. $p \rightarrow (q \rightarrow r)$ and $(p \wedge q) \rightarrow r$
B. $(p \rightarrow q) \wedge (p \rightarrow r)$ and $p \rightarrow (q \wedge r)$
C. $(p \rightarrow r) \wedge (q \rightarrow r)$ and $(p \vee q) \rightarrow r$
D. $(p \rightarrow q) \rightarrow r$ and $p \rightarrow (q \rightarrow r)$

290. One of A, B and C broke the vase. A says: "B did it." B says: "A is lying." C says: "I did not do it." Exactly one of them tells the truth. Who broke the vase? Analysis

A. C
B. A
C. B
D. It cannot be determined

291. "No managers are lazy. Some employees are lazy." Which conclusion must follow? Analysis

A. Some employees are not managers
B. No employees are managers
C. Some managers are not employees
D. Some employees are managers

292. "All 500 bulbs randomly sampled from a batch of 20,000 worked, so every bulb in the batch works." How is this argument best evaluated? Analysis

A. A deductively valid argument: the conclusion must be true
B. Worthless, because it is an invalid deduction
C. Circular, because it assumes what it proves
D. A strong inductive argument: the conclusion is probable but not guaranteed

293. "The app is reliable because it never fails, and it never fails because it is reliable." What is the main flaw in this reasoning? Analysis

A. It is a hasty generalisation from too few cases
B. It presents a false choice between two options
C. It denies the antecedent of a conditional
D. It is circular: the conclusion is used as its own support

294. "I asked three classmates and all of them found the test hard, so every candidate in Pakistan found it hard." What is the main flaw? Analysis

A. Circular reasoning
B. Denying the antecedent
C. Hasty generalisation from a small, unrepresentative sample
D. The fallacy of the undistributed middle

295. Knights always tell the truth and knaves always lie. A says: "Exactly one of us three is a knave." B says: "All three of us are knaves." What can you conclude for certain? Analysis

A. A and C are both knights
B. C is a knight, but A's type cannot be determined
C. C is a knave
D. All three are knaves

296. Knights always tell the truth and knaves always lie. A says: "If I am a knight, then B is a knight." What are A and B? Analysis

A. Both are knights
B. A is a knight and B is a knave
C. A is a knave and B is a knight
D. Both are knaves

297. "Every programmer who knows Python also knows loops. Hina does not know loops." Which conclusion is justified? Analysis

A. Hina is not a programmer
B. Hina does not know Python
C. Hina is not a programmer who knows Python
D. Hina knows Python but is not a programmer

298. Four books A, B, C and D stand on a shelf in positions 1 to 4 from the left. B is immediately to the right of A. D is not at either end. C is somewhere to the left of A. Which book is in position 3? Analysis

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

299. Each card has a city on one side and a colour on the other. Rule: "If a card shows Lahore on one side, it is blue on the other." The visible faces are Lahore, Karachi, blue and green. Which cards must you turn over to test the rule? Analysis

A. Lahore and blue
B. Lahore and green
C. Lahore only
D. Lahore, blue and green

300. How many of the 8 assignments of truth values to p, q and r make $(p \rightarrow q) \wedge (q \rightarrow r)$ true? Analysis

A. 3
B. 6
C. 2
D. 4

Algorithms & Flow Control MCQs for NSCT

301. In a standard flowchart, which symbol represents a decision such as "Is x > 0?" Understanding

A. Rectangle
B. Diamond
C. Parallelogram
D. Oval

302. Which flowchart symbol is used for "Read marks" or "Print result"? Understanding

A. Rectangle
B. Diamond
C. Parallelogram
D. Small circle

303. What does the oval (rounded rectangle) symbol mark in a flowchart? Understanding

A. A calculation or assignment
B. A loop condition
C. The start or end of the algorithm
D. A call to a stored file

304. Which three control structures are enough to express the flow of any algorithm in structured programming? Understanding

A. Sequence, selection and iteration
B. Input, process and output
C. Sequence, recursion and GOTO
D. Selection, functions and arrays

305. Which property of an algorithm requires it to stop after a finite number of steps for every valid input? Understanding

A. Definiteness
B. Effectiveness
C. Generality
D. Finiteness

306. Which statement best describes pseudocode? Understanding

A. An informal, language-independent description of an algorithm's steps
B. A low-level language that a compiler translates directly to machine code
C. A diagram built from standard shapes joined by arrows
D. A programming language that runs on a virtual machine

307. You must repeat a step exactly 12 times, and the count is known before the loop starts. Which loop fits best? Understanding
    A. A sentinel-controlled WHILE loop
    B. A count-controlled FOR loop
    C. A REPEAT-UNTIL loop that waits for user input
    D. A nested IF-ELSE chain

308. An algorithm step says "add a few more items if needed". Which algorithm property does this step break? Understanding
    A. Finiteness
    B. Definiteness
    C. Output
    D. Input

309. What is the key difference between a pre-test loop (WHILE) and a post-test loop (REPEAT-UNTIL or do-while)? Understanding
    A. A post-test loop always runs its body at least once
    B. A pre-test loop always runs its body at least once
    C. Only a post-test loop can run forever
    D. Only a pre-test loop can use a counter variable

310. In an input loop, what is a sentinel value? Understanding
    A. The first value read, used to set the loop counter
    B. A value that the loop body must never change
    C. The largest value the loop can process
    D. A special value that signals the end of the input

311. In a standard flowchart, how many flow lines does a decision diamond testing "x > 0" normally have? Understanding
    A. One entering and two leaving (Yes and No)
    B. Two entering and one leaving
    C. One entering and one leaving
    D. None entering and two leaving

312. A CASE (switch) statement that picks one of several branches based on a single value is an example of which control structure? Understanding
    A. Iteration
    B. Sequence
    C. Recursion
    D. Multi-way selection

313. A flowchart is too long to draw its flow lines without crossings, so the author uses a small labelled circle at both ends of a break. What is that circle called? Understanding
    A. A terminator
    B. A predefined process
    C. A decision
    D. An on-page connector

314. What does the structured program theorem (Böhm and Jacopini, 1966) state? Understanding
    A. Every program needs at least one GOTO to handle loops
    B. Any computable flow of control can be built from sequence, selection and iteration alone
    C. Any algorithm can be drawn using only sequence and selection
    D. Every loop can be replaced by a fixed number of IF statements

315. What is a loop invariant? Understanding

A. A variable whose value never changes inside the loop
B. The condition that makes the loop stop
C. A condition that is true before the loop starts and remains true after every iteration
D. A loop whose body does not change any variable

316. In the condition IF b ≠ 0 AND a / b > 2, why does the order of the two tests matter in a language with short-circuit evaluation? Understanding

A. AND always evaluates both sides, so the order only affects speed
B. If b ≠ 0 is false, the division is never evaluated, so division by zero is avoided
C. The division must come first because arithmetic has higher precedence than AND
D. Short-circuit evaluation evaluates the right-hand test first

317. After these three assignments run in sequence, what does the pseudocode output? Application

```
x ← 4
y ← x * 2
x ← y + 1
OUTPUT x, y
```

A. 5, 8
B. 9, 9
C. 4, 8
D. 9, 8

318. What grade does this pseudocode output? Application

```
marks ← 67
IF marks >= 80 THEN
    grade ← "A"
ELSE IF marks >= 60 THEN
    grade ← "B"
ELSE
    grade ← "C"
END IF
OUTPUT grade
```

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

319. What is the value of sum after this loop? Application

```
sum ← 0
FOR i ← 1 TO 5
    sum ← sum + i
END FOR
```

A. 10
B. 5
C. 15
D. 21

320. How many values does this loop output? Application

```
FOR i ← 2 TO 10 STEP 2
    OUTPUT i
END FOR
```

A. 4
B. 9
C. 10
D. 5

321. What is the value of count when this loop ends? Application

```
count ← 0
i ← 1
WHILE i < 50 DO
    i ← i * 2
    count ← count + 1
END WHILE
```

A. 6
B. 5
C. 7
D. 25

322. How many times does count ← count + 1 run? Application

```
count ← 0
FOR i ← 1 TO 4
    FOR j ← 1 TO i
        count ← count + 1
    END FOR
END FOR
```

A. 16
B. 10
C. 4
D. 8

323. What does this REPEAT-UNTIL loop output? Application

```
x ← 10
REPEAT
    x ← x - 3
UNTIL x < 0
OUTPUT x
```

A. -2
B. 1
C. -3
D. 0

324. A student tries to swap a and b. What does the pseudocode output? Application

```
a ← 5
b ← 9
a ← b
b ← a
OUTPUT a, b
```

A. 9, 5
B. 5, 9
C. 9, 9
D. 5, 5

325. What does this pseudocode output? (MOD gives the remainder and DIV gives integer division.) Application

```
n ← 4072
s ← 0
WHILE n > 0 DO
    s ← s + (n MOD 10)
    n ← n DIV 10
END WHILE
OUTPUT s
```

A. 13
B. 2704
C. 4
D. 12

326. A flowchart: Start → i ← 1, p ← 1 → process box "p ← p × i; i ← i + 1" → decision "i > 4?" (No goes back to the process box, Yes goes on) → Output p → Stop. What is output? Application

A. 120
B. 24
C. 6
D. 4

327. The array A (1-based) holds [3, 9, 2, 9, 5]. What does this pseudocode output? Application

```
best ← 1
FOR i ← 2 TO 5
    IF A[i] > A[best] THEN
        best ← i
    END IF
END FOR
OUTPUT best
```

A. 4
B. 9
C. 5
D. 2

328. What are the final values of c and d? Application

```
c ← 0
d ← 0
FOR i ← 1 TO 20
    IF i MOD 3 = 0 THEN
        c ← c + 1
    ELSE IF i MOD 5 = 0 THEN
        d ← d + 1
    END IF
END FOR
```

A. c = 6, d = 4
B. c = 5, d = 4
C. c = 6, d = 3
D. c = 7, d = 3

329. What does this Python code print? Application

```
total = 0
for k in range(1, 20):
    if k % 4 == 0:
        continue
    if total > 20:
        break
    total += k
print(total, k)
```

A. 17 7
B. 24 8
C. 24 9
D. 36 9

330. How many times does the loop body run, and what is output? Application

```
a ← 48
b ← 18
WHILE b ≠ 0 DO
    t ← b
    b ← a MOD b
    a ← t
END WHILE
OUTPUT a
```

A. 2 times, output 6
B. 3 times, output 0
C. 4 times, output 12
D. 3 times, output 6

331. This pseudocode should add every element of the 1-based array A[1..n], but for A = [4, 1, 7, 3] it outputs 12 instead of 15. Which single change fixes it? Analysis

```
sum ← 0
i ← 1
WHILE i < n DO
    sum ← sum + A[i]
    i ← i + 1
END WHILE
OUTPUT sum
```

A. Change i < n to i <= n
B. Start i at 0 instead of 1
C. Move i ← i + 1 above the sum line
D. Change i < n to i > n

332. What happens when this pseudocode runs? Analysis

```
i ← 10
WHILE i ≠ 0 DO
    i ← i - 3
END WHILE
```

A. It never terminates, because i jumps from 1 to -2 and never equals 0
B. It stops after 3 iterations
C. It stops after 4 iterations, when i becomes negative
D. It stops immediately, because 10 is not a multiple of 3

333. Which rewrite gives exactly the same result r as the nested version for every integer x? Analysis

```
IF x > 0 THEN
    IF x > 100 THEN
        r ← "big"
    ELSE
        r ← "small"
    END IF
ELSE
    r ← "neg"
END IF
```

A. IF x > 0 THEN small, ELSE IF x > 100 THEN big, ELSE neg
B. IF x >= 100 THEN big, ELSE IF x > 0 THEN small, ELSE neg
C. IF x > 100 THEN big, ELSE IF x >= 0 THEN small, ELSE neg
D. IF x > 100 THEN big, ELSE IF x > 0 THEN small, ELSE neg

334. A flowchart computes an average: Input v → sum ← sum + v → count ← count + 1 → decision "v = -1?" (No loops back to Input; Yes goes on) → Output sum / count. With inputs 4, 8, -1, what is printed and what is the fault? Analysis

A. 6, because the flowchart is correct
B. 11 / 3 ≈ 3.67, because the sentinel is added and counted before it is tested
C. 4, because the loop exits after the first value
D. 12 / 3 = 4, because only count includes the sentinel

335. Which statement is a loop invariant for this loop (true before the first iteration and after every iteration)? Analysis

```
p ← 1
i ← 0
WHILE i < n DO
    p ← p * 2
    i ← i + 1
END WHILE
```

A. p = 2^n
B. p = 2 × i
C. p = 2^i
D. i < n

336. For n = 64, how many times does each loop body run? Analysis

```
// Loop A
i ← 1
WHILE i <= n DO
    i ← i + 1
END WHILE

// Loop B
i ← 1
WHILE i <= n DO
    i ← i * 2
END WHILE
```

A. A: 64, B: 6
B. A: 63, B: 7
C. A: 64, B: 32
D. A: 64, B: 7

337. A linear search scans a 1-based array A[1..n]. When the key is not in the array, what goes wrong, assuming short-circuit AND? Analysis

```
i ← 1
WHILE A[i] ≠ key AND i <= n DO
    i ← i + 1
END WHILE
```

A. Nothing; short-circuit AND checks i <= n first
B. The loop stops one element early and misses A[n]
C. When i reaches n + 1, A[n + 1] is read before i <= n is checked; swap the two tests
D. The loop never starts, because A[1] ≠ key is false

338. Why does a student with 90 marks get grade C from this pseudocode? Analysis

```
IF marks >= 50 THEN
    grade ← "C"
ELSE IF marks >= 70 THEN
    grade ← "B"
ELSE IF marks >= 85 THEN
    grade ← "A"
ELSE
    grade ← "F"
END IF
```

A. ELSE IF runs every branch whose condition is true
B. The final ELSE overrides the earlier branches
C. The widest condition is tested first, so every mark of 50 or more stops at C
D. 90 is outside every range, so the first branch is the default

339. For n = 10, how many more times does fragment P increment x than fragment Q? Analysis

```
// Fragment P
FOR i ← 1 TO n
    FOR j ← 1 TO n
        x ← x + 1

// Fragment Q
FOR i ← 1 TO n
    FOR j ← 1 TO i
        x ← x + 1
```

A. 50
B. 45
C. 55
D. 0

340. Which algorithm property does this pseudocode violate? Analysis

```
count ← 0
WHILE count < 10 DO
    OUTPUT count
END WHILE
```

A. Finiteness
B. Definiteness
C. Input
D. Output

341. In a standard flowchart, which symbol holds a calculation step such as "total ← total + price"? Understanding

A. Diamond
B. Parallelogram
C. Oval
D. Rectangle

342. What do the arrows (flow lines) in a flowchart show? Understanding

A. Where data is stored
B. Which steps are decisions
C. The order in which the steps are carried out
D. Where the algorithm ends

343. In a flowchart, what does a rectangle with an extra vertical line inside each side (the predefined process symbol) represent? Understanding

A. A decision with more than two outcomes
B. Manual input typed by the user
C. Data written to a file
D. A call to a sub-process or module defined elsewhere

344. According to Knuth's definition of an algorithm, how many inputs and outputs must it have? Understanding

A. One or more inputs and zero or more outputs
B. Zero or more inputs and one or more outputs
C. Exactly one input and exactly one output
D. At least one input and at least one output

345. What does the effectiveness property of an algorithm require? Understanding

A. The algorithm is the fastest one known for the problem
B. The algorithm always produces at least one output
C. Every step is written in a real programming language
D. Each step is basic enough to be carried out exactly, in finite time, by a person with pencil and paper

346. What is a trace table (dry run) used for? Understanding

A. Recording the value of each variable, step by step, while running an algorithm by hand
B. Converting pseudocode into machine code
C. Drawing the symbols of a flowchart
D. Measuring how fast a program runs on a computer

347. Inside a loop, the statement count ← count + 1 runs once each time a condition is met. What role does count play? Understanding

A. Accumulator
B. Sentinel
C. Flag
D. Counter

348. A search loop sets found ← TRUE when it meets the target, and the program checks found after the loop. What is found called? Understanding

A. A flag
B. A counter
C. An accumulator
D. A sentinel

349. How does an algorithm differ from a program? Understanding

A. An algorithm is a language-independent sequence of steps; a program implements it in a particular programming language
B. An algorithm must run on a computer; a program need not
C. An algorithm is always longer than the program written from it
D. There is no difference; the two words mean the same thing

350. What is top-down design (stepwise refinement)? Understanding

A. Writing the code first and drawing the flowchart afterwards
B. Testing the last module before the first one
C. Running an algorithm from its last line to its first
D. Breaking a problem into smaller sub-problems and refining each until the steps are simple enough to code

351. A count-controlled loop FOR i ← a TO b uses a step of +1. If a is already greater than b when the loop starts, how many times does its body run? Understanding

A. Once
B. Zero times
C. b − a times
D. It loops forever

352. A program must ask for a PIN at least once and keep asking until the PIN is correct. Which loop fits best? Understanding

A. A post-test loop such as REPEAT ... UNTIL
B. A count-controlled FOR loop
C. A single IF statement
D. A CASE statement

353. An outer loop runs m times and, on every pass, an inner loop runs n times. How many times does the inner body run in total? Understanding

A. m + n
B. m
C. n
D. m × n

354. What must be true for a WHILE loop to stop eventually? Understanding

A. The condition must be true when the loop starts
B. The body must contain an OUTPUT statement
C. The loop must be inside an IF statement
D. The body must change a variable used in the condition so that the condition becomes false

355. To prove that a loop terminates, what do you usually show? Understanding

A. A loop invariant holds before and after every iteration
B. The loop condition is true when the loop starts
C. Some non-negative integer quantity strictly decreases on every iteration
D. The loop body contains no IF statements

356. Using the usual operator precedence (NOT before AND before OR), how is NOT a OR b AND c evaluated? Understanding

A. NOT (a OR b) AND c
B. (NOT a) OR (b AND c)
C. ((NOT a) OR b) AND c
D. NOT (a OR (b AND c))

357. When is a CASE (switch) statement a better choice than a chain of IF ... ELSE IF tests? Understanding

A. When each branch tests a different range such as marks >= 80
B. When the conditions combine several variables with AND
C. When one variable is compared against several fixed values
D. When a step must repeat until a condition is true

358. What does a break statement do when it runs inside a loop? Understanding

A. Skips to the next iteration of the loop
B. Restarts the loop from its first iteration
C. Ends the loop at once, and execution continues after the loop
D. Ends the whole program

359. A break statement runs inside the inner loop of two nested loops. What happens? Understanding

A. Both loops end at once
B. The outer loop ends but the inner loop continues
C. Only the inner loop ends; the outer loop continues with its next iteration
D. The inner loop restarts from its first value

360. When is an algorithm called deterministic? Understanding

A. When the same input always produces the same sequence of steps and the same output
B. When it always finishes in less than one second
C. When it uses no loops
D. When it can be drawn as a flowchart

361. Why does structured programming discourage the unrestricted GOTO statement, as Dijkstra argued in 1968? Understanding

A. GOTO statements run slower than loops
B. GOTO cannot be drawn in a flowchart
C. Jumps to arbitrary points make the flow of control hard to follow, test and verify
D. Compilers cannot translate GOTO statements

362. In a flowchart loop, the decision diamond comes before the process boxes it repeats. Which kind of loop is this? Understanding

A. A pre-test loop such as WHILE
B. A post-test loop such as REPEAT ... UNTIL
C. A selection, not a loop
D. An infinite loop

363. Consider IF a THEN IF b THEN x ← 1 ELSE x ← 2, written without END IF markers. Under the usual rule, which IF does the ELSE belong to? Understanding

A. The outer IF, which is IF a
B. The nearest unmatched IF, which is IF b
C. Both IFs at once
D. Neither, because the statement is invalid

364. Which property do sequence, selection and iteration blocks share in structured programming? Understanding

A. Each must contain at least one loop
B. Each must end with an OUTPUT statement
C. Each has exactly one entry point and one exit point
D. Each can only appear once in an algorithm

365. What does this pseudocode output? Application

```
a ← 1
b ← 2
c ← 3
t ← a
a ← b
b ← c
c ← t
OUTPUT a, b, c
```

A. 2 3 1
B. 1 2 3
C. 3 1 2
D. 2 3 2

366. What does this pseudocode output when n = 14? Application

```
CASE n MOD 4 OF
    0: OUTPUT "A"
    1, 3: OUTPUT "B"
    2: OUTPUT "C"
    OTHERWISE: OUTPUT "D"
END CASE
```

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

367. What is the last value this loop outputs? Application

```
FOR i ← 20 TO 1 STEP -4
    OUTPUT i
END FOR
```

A. 1
B. 0
C. 8
D. 4

368. What does this pseudocode output? Application

```
p ← 1
FOR i ← 1 TO 7 STEP 2
    p ← p * i
END FOR
OUTPUT p
```

A. 5040
B. 105
C. 16
D. 945

369. What does this pseudocode output? Application

```
x ← 8
REPEAT
    x ← x + 2
UNTIL x > 5
OUTPUT x
```

A. 8
B. 6
C. 10
D. 12

370. What does this pseudocode output? Application

```
n ← 1230
r ← 0
WHILE n > 0 DO
    r ← r * 10 + n MOD 10
    n ← n DIV 10
END WHILE
OUTPUT r
```

A. 3210
B. 321
C. 1230
D. 32

371. What does this pseudocode output? Application

```
a ← 0
b ← 1
FOR i ← 1 TO 6
    t ← a + b
    a ← b
    b ← t
END FOR
OUTPUT a
```

A. 8
B. 5
C. 13
D. 21

372. What does this pseudocode output? Application

```
n ← 91
isPrime ← TRUE
d ← 2
WHILE d * d <= n AND isPrime DO
    IF n MOD d = 0 THEN
        isPrime ← FALSE
    ELSE
        d ← d + 1
    END IF
END WHILE
OUTPUT isPrime, d
```

A. TRUE 10
B. FALSE 13
C. FALSE 7
D. TRUE 9

373. What does this pseudocode output when x = 15? Application

```
IF x MOD 3 = 0 AND NOT (x MOD 5 = 0) THEN
    OUTPUT "A"
ELSE IF x MOD 3 = 0 OR x MOD 5 = 0 THEN
    OUTPUT "B"
ELSE
    OUTPUT "C"
END IF
```

A. A
B. C
C. A and then B
D. B

374. How many times does count ← count + 1 run? Application

```
count ← 0
FOR i ← 1 TO 5
    FOR j ← i TO 5
        count ← count + 1
    END FOR
END FOR
```

A. 10
B. 15
C. 25
D. 5

375. What does this pseudocode output? Application

```
n ← 6
steps ← 0
WHILE n ≠ 1 DO
    IF n MOD 2 = 0 THEN
        n ← n DIV 2
    ELSE
        n ← 3 * n + 1
    END IF
    steps ← steps + 1
END WHILE
OUTPUT steps
```

A. 7
B. 9
C. 16
D. 8

376. What does this Python code print? Application

```
for i in range(10, 0, -3):
    print(i, end=" ")
```

A. 10 7 4 1 -2
B. 10 7 4
C. 10 7 4 1
D. 7 4 1

377. What does this JavaScript code print? Application

```
let s = 0;
for (let i = 1; i <= 10; i++) {
  if (i % 2 !== 0) continue;
  if (i > 8) break;
  s += i;
}
console.log(s);
```

A. 20
B. 30
C. 25
D. 12

378. What does this Python code print? Application

```
count = 0
for i in range(1, 5):
    for j in range(1, 5):
        if i * j > 6:
            break
        count += 1
print(count)
```

A. 16
B. 10
C. 8
D. 4

379. The 1-based array A holds [5, 1, 4, 2, 8]. What does A hold after this single pass? Application

```
FOR i ← 1 TO 4
    IF A[i] > A[i + 1] THEN
        temp ← A[i]
        A[i] ← A[i + 1]
        A[i + 1] ← temp
    END IF
END FOR
```

A. [1, 2, 4, 5, 8]
B. [1, 5, 4, 2, 8]
C. [1, 4, 2, 5, 8]
D. [5, 1, 4, 2, 8]

380. The 1-based array A holds [7, 3, 9, 3, 5]. What does this pseudocode output? Application

```
pos ← 0
FOR i ← 1 TO 5
    IF A[i] = 3 THEN
        pos ← i
    END IF
END FOR
OUTPUT pos
```

A. 2
B. 0
C. 4
D. 5

381. A flowchart reads a and b, then loops: decision "a = b?" (Yes → Output a → Stop). On No, a second decision "a > b?" sends Yes to "a ← a − b" and No to "b ← b − a", and both flow back to "a = b?". What is output for a = 12, b = 18? Application

A. 3
B. 12
C. 6
D. 2

382. What is the value of count when this loop ends? Application

```
i ← 100
count ← 0
WHILE i > 1 DO
    i ← i DIV 3
    count ← count + 1
END WHILE
```

A. 3
B. 4
C. 5
D. 33

383. What does this pseudocode output? Application

```
s ← 0
FOR i ← 1 TO 4
    FOR j ← i TO 4 STEP i
        s ← s + j
    END FOR
END FOR
OUTPUT s
```

A. 23
B. 30
C. 10
D. 16

384. What does this pseudocode output? Application

```
x ← 3
n ← 5
r ← 1
WHILE n > 0 DO
    IF n MOD 2 = 1 THEN
        r ← r * x
    END IF
    x ← x * x
    n ← n DIV 2
END WHILE
OUTPUT r
```

A. 15
B. 81
C. 729
D. 243

385. A video lasts t = 137 minutes. What does this pseudocode output? Application

```
h ← t DIV 60
m ← t MOD 60
OUTPUT h, m
```

A. 2 37
B. 2 17
C. 137 2
D. 2.28 17

386. This pseudocode should output the largest value in the 1-based array A[1..n]. For A = [-4, -9, -2] it outputs 0. Which change fixes it for every array? Analysis

```
max ← 0
FOR i ← 1 TO n
    IF A[i] > max THEN
        max ← A[i]
    END IF
END FOR
OUTPUT max
```

A. Change > to >=
B. Initialise max ← A[1] instead of 0
C. Initialise max ← -1 instead of 0
D. Change max ← A[i] to max ← i

387. Which WHILE-based fragment behaves exactly like REPEAT body UNTIL c for every condition c? Analysis

A. WHILE NOT c DO body END WHILE
B. body, then WHILE NOT c DO body END WHILE
C. WHILE c DO body END WHILE
D. body, then WHILE c DO body END WHILE

388. Which WHILE loop does the same work as FOR i ← 1 TO n STEP 2 ... END FOR for every positive n? Analysis

A. i ← 1, then WHILE i < n DO (body; i ← i + 2) END WHILE
B. i ← 1, then WHILE i <= n DO (body; i ← i + 2) END WHILE
C. i ← 1, then WHILE i <= n DO (i ← i + 2; body) END WHILE
D. i ← 1, then WHILE i <= n DO (body; i ← i + 1) END WHILE

389. Which output can this pseudocode never produce, whatever integer x is? Analysis

```
IF x < 5 AND x > 10 THEN
    OUTPUT "P"
ELSE IF x >= 5 THEN
    OUTPUT "Q"
ELSE
    OUTPUT "R"
END IF
```

A. P
B. Q
C. R
D. It can produce all three

390. Which statement is a loop invariant of this loop (true before the first iteration and after every iteration)? Analysis

```
s ← 0
i ← 0
WHILE i < n DO
    i ← i + 1
    s ← s + i
END WHILE
```

A. s = n(n + 1) / 2
B. s = i(i + 1) / 2
C. s = i(i − 1) / 2
D. i < n

391. This pseudocode should output the even numbers from 1 to 10. What actually happens? Analysis

```
i ← 1
WHILE i <= 10 DO
    IF i MOD 2 = 0 THEN
        OUTPUT i
        i ← i + 1
    END IF
END WHILE
```

A. It loops forever without outputting anything, because i stays 1
B. It outputs 2 4 6 8 10
C. It outputs 1 3 5 7 9
D. It outputs 2 and then stops

392. For n = 1000, how many times does the loop body run? Analysis

```
i ← 1
count ← 0
WHILE i * i <= n DO
    count ← count + 1
    i ← i + 1
END WHILE
```

A. 32
B. 30
C. 31
D. 500

393. A shop gives a discount on purchases of Rs 5,000 or more. The code tests IF amount > 5000 THEN apply discount. Which single test input exposes the bug? Analysis

A. 5000
B. 4999
C. 6000
D. 0

394. Which set of (a, b) test inputs makes every one of X, Y and Z run at least once? Analysis

```
IF a > 0 THEN
    IF b > 0 THEN
        X
    ELSE
        Y
    END IF
ELSE
    Z
END IF
```

A. (1, 1), (-1, -1), (-1, 1)
B. (1, 1), (1, -1)
C. (-1, 1), (-1, -1), (1, 1)
D. (1, 1), (1, -1), (-1, 1)

395. Which fragment correctly swaps the integer values of a and b (ignoring overflow)? Analysis

A. a ← a − b; b ← a + b; a ← a − b
B. a ← a + b; b ← a − b; a ← a − b
C. t ← a; a ← b; b ← a
D. b ← a; t ← b; a ← t

396. Which fragment outputs exactly 1 2 3 4 when n = 4? Analysis

A. i ← 1; WHILE i < n DO (OUTPUT i; i ← i + 1) END WHILE
B. i ← 0; WHILE i <= n DO (i ← i + 1; OUTPUT i) END WHILE
C. i ← 0; WHILE i < n DO (i ← i + 1; OUTPUT i) END WHILE
D. i ← 1; REPEAT (i ← i + 1; OUTPUT i) UNTIL i = n

397. For n = 8, how many times does count ← count + 1 run in total? Analysis

```
count ← 0
FOR i ← 1 TO n
    j ← 1
    WHILE j < i DO
        j ← j * 2
        count ← count + 1
    END WHILE
END FOR
```

A. 24
B. 12
C. 28
D. 17

398. What does this pseudocode compute for a positive integer n? Analysis

```
s ← 0
WHILE n > 0 DO
    s ← s + n MOD 2
    n ← n DIV 2
END WHILE
OUTPUT s
```

A. The number of binary digits of n
B. Whether n is even or odd
C. The number of even decimal digits of n
D. The number of 1s in the binary form of n

399. A student's flowchart has a decision diamond "n > 0?" whose Yes arrow goes to "Print n", but there is no No arrow. What is wrong? Analysis

A. The decision has no path for n <= 0, so the algorithm's behaviour is undefined for those inputs
B. A decision diamond should have only one exit, so the chart is correct
C. "Print n" must be drawn in a rectangle
D. Decisions cannot test a number against 0

400. For a 1-based array A[1..n], what does this pseudocode detect? Analysis

```
found ← FALSE
FOR i ← 1 TO n - 1
    FOR j ← i + 1 TO n
        IF A[i] = A[j] THEN
            found ← TRUE
        END IF
    END FOR
END FOR
```

A. Whether any value appears more than once anywhere in the array
B. Whether the array is sorted
C. Whether two neighbouring elements are equal
D. How many distinct values the array holds

Data Representation & Abstraction MCQs for NSCT

401. How many distinct digit symbols does the hexadecimal (base-16) system use? Understanding

A. 15 (1-9 and A-F)
B. 10 (0-9)
C. 16 (0-9 and A-F)
D. 8 (0-7)

402. How many bits make up one byte? Understanding

A. 8
B. 4
C. 16
D. 10

403. In computational thinking, what does abstraction mean? Understanding

A. Hiding or removing details that are not needed, to focus on what matters for the problem
B. Breaking a problem into smaller sub-problems
C. Finding similarities between different problems
D. Writing the exact step-by-step instructions of a solution

404. Why do programmers often write binary values in hexadecimal? Understanding

A. Each hex digit maps to exactly 4 bits, so long binary strings become short and easy to read
B. Computers store data internally in base 16
C. Hexadecimal uses less memory than binary
D. Hexadecimal avoids the need for negative numbers

405. How many distinct bit patterns can n bits represent? Understanding

A. 2n
B. n^2
C. 2^n - 1
D. 2^n

406. Why do almost all modern computers store signed integers in two's complement? Understanding

A. It can store larger positive numbers than unsigned binary in the same bits
B. It makes negative numbers easier for humans to read
C. It stores the sign in a separate byte so no bits are wasted
D. It has a single zero and lets the same adder circuit handle both addition and subtraction

407. Standard ASCII uses 7 bits per character. How many characters can it represent? Understanding

A. 256
B. 128
C. 127
D. 64

408. Which statement about UTF-8 is correct? Understanding

A. It always uses exactly 2 bytes per character
B. It can only encode English letters and digits
C. It uses 7 bits per character like ASCII
D. It uses 1 to 4 bytes per character, and plain ASCII text is unchanged in UTF-8

409. What defines an abstract data type (ADT) such as a Stack? Understanding

A. The programming language used to write it
B. The operations it supports and how they behave, independent of how it is implemented
C. The exact memory layout of its elements
D. Whether it is stored in an array or a linked list

410. Which list goes from the highest level of abstraction to the lowest? Understanding

A. Machine code → assembly language → high-level language → logic gates
B. Logic gates → machine code → assembly language → high-level language
C. High-level language → assembly language → machine code → logic gates
D. Assembly language → high-level language → logic gates → machine code

411. Each octal (base-8) digit corresponds to how many binary bits? Understanding

A. 4
B. 3
C. 2
D. 8

412. Which representation of signed integers has two different bit patterns for zero (+0 and -0)? Understanding

A. Two's complement
B. Unsigned binary
C. Sign-magnitude
D. Excess-127 (biased)

413. How are the 32 bits of an IEEE 754 single-precision floating-point number divided? Understanding

A. 1 sign bit, 7 exponent bits, 24 fraction bits
B. 1 sign bit, 11 exponent bits, 20 fraction bits
C. 1 sign bit, 8 exponent bits, 23 fraction bits
D. 8 sign bits, 8 exponent bits, 16 fraction bits

414. Why can 0.1 (decimal) not be stored exactly in binary floating point? Understanding

A. Floating point can only store whole numbers exactly
B. 0.1 is too small for the exponent field
C. Binary has no way to represent any fraction
D. 0.1 is 1/10, and a denominator with a factor of 5 gives a never-ending binary fraction

415. A student calls sqrt(x) without knowing which method the library uses to compute the square root. This is an example of: Understanding

A. Data encryption
B. Procedural abstraction
C. Decomposition
D. Type conversion

416. When adding two signed numbers in two's complement, when does overflow occur? Understanding

A. When both operands have the same sign and the result has the opposite sign
B. Whenever there is a carry out of the leftmost bit
C. Whenever the two operands have different signs
D. Whenever the result is negative

417. What is the binary number 1011 in decimal? Application

A. 11
B. 13
C. 9
D. 1011

418. What is decimal 25 in binary? Application

A. 10011
B. 11010
C. 10101
D. 11001

419. What is hexadecimal 2F in decimal? Application

A. 32
B. 215
C. 62
D. 47

420. What is 11010110 (binary) in hexadecimal? Application

A. 6D
B. D6
C. C6
D. D5

421. What is decimal 157 in octal? Application

A. 532
B. 175
C. 237
D. 235

422. What is -20 in 8-bit two's complement? Application

A. 10010100
B. 11101011
C. 00010100
D. 11101100

423. The 8-bit pattern 10110011 is a two's complement integer. What is its decimal value? Application

A. 179
B. -51
C. -76
D. -77

424. What range of integers can 8-bit two's complement represent? Application

A. -128 to 127
B. -127 to 127
C. -128 to 128
D. 0 to 255

425. What is 01101101 + 00110110 in binary (8-bit unsigned)? Application

A. 10010011
B. 10100011
C. 10100111
D. 01100011

426. What is the minimum number of bits needed to give each of 300 students a unique ID? Application

A. 8
B. 10
C. 9
D. 300

427. An uncompressed image is 100 × 100 pixels with 24-bit colour. How many bytes does the pixel data need? Application

A. 30,000 bytes
B. 240,000 bytes
C. 10,000 bytes
D. 2,400 bytes

428. In 4-bit two's complement, what is the result of 0111 + 0001, and what does it mean? Application

A. 1000, which reads as -8: the true sum 8 overflows the range -8 to 7
B. 1000, which reads as +8: the addition is correct
C. 0000, because the carry is discarded
D. 10000, because a fifth bit is added

429. What is the binary number 101.101 in decimal? Application

A. 5.5
B. 5.101
C. 5.625
D. 5.25

430. What is 12 XOR 10 (bitwise exclusive OR)? Application

A. 8
B. 6
C. 14
D. 2

431. You must model cities and the roads between them, with a distance on each road, to find shortest routes. Which representation fits best? Analysis

A. A one-dimensional array of city names
B. A weighted graph: cities as vertices, roads as edges labelled with distances
C. A stack of cities in the order they were added
D. A tree with the capital city as the root

432. An 8-bit unsigned counter holds 250. It is incremented 10 times, and overflow is ignored. What value does it hold? Analysis

    A. 260
    B. 255
    C. 4
    D. 5

433. Which of these values is the largest? Analysis

    A. 177 (octal)
    B. 1111111 (binary)
    C. 80 (hexadecimal)
    D. 126 (decimal)

434. A metro journey planner must find the route with the fewest stops between two stations. Which detail can safely be left out of its model? Analysis

    A. The exact track length between stations
    B. Which pairs of stations are directly connected
    C. The starting station
    D. The destination station

435. What does this Python code print (line 1, then line 2)? Analysis

```
print(0.1 + 0.2 == 0.3)
print(round(0.1 + 0.2, 2) == 0.3)
```

    A. False, then True
    B. True, then True
    C. False, then False
    D. True, then False

436. The 4-bit two's complement value 1010 must be widened to 8 bits without changing its value. What is the 8-bit pattern? Analysis

    A. 00001010
    B. 10001010
    C. 11111010
    D. 11110101

437. A program uses a Stack ADT only through push, pop and isEmpty. The stack's implementation is changed from an array to a linked list with the same operations and behaviour. What must change in the program? Analysis

    A. Every call to push and pop must be rewritten
    B. Nothing; the program depends only on the interface
    C. The program must now manage the list's pointers itself
    D. The program must be changed to read elements by index

438. Bytes are sent with even parity (the total number of 1s in each 8-bit byte must be even). Which received byte must contain an error? Analysis

    A. 10010110
    B. 11010110
    C. 11110000
    D. 00000000

439. IEEE 754 single precision has 24 bits of significand precision. What is the smallest positive integer it cannot store exactly? Analysis

A. 8,388,609 (2^23 + 1)
B. 16,777,216 (2^24)
C. 4,294,967,296 (2^32)
D. 16,777,217 (2^24 + 1)

440. In which base is the sum 13 + 5 = 20 correct? Analysis

A. 6
B. 9
C. 8
D. 10

441. What is a group of 4 bits called? Understanding

A. A nibble
B. A byte
C. A word
D. An octet

442. In a binary number, which bit has the greatest place value? Understanding

A. The rightmost bit, called the least significant bit (LSB)
B. The middle bit, because it splits the number in half
C. The leftmost bit, called the most significant bit (MSB)
D. Whichever bit is the parity bit

443. Which statement describes lossless compression? Understanding

A. It permanently discards detail that people are unlikely to notice
B. It always reduces the file to less than half its original size
C. It can only be used on photographs, not on text
D. The original data can be rebuilt exactly, bit for bit, from the compressed file

444. Which of these file formats uses lossy compression? Understanding

A. PNG
B. JPEG
C. ZIP
D. FLAC

445. What does a computer's endianness describe? Understanding

A. Whether integers are stored as signed or unsigned
B. The order of the bits inside a single character
C. The order in which the bytes of a multi-byte value are stored in memory
D. Which bit of a byte is used as the parity bit

446. What is the main purpose of adding a parity bit to a byte? Understanding

A. To detect that a single bit was changed in storage or transmission
B. To correct any number of flipped bits automatically
C. To encrypt the byte so others cannot read it
D. To compress the byte into fewer bits

447. How is -x formed from x in one's complement representation? Understanding

A. Invert every bit of x and then add 1
B. Set only the leftmost bit of x to 1
C. Shift x one place to the left
D. Invert every bit of x

448. In IEEE 754 single precision, what bias is added to the true exponent before it is stored in the 8-bit exponent field? Understanding

A. 128
B. 127
C. 1023
D. 255

449. How are the 64 bits of an IEEE 754 double-precision number divided? Understanding

A. 1 sign bit, 11 exponent bits, 52 fraction bits
B. 1 sign bit, 8 exponent bits, 55 fraction bits
C. 1 sign bit, 15 exponent bits, 48 fraction bits
D. 2 sign bits, 10 exponent bits, 52 fraction bits

450. In IEEE 754, what does a value with every exponent bit set to 1 and a non-zero fraction represent? Understanding

A. NaN (Not a Number)
B. Positive or negative infinity
C. Zero
D. The largest finite number

451. What is Binary-Coded Decimal (BCD)? Understanding

A. The whole decimal number is converted to one pure binary value
B. Each decimal digit is stored separately in its own 4-bit group
C. Each digit is stored as its 8-bit ASCII character code
D. A base-12 system used for clocks

452. What is the defining property of a Gray code sequence? Understanding

A. Each value has an even number of 1s
B. Each value differs from the next in exactly one bit
C. Each value is the bitwise inverse of the one before it
D. Its values count upward exactly like ordinary binary

453. What does a dictionary (map) abstract data type let you do? Understanding

A. Keep items strictly in last-in, first-out order
B. Store only numbers in sorted order
C. Store values and look each one up by a unique key
D. Access items only by their position number

454. Why do banking systems usually store money as a whole number of paisa (or cents) rather than as a floating-point number of rupees? Understanding

A. Floating-point numbers cannot be negative
B. Integers can hold much larger values than any floating-point type
C. Floating-point numbers take less memory, so they are reserved for scientific code
D. Integers are exact, while binary floating point cannot store values such as 0.10 exactly

455. In the three-schema architecture of a database, which level describes how the data is physically stored, such as files and indexes? Understanding

A. The internal level
B. The external level
C. The conceptual level
D. The presentation level

456. How many bytes are in one kibibyte (KiB)? Understanding

A. 1,000
B. 8,192
C. 1,048,576
D. 1,024

457. If no bits are lost off the left end, what does shifting an unsigned binary number one place to the left do to its value? Understanding

A. It halves the value
B. It adds 1 to the value
C. It doubles the value
D. It changes the value's sign

458. How does an arithmetic right shift differ from a logical right shift? Understanding

A. An arithmetic shift fills the left position with 1 and a logical shift with 0, whatever the sign
B. An arithmetic shift moves bits two places, a logical shift one place
C. There is no difference for any value; the names are interchangeable
D. An arithmetic shift copies the sign bit into the vacated left position, while a logical shift fills it with 0

459. What is the relationship between Unicode and UTF-8? Understanding

A. They are two names for the same 8-bit character table
B. UTF-8 is the older standard that Unicode replaced
C. Unicode covers only English, while UTF-8 adds other scripts
D. Unicode assigns a number (code point) to each character, and UTF-8 is one way of encoding those numbers as bytes

460. Compared with a fixed-point format of the same bit width, what is true of a floating-point format? Understanding

A. It covers a much wider range of values, but the gap between neighbouring representable values grows as the magnitude grows
B. It stores every decimal fraction exactly
C. It has the same spacing between all representable values
D. It can only represent whole numbers

461. In standard 24-bit RGB colour, how many bits are used for each of the red, green and blue channels? Understanding

A. 24
B. 3
C. 16
D. 8

462. A chess program stores the board as an 8 × 8 grid of piece codes and ignores the board's material and colour. What is this an example of? Understanding

A. Decomposition: breaking the program into smaller functions
B. Abstraction: keeping only the details the problem needs
C. Encryption: hiding the board from other players
D. Compression: reducing the file size of the board

463. What is the Hamming distance between two bit strings of equal length? Understanding

A. The difference between their decimal values
B. The number of positions in which their bits differ
C. The total number of 1s in both strings
D. The position of the first bit where they differ

464. When two signed numbers are added in two's complement, how is signed overflow detected? Understanding

A. Any carry out of the most significant bit means overflow
B. The result's most significant bit is 1
C. The result is an even number
D. The carry into the most significant bit differs from the carry out of it

465. What is hexadecimal 3C in 8-bit binary? Application

A. 00111100
B. 00110011
C. 00111010
D. 11000011

466. What is decimal 100 in hexadecimal? Application

A. 64
B. 100
C. A0
D. 46

467. What is octal 17 in decimal? Application

A. 15
B. 17
C. 13
D. 23

468. In ASCII, uppercase 'A' has code 65 and lowercase letters start 32 codes later. What is the code for lowercase 'a'? Application

A. 66
B. 33
C. 97
D. 129

469. What is decimal 59 in Binary-Coded Decimal (BCD)? Application

A. 0011 1011
B. 0101 1001
C. 0101 1010
D. 1001 0101

470. How many hexadecimal digits are needed to write any 64-bit value? Application

A. 8
B. 16
C. 32
D. 64

471. Run-length encoding replaces each run of a repeated character with its count followed by the character. What is the encoding of AAAABBBCCD? Application

A. A4B3C2D1
B. 4A3B2C
C. 4A3B2C1D
D. AAAA3B2CD

472. What does this Python code print? Application

```
print(ord('7') - ord('0'), chr(ord('C') + 32))
```

A. 55 c
B. 7 c
C. 7 C
D. 55 D

473. A little-endian machine stores the 32-bit value 0x12345678 starting at address 100. Which byte is at address 100? Application

A. 0x12
B. 0x78
C. 0x34
D. 0x87

474. What is -5 in 8-bit one's complement? Application

A. 11111010
B. 11111011
C. 10000101
D. 00000101

475. What is -37 in 8-bit sign-magnitude representation? Application

A. 11011011
B. 11011010
C. 10100101
D. 00100101

476. What is 10101100 AND 00001111? Application

A. 10101111
B. 10100011
C. 00000011
D. 00001100

477. What does this Python code print? Application

```
print(13 << 2, 13 >> 1)
```

A. 52 6
B. 26 6
C. 52 7
D. 15 6

478. What is decimal 0.375 in binary? Application

A. 0.101
B. 0.011
C. 0.0011
D. 0.375

479. An uncompressed mono audio clip lasts 10 seconds, sampled 44,100 times per second at 16 bits per sample. How many bytes does it need? Application

A. 7,056,000 bytes
B. 441,000 bytes
C. 1,764,000 bytes
D. 882,000 bytes

480. What is 10010110 - 00111011 in 8-bit unsigned binary? Application

A. 01011101
B. 10101101
C. 01011011
D. 01101011

481. What is the fewest bits needed to store every integer from -100 to 100 in two's complement? Application

A. 7
B. 9
C. 201
D. 8

482. A web page uses the colour #FF8000 (RRGGBB in hexadecimal). What are its red, green and blue values? Application

A. Red 255, green 80, blue 0
B. Red 0, green 128, blue 255
C. Red 255, green 128, blue 0
D. Red 128, green 255, blue 0

483. The 32 bits of an IEEE 754 single-precision value are 0x40400000. What number is stored? Application

A. 1.5
B. 4.0
C. 3.0
D. 2.5

484. To convert binary to Gray code, keep the first bit and then XOR each bit with the bit to its left. What is the Gray code for binary 0110? Application

A. 0101
B. 0011
C. 1001
D. 0111

485. What does this Python code print? (Python's >> on integers is an arithmetic shift.) Application

```
print(-7 >> 1, 7 >> 1)
```

A. -3 3
B. -4 4
C. -3 4
D. -4 3

486. A company wants to model its staff so that every employee except the CEO has exactly one manager, and a manager can have many staff. Which structure fits best? Analysis

A. A stack of employees
B. A tree, with the CEO as the root
C. An undirected graph with cycles allowed
D. A flat list sorted by salary

487. Which of these does NOT equal decimal 45? Analysis

A. 101101 (binary)
B. 2D (hexadecimal)
C. 45 (hexadecimal)
D. 55 (octal)

488. A form stores Pakistani mobile numbers such as 03001234567. Which data type is the best choice for this field? Analysis

A. A 32-bit integer, because the value contains only digits
B. A floating-point number, so very long numbers fit
C. A string, because the leading zero must be kept and no arithmetic is done on it
D. A Boolean, because a number is either valid or not

489. A byte is sent with even parity, but on the way two of its bits are flipped. What does the receiver's parity check report? Analysis

A. An error in the first flipped bit
B. An error, and both flipped bits are corrected
C. No error, because flipping two bits leaves the count of 1s even
D. An error, because any change makes the count odd

490. A game stores 8 independent on/off settings (sound, music, vibration and so on) for millions of players, and memory must be minimised. Which representation fits best? Analysis

A. Eight 32-bit integers per player, one per setting
B. One byte per player, with each setting as one bit (a bit field)
C. One string per player, such as "on,off,on,..."
D. Eight floating-point numbers per player

491. What does this Python code print? Analysis

```
s = 'café'
print(len(s.encode('utf-8')), len(s))
```

A. 4 4
B. 5 4
C. 4 5
D. 8 4

492. A memory dump shows the byte C8 (hexadecimal). Which statement about its value is correct? Analysis

A. It is 200 under every interpretation
B. It is -72 if read as 8-bit two's complement
C. It is 200 if read as unsigned and -56 if read as 8-bit two's complement
D. It is -200 if read as signed, because the top bit is 1

493. A program stores product codes as text and sorts them. What does this Python code print? Analysis

```
print(sorted(['9', '10', '100']))
```

A. ['10', '100', '9']
B. ['9', '10', '100']
C. ['100', '10', '9']
D. ['9', '100', '10']

494. Which of these decimal values cannot be stored exactly in binary floating point? Analysis

A. 0.75
B. 0.5
C. 2.5
D. 0.2

495. A social network has 1 million users, and each user has about 100 friends. To store the friendships, which graph representation should be chosen, and why? Analysis

    A. An adjacency list, because the graph is sparse and storage grows with the number of edges
    B. An adjacency matrix, because it uses the least memory for any graph
    C. An adjacency matrix, because 100 friends per user makes the graph dense
    D. A single sorted array of user names

496. A 16-bit signed (two's complement) counter holds 32767, its maximum, and is incremented once with overflow ignored. What value does it then hold? Analysis

    A. 32768
    B. -32768
    C. 0
    D. -1

497. What does this Python code print? Analysis

```
print(1e16 + 1 - 1e16)
```

    A. 0.0
    B. 1.0
    C. 1e16
    D. 2.0

498. A sensor reports temperatures from -40 °C to 85 °C in steps of 0.5 °C. Each reading is stored as a step number. What is the fewest bits per reading? Analysis

    A. 7
    B. 9
    C. 125
    D. 8

499. A little-endian file stores a 16-bit unsigned integer as two bytes, 34 then 12 (hexadecimal), at increasing addresses. What decimal value is stored? Analysis

    A. 13330
    B. 1234
    C. 52
    D. 4660

500. In 8-bit two's complement, which addition overflows? Analysis

    A. 100 + 27
    B. -100 + -28
    C. 64 + 64
    D. -128 + 127

Pattern Recognition & Generalization MCQs for ...

501. What makes a sequence arithmetic? Understanding

    A. The ratio between consecutive terms is constant
    B. Each term is the sum of the two before it
    C. The difference between consecutive terms is constant
    D. Every term is a perfect square

502. Which of these is a geometric sequence? Understanding

A. 3, 6, 9, 12
B. 1, 4, 9, 16
C. 1, 1, 2, 3
D. 3, 6, 12, 24

503. What rule produces the Fibonacci sequence 0, 1, 1, 2, 3, 5, 8, ...? Understanding

A. Each term is the sum of the two terms before it
B. Each term is double the previous term
C. Each term is the previous term plus 1
D. Each term is the square of its position

504. For an arithmetic sequence with first term a and common difference d, what is the nth term? Understanding

A. a + nd
B. a + (n - 1)d
C. a × d^(n - 1)
D. n(a + d)

505. The first differences of a sequence are not constant, but its second differences are constant. What kind of formula generates it? Understanding

A. A linear formula: an + b
B. An exponential: a × r^n
C. A quadratic: $an^2 + bn + c$
D. No formula exists

506. You observe several specific cases and form a general rule that seems to cover them all. What kind of reasoning is this? Understanding

A. Deductive reasoning
B. Inductive reasoning
C. Circular reasoning
D. Proof by contradiction

507. Which of these defines a sequence by a recurrence relation? Understanding

A. t(n) = 4n - 1
B. $t(n) = n^2 + 2$
C. t(n) = 3 × 2^n
D. t(1) = 3, t(n) = t(n - 1) + 4

508. A question shows only 1, 2, 4, ? and asks for the next term. Why is it risky to answer without more context? Understanding

A. Three terms are always enough to fix a unique rule
B. Several different rules fit three terms, for example doubling (8) or adding 1, 2, 3 (7)
C. Sequences starting with 1 cannot have a pattern
D. Only geometric sequences can start with 1, 2

509. What is the formula for the sum 1 + 2 + 3 + ... + n? Understanding

A. $n^2 / 2$
B. n(n + 1) / 2
C. n(n - 1) / 2
D. 2n + 1

510. What is the sum of the first n terms of a geometric sequence with first term a and ratio r (r ≠ 1)? Understanding

A. a × r^(n - 1)
B. n(a + ar) / 2
C. a(r^n - 1) / (r - 1)
D. a(r - 1) / (r^n - 1)

511. The triangular numbers are 1, 3, 6, 10, 15, ... How does each term relate to the one before? Understanding

A. Each term is double the previous one
B. The amount added goes up by 1 each time
C. Each term is the previous one plus 2
D. Each term is a perfect square

512. In a number series such as 2, 50, 4, 45, 6, 40, ..., what is the usual pattern to look for? Understanding

A. A single rule applied to every consecutive pair
B. Two separate sequences in alternate positions
C. A random list with no rule
D. A geometric sequence with a negative ratio

513. To prove that a formula holds for every positive integer n by mathematical induction, what must you show? Understanding

A. It holds for the first ten values of n
B. It holds for n = 1, and if it holds for n = k it also holds for n = k + 1
C. It holds for n = 1 and for one very large n
D. It holds for n = k + 1 assuming it holds for n = k + 1

514. The last digits of 7^1, 7^2, 7^3, ... repeat in a cycle. How long is that cycle? Understanding

A. 4
B. 2
C. 7
D. 10

515. What is the main advantage of a closed-form formula over a recurrence for the same sequence? Understanding

A. It always uses fewer symbols
B. It works for sequences that have no pattern
C. It gives any term directly from n, without computing all earlier terms
D. It avoids the need for a starting value only in geometric sequences

516. In computational thinking, what does generalisation mean? Understanding

A. Making a solution vaguer so it is easier to explain
B. Testing a program on one typical input
C. Turning a solution for specific cases into one rule or algorithm that works for a whole class of problems
D. Replacing every variable in a program with a constant

517. What is the next term: 3, 7, 11, 15, ? Application

A. 18
B. 20
C. 21
D. 19

518. What is the next term: 2, 6, 18, 54, ? Application

A. 162
B. 108
C. 72
D. 216

519. What is the next term: 1, 4, 9, 16, 25, ? Application

A. 34
B. 30
C. 49
D. 36

520. What is the next term: 2, 3, 5, 8, 12, 17, ? Application

A. 22
B. 25
C. 26
D. 23

521. What is the 50th term of 7, 12, 17, 22, ...? Application

A. 257
B. 250
C. 252
D. 247

522. What is the next term: 2, 5, 7, 12, 19, ? Application

A. 31
B. 26
C. 29
D. 38

523. What is the next term: 3, 6, 11, 18, 27, ? Application

A. 36
B. 38
C. 37
D. 40

524. What is the next term: 2, 5, 11, 23, 47, ? Application

A. 95
B. 94
C. 71
D. 96

525. What is the next letter: A, C, F, J, O, ? Application

A. U
B. T
C. V
D. S

526. A row of 1 square uses 4 matchsticks, 2 joined squares use 7 and 3 joined squares use 10. How many matchsticks do 25 joined squares need? Application

A. 100
B. 75
C. 79
D. 76

527. What is the next term: 1, 10, 3, 20, 5, 30, 7, ? Application

A. 9
B. 35
C. 40
D. 37

528. What is the last digit of 7^2026? Application

A. 7
B. 3
C. 9
D. 1

529. What is the sum of the first 20 terms of 4, 7, 10, 13, ...? Application

A. 610
B. 670
C. 650
D. 1300

530. 12 people at a meeting each shake hands once with every other person. How many handshakes take place? Application

A. 66
B. 132
C. 144
D. 78

531. One term in this series is wrong: 2, 6, 12, 20, 30, 40, 56. Which is it? Analysis

A. 40
B. 30
C. 20
D. 56

532. This Python code prints a triangle pattern but counts the stars instead. What does it print? Analysis

```
n = 10
total = 0
for i in range(1, n + 1):
    row = "*" * (2 * i - 1)
    total += len(row)
print(total)
```

A. 55
B. 110
C. 19
D. 100

533. Place n points on a circle and join every pair with a chord, with no three chords meeting at one interior point. For n = 1 to 5 the circle is cut into 1, 2, 4, 8 and 16 regions. How many regions for n = 6? Analysis

A. 31
B. 32
C. 30
D. 24

534. Which formula generates all four terms 5, 8, 11, 14 (for n = 1, 2, 3, 4)? Analysis

A. $n^2 + 4$
B. 5n
C. 3n + 5
D. 3n + 2

535. A sequence is defined by a(1) = 2 and a(n) = a(n - 1) + 2n. Which closed form gives the same sequence? Analysis

A. $a(n) = n^2 + 1$
B. a(n) = n(n + 1)
C. a(n) = 2n
D. a(n) = 2^n

536. What is the last digit of 3^1 + 3^2 + 3^3 + ... + 3^100? Analysis

A. 1
B. 3
C. 9
D. 0

537. $n^2 + n + 41$ is prime for every n from 0 to 39. A student concludes it is prime for every n. What does n = 40 show? Analysis

A. The rule holds, so the student's conclusion is proved
B. Nothing, because n = 40 is outside the range of whole numbers
C. The rule fails: $40^2 + 40 + 41 = 1681 = 41^2$, so many matching cases do not prove a pattern
D. The rule fails only because 40 is even

538. The minimum moves for the Tower of Hanoi with 1, 2, 3 and 4 discs are 1, 3, 7 and 15. Generalising the pattern, how many moves are needed for 10 discs? Analysis

A. 1023
B. 1024
C. 511
D. 19

539. What is the next term: 1, 2, 6, 24, 120, ? Analysis

A. 240
B. 600
C. 144
D. 720

540. How many squares of all sizes are there in a 4 × 4 grid of unit squares? Analysis

A. 16
B. 30
C. 20
D. 25

541. Which list shows the first four cube numbers? Understanding

A. 1, 8, 27, 64
B. 1, 4, 9, 16
C. 3, 6, 9, 12
D. 1, 3, 9, 27

542. What is the common ratio of the geometric sequence 81, 27, 9, 3? Understanding

A. 1/3
B. 3
C. -54
D. -3

543. What kind of sequence is 3, -6, 12, -24, ...? Understanding

A. Geometric with common ratio -2
B. Arithmetic with common difference -9
C. Geometric with common ratio 2
D. Neither arithmetic nor geometric

544. Using the proven formula $n(n + 1)/2$ to work out $1 + 2 + ... + 50$ is an example of which kind of reasoning? Understanding

A. Inductive reasoning: forming a rule from examples
B. Abductive reasoning: guessing the most likely cause
C. Trial and error
D. Deductive reasoning: applying a general rule to a specific case

545. A student claims a rule holds for every whole number. What does finding one value where it fails show? Understanding

A. The claim is still true for most numbers, so it stands
B. The claim needs one more confirming example
C. Nothing, because one case cannot matter
D. The general claim is false

546. A sequence's first and second differences change, but its third differences are constant. What kind of formula generates it? Understanding

A. A cubic: $an^3 + bn^2 + cn + d$
B. A quadratic: $an^2 + bn + c$
C. A geometric formula: a × r^n
D. A linear formula: an + b

547. What is the sum of the first n odd numbers, $1 + 3 + 5 + ... + (2n - 1)$? Understanding

A. n^2
B. $n(n + 1)/2$
C. $2n - 1$
D. $n(n + 1)$

548. Which formula gives $1^2 + 2^2 + 3^2 + ... + n^2$? Understanding

A. $n(n + 1)(2n + 1)/6$
B. $n(n + 1)/2$
C. $[n(n + 1)/2]^2$
D. $n^3/3$

549. When does an infinite geometric series $a + ar + ar^2 + ...$ have a finite sum, and what is it? Understanding

A. When $r > 1$; the sum is $a / (r - 1)$
B. When $|r| < 1$; the sum is $a / (1 - r)$
C. For every r; the sum is a × r
D. When $r = 1$; the sum is a

550. In computational thinking, noticing that several sub-problems share the same structure, so one method can solve them all, is called what? Understanding

A. Pattern recognition
B. Decomposition
C. Debugging
D. Encryption

551. Sequence P is arithmetic with a positive difference, and sequence Q is geometric with a ratio greater than 1. Both start with positive terms. What happens for large n? Understanding

A. Q eventually exceeds P and the gap keeps growing, however large P's difference is
B. P always stays ahead if its difference is larger than Q's ratio
C. They grow at the same rate after enough terms
D. Q eventually falls back below P

552. As the Fibonacci sequence continues, the ratio of each term to the one before it approaches which value? Understanding

A. Exactly 2
B. About 1.414 (the square root of 2)
C. About 1.618 (the golden ratio)
D. It keeps growing without limit

553. In Pascal's triangle, with the top row counted as row 0, what is the sum of the numbers in row n? Understanding

A. n^2
B. 2^n
C. n(n + 1)/2
D. 2n

554. A recurrence is t(n) = t(n - 1) + t(n - 2). How many starting values must be given before every term is determined? Understanding

A. One
B. Three
C. Two
D. None, because the rule fixes every term

555. Plotted against n, what kind of formula is the nth term of any arithmetic sequence? Understanding

A. Quadratic
B. Exponential
C. Constant for every arithmetic sequence
D. Linear, with slope equal to the common difference

556. A student fits a degree-4 polynomial exactly through the first 5 terms of a sequence and treats this as strong evidence for the rule. Why is that reasoning weak? Understanding

A. A degree-4 polynomial can pass through any 5 points, so the fit was guaranteed whatever the true rule
B. Polynomials can never describe number sequences
C. Five terms are always enough to prove a rule
D. A degree-4 fit only works when all terms are even

557. Why is the digit sum of every multiple of 9 itself a multiple of 9? Understanding

A. Because 9 is the largest single digit
B. Because multiples of 9 always have exactly two digits
C. Every power of 10 leaves remainder 1 when divided by 9, so a number and its digit sum leave the same remainder
D. Because 9 is a square number

558. What is always true of the units digit of n^5 for any whole number n? Understanding

A. It is always 5
B. It equals the units digit of n
C. It is always 0 or 5
D. It is always 1 more than the units digit of n

559. What is the entry in position k of row n of Pascal's triangle (counting both from 0)? Understanding

A. n × k
B. 2^k
C. n + k
D. The binomial coefficient C(n, k), the number of ways to choose k items from n

560. Why does a naive recursive function for t(n) = t(n - 1) + t(n - 2) become very slow for large n? Understanding

A. Recursion cannot handle addition efficiently
B. Each call stores the whole sequence in memory
C. Larger n always needs larger data types
D. It recomputes the same smaller terms over and over, so the number of calls grows exponentially

561. How should the constant sequence 7, 7, 7, 7, ... be classified? Understanding

A. Only arithmetic
B. Only geometric
C. Neither, because the terms do not change
D. Both arithmetic (d = 0) and geometric (r = 1)

562. Square numbers go 1, 4, 9, 16, 25. By how much does n^2 increase to reach $(n + 1)^2$? Understanding

A. 2n
B. n + 1
C. 2n + 1
D. n^2

563. What happens to the sum 1 + 1/2 + 1/3 + 1/4 + ... (the harmonic series) as more terms are added? Understanding

A. It approaches 2
B. It approaches 1
C. It approaches e, about 2.718
D. It grows without limit, although very slowly

564. What kind of sequence is 1, 1/2, 1/4, 1/8, ...? Understanding

A. Arithmetic with common difference -1/2
B. Geometric with common ratio 2
C. Geometric with common ratio 1/2
D. Arithmetic with common difference 1/2

565. What is the next term: 64, 32, 16, 8, ? Application

A. 4
B. 0
C. 6
D. 2

566. What is the next term: 125, 216, ? Application

A. 307
B. 256
C. 343
D. 324

567. What is the next term: 13, 17, 19, 23, ? Application

A. 25
B. 29
C. 27
D. 31

568. What is the next term: 50, 43, 36, 29, ? Application

A. 21
B. 23
C. 22
D. 15

569. What comes next: AZ, BY, CX, ? Application

A. DY
B. DW
C. EW
D. DV

570. What is the sum of the first 15 odd numbers, 1 + 3 + ... + 29? Application

A. 120
B. 450
C. 240
D. 225

571. What does this Python code print? Application

```
print(sum(i * i for i in range(1, 11)))
```

A. 55
B. 3025
C. 100
D. 385

572. How many terms are in the sequence 7, 11, 15, ..., 203? Application

A. 49
B. 51
C. 50
D. 196

573. What is the next term: 1, 2, 6, 15, 31, ? Application

A. 47
B. 62
C. 52
D. 56

574. What is the next letter: Z, X, U, Q, L, ? Application

A. G
B. F
C. E
D. H

575. What is the next term: 2, 5, 14, 41, 122, ? Application

A. 365
B. 243
C. 203
D. 366

576. What is the missing term: 5, 12, ?, 32, 45 Application

A. 19
B. 21
C. 22
D. 24

577. What is 3 + 6 + 12 + 24 + ... up to and including the 10th term? Application

A. 3069
B. 1536
C. 3072
D. 1533

578. A quadrilateral has 2 diagonals, a pentagon 5 and a hexagon 9. Using the general rule, how many diagonals does a 12-sided polygon have? Application

A. 66
B. 108
C. 45
D. 54

579. A staircase made of cubes has 1 cube in the first column, 2 in the second, and so on. How many cubes are needed for 20 columns? Application

A. 200
B. 210
C. 400
D. 420

580. What is the last digit of 2^100? Application

A. 2
B. 4
C. 8
D. 6

581. Today is Monday. What day of the week will it be 100 days from today? Application

A. Wednesday
B. Tuesday
C. Thursday
D. Monday

582. What does this Python code print? Application

```
print([n * n - n for n in range(1, 6)])
```

A. [1, 4, 9, 16, 25]
B. [0, 2, 6, 12, 20, 30]
C. [0, 2, 6, 12, 20]
D. [2, 6, 12, 20, 30]

583. What does this Python code print? Application

```
def f(n):
    if n == 1:
        return 1
    return f(n - 1) + 2 * n - 1

print(f(12))
```

A. 23
B. 78
C. 144
D. 169

584. In the look-and-say sequence, each term describes the previous one aloud: 1, 11, 21, 1211, 111221, ... What is the next term? Application

A. 1112221
B. 13112221
C. 312211
D. 312221

585. What does this Python code print? Application

```
t = 1
for n in range(2, 7):
    t = 2 * t + n
print(t)
```

A. 57
B. 63
C. 120
D. 247

586. Which rule produces all four terms 2, 4, 8, 14 for n = 1, 2, 3, 4? Analysis

A. 2^n
B. n^2 - n + 2
C. 3n - 1
D. 2n + 2^(n-2)

587. One term in this series is wrong: 1, 3, 9, 27, 81, 242, 729. Which is it? Analysis

A. 81
B. 242
C. 729
D. 27

588. Which number does not belong: 121, 144, 169, 186, 196? Analysis

A. 121
B. 169
C. 196
D. 186

589. This function handles only three inputs with special cases. Which single return statement generalises it for every positive n while keeping these outputs? Analysis

```
def f(n):
    if n == 1:
        return 3
    if n == 2:
        return 5
    if n == 3:
        return 7
```

A. return n + 2
B. return 2 * n
C. return 2 * n + 1
D. return n * n + 2

590. A student checks that $n^2 + n$ is even for n = 1 to 5 (2, 6, 12, 20, 30). Which conclusion is justified? Analysis

A. It is even for every n, because $n^2 + n = n(n + 1)$ is the product of two consecutive integers, one of which is even
B. It is even for every n, because five cases are enough to establish a pattern
C. It is even only for n up to 5, since no more cases were checked
D. It is even only when n is even

591. A 2 × n strip is tiled with 1 × 2 dominoes. There is 1 tiling for n = 1, 2 for n = 2, 3 for n = 3 and 5 for n = 4. How many tilings are there for n = 7? Analysis

A. 13
B. 21
C. 14
D. 34

592. A naive recursive function computes f(n) = f(n - 1) + f(n - 2), returning 1 when n ≤ 1. Including the first call, how many calls are made to compute f(5)? Analysis

A. 5
B. 8
C. 15
D. 9

593. A robot moves on a 3 × 3 grid of squares from the top-left corner point to the bottom-right corner point along the grid lines, only right or down. How many different shortest paths exist? Analysis

A. 9
B. 6
C. 20
D. 27

594. Plan A pays Rs 1,000 × n in month n. Plan B pays Rs 10 in month 1 and doubles every month. In which month does Plan B's payment first exceed Plan A's? Analysis

A. Month 10
B. Month 11
C. Month 7
D. Month 12

595. In the Collatz process, an even number is halved and an odd number n becomes 3n + 1. Starting from 6, how many steps does it take to reach 1? Analysis

A. 6
B. 9
C. 5
D. 8

596. A program's test table gives x = 0, 1, 2, 3 and outputs 1, 3, 7, 13. Which formula matches every row? Analysis

A. 2x + 1
B. $x^2 + x + 1$
C. 2^(x+1) - 1
D. 3x + 1

597. A student claims the sum of the first n even numbers, 2 + 4 + ... + 2n, is n^2. What is the correct general formula? Analysis

A. n(n + 1)
B. n^2
C. $2n^2$
D. n(n + 1)/2

598. This code should print a centred pyramid of height 4: row i has 2i - 1 stars and the last row starts at the left margin. What expression should replace ??? Analysis

```
h = 4
for i in range(1, h + 1):
    print(" " * ??? + "*" * (2 * i - 1))
```

A. i
B. h - i
C. h - i + 1
D. 2 * i - 1

599. What is the remainder when 2^50 is divided by 7? Analysis

A. 1
B. 4
C. 2
D. 0

600. Straight lines are drawn so that no two are parallel and no three meet at one point. 1 line makes 2 regions, 2 lines make 4, 3 lines make 7. How many regions do 10 lines make? Analysis

A. 1024
B. 56
C. 46
D. 55

Mathematical & Quantitative Reasoning MCQs for...

601. What is 3/4 written as a percentage? Understanding

A. 34%
B. 7.5%
C. 75%
D. 43%

602. What is the ratio 12:18 in its simplest form? Understanding

A. 3:2
B. 6:9
C. 1:6
D. 2:3

603. Which value can NOT be the probability of an event? Understanding

A. 0
B. 0.5
C. 1
D. 1.2

604. In how many different orders can 4 different books be placed in a row on a shelf? Understanding

A. 24
B. 16
C. 10
D. 4

605. Events A and B are mutually exclusive. Which formula gives P(A or B)? Understanding

A. $P(A) \times P(B)$
B. $P(A) + P(B) - P(A) \times P(B)$
C. $P(A) + P(B)$
D. $1 - P(A) - P(B)$

606. What is the key difference between a permutation and a combination? Understanding

A. A permutation allows repeated items; a combination never does
B. In a permutation the order of the chosen items matters; in a combination it does not
C. A combination always gives the larger count
D. Permutations apply only to numbers and combinations only to letters

607. A pass rate rises from 20% to 25%. Which description is correct? Understanding

A. It rose by 5%, which is the same as 5 percentage points
B. It rose by 25 percentage points
C. It rose by 5 percentage points, which is a 25% relative increase
D. It rose by 20% relative to the old rate

608. If A and B are independent events, which formula gives P(A and B)? Understanding

A. $P(A) + P(B)$
B. $P(A) - P(B)$
C. $P(A) \times P(B)$
D. $P(A) \div P(B)$

609. Why is the complement rule P(at least one) = 1 − P(none) often used in probability questions? Understanding

A. Counting the single 'none' case is usually easier than adding up every 'at least one' case
B. Because 'at least one' events are always independent
C. Because P(none) is always 0.5
D. Because it works only when the events are mutually exclusive

610. y is directly proportional to x. If x is doubled, what happens to y? Understanding

A. y halves
B. y stays the same
C. y doubles
D. y is squared

611. A data set of salaries contains one extremely large value. Which average is least affected by it? Understanding

A. Mean
B. Median
C. Sum divided by count
D. Midpoint of the smallest and largest values

612. A price is increased by 10% and then decreased by 10%. Why does it not return to the original price? Understanding

A. Percentages cannot be applied twice
B. The decrease is calculated on the original price
C. Because 10% of any number is always rounded down
D. The 10% decrease is calculated on the larger, increased price

613. Which value is equal to C(10, 3), the number of ways to choose 3 items from 10? Understanding

A. C(10, 4)
B. C(10, 7)
C. P(10, 3)
D. 10 × 3

614. A student has 3 shirts and 4 pairs of trousers. Which principle gives the number of different outfits? Understanding

A. The multiplication principle: 3 × 4
B. The addition principle: 3 + 4
C. The complement rule: 1 − 3/4
D. The pigeonhole principle

615. Several workers do a fixed job at the same rate. How are the number of workers and the days needed related? Understanding

A. Directly proportional: more workers need more days
B. Inversely proportional: more workers need fewer days
C. Unrelated
D. Their sum is always constant

616. Why is the probability of at least one six in 4 rolls of a fair die NOT 4 × 1/6 = 2/3? Understanding

A. Sixes on different rolls can happen together, so adding the four probabilities double-counts outcomes
B. The die is no longer fair after the first roll
C. The rolls are mutually exclusive, so the probabilities must be multiplied
D. The probability of a six changes on every roll

617. What is 15% of 240? Application

A. 24
B. 16
C. 3.6
D. 36

618. A course fee rises from 80 to 100 units. What is the percentage increase? Application

A. 25%
B. 20%
C. 80%
D. 125%

619. 1,200 units are shared between two partners in the ratio 3:5. How much does the larger share get? Application

A. 450
B. 720
C. 750
D. 800

620. Two fair dice are rolled. What is the probability that the total is 7? Application

A. 1/12
B. 1/6
C. 7/36
D. 1/11

621. In how many ways can a 3-member committee be chosen from 7 students? Application

A. 210
B. 21
C. 343
D. 35

622. How many distinct arrangements are there of the letters of the word LEVEL? Application

A. 120
B. 30
C. 60
D. 20

623. A shop gives a 20% discount and then a further 10% off the discounted price. What is the overall discount? Application

A. 30%
B. 32%
C. 28%
D. 27%

624. A student travels 60 km at 30 km/h and then another 60 km at 60 km/h. What is the average speed for the whole trip? Application

A. 40 km/h
B. 45 km/h
C. 50 km/h
D. 35 km/h

625. A bag has 5 red and 3 blue balls. Two are drawn without replacement. What is the probability that both are red? Application

A. 25/64
B. 5/14
C. 5/8
D. 15/28

626. A fair coin is tossed 3 times. What is the probability of getting at least one head? Application

A. 1/2
B. 3/8
C. 7/8
D. 1/8

627. 10 litres of water are added to 20 litres of a solution that is 30% acid. What percentage of the new solution is acid? Application

A. 15%
B. 30%
C. 25%
D. 20%

628. After a 25% increase, a laptop costs 150,000 units. What was the original price? Application

A. 112,500
B. 120,000
C. 125,000
D. 187,500

629. 5,000 units are deposited at 8% simple interest per year. How much interest is earned in 3 years? Application

A. 400
B. 1,200
C. 1,298.56
D. 6,200

630. A 4-digit PIN uses the digits 0 to 9 and no digit may repeat. How many PINs are possible? Application

A. 10,000
B. 210
C. 6,561
D. 5,040

631. A student's marks rose from 40 to 50 and then fell back to 40. Which statement is correct? Analysis

A. The rise was 25% and the fall was 20%
B. The rise and the fall were both 25%
C. The rise and the fall were both 20%
D. The rise was 20% and the fall was 25%

632. Section A has 30 students averaging 60 marks. Section B has 20 students averaging 70. What is the average of all 50 students? Analysis

A. 65
B. 66
C. 64
D. 62

633. 1% of students have a rare condition. A screening test detects 90% of real cases but also flags 10% of healthy students. A student tests positive. Roughly what is the chance they have the condition? Analysis

A. About 90%
B. About 10%
C. About 8%
D. About 1%

634. Three people are picked at random. Assuming each day of the week is equally likely, what is the probability that at least two of them were born on the same weekday? Analysis

A. 3/7
B. 1/49
C. 30/49
D. 19/49

635. A:B = 2:3 and B:C = 4:5. What is A:B:C? Analysis

A. 8:12:15
B. 2:3:5
C. 2:4:5
D. 8:12:20

636. At a meeting of 10 people, everyone shakes hands with everyone else exactly once. A student says there are 90 handshakes. What is the correct number, and what was the mistake? Analysis

A. 45; each handshake was counted twice, once for each person
B. 90; the student is correct
C. 100; each person shakes 10 hands
D. 9; each person shakes only one hand per other person

637. A student claims P(at least one six in 2 rolls of a fair die) = 1/6 + 1/6 = 1/3. What is the correct value? Analysis

A. 1/3
B. 1/36
C. 25/36
D. 11/36

638. 6 workers can finish a job in 10 days. After 4 days, 2 workers leave. If everyone works at the same rate, how many days does the whole job take? Analysis

A. 15
B. 13
C. 12
D. 9

639. A town of 50,000 grows by 10% each year. Which population after 2 years is correct? Analysis

A. 60,000
B. 55,000
C. 61,000
D. 60,500

640. A 2-person team must be chosen from 5 men and 4 women, with at least one woman. How many teams are possible? Analysis

A. 26
B. 36
C. 20
D. 16

641. What is 0.125 written as a fraction in its simplest form? Understanding

A. 1/80
B. 1/8
C. 5/8
D. 1/12

642. The probability that it rains tomorrow is 0.35. What is the probability that it does NOT rain? Understanding

A. 0.35
B. 0.65
C. 0.75
D. 1.35

643. In counting formulas, what is the value of 0! (zero factorial)? Understanding

A. 0
B. 1
C. Undefined
D. Infinity

644. A fair six-sided die is rolled once. What is the expected (mean) value of the number shown? Understanding

A. 3
B. 6
C. 21
D. 3.5

645. Which average is defined as the value that occurs most often in a data set? Understanding

A. Mode
B. Median
C. Mean
D. Range

646. y is inversely proportional to x. If x is tripled, what happens to y? Understanding

A. y becomes one third of its old value
B. y triples
C. y decreases by 3
D. y stays the same

647. What does the range of a data set measure? Understanding

A. The spread between the largest and smallest values
B. The most frequent value
C. The middle value after sorting
D. The total of all values divided by the count

648. Two fair coins are tossed together. How many outcomes are in the sample space? Understanding

A. 3
B. 2
C. 6
D. 4

649. In a class, the ratio of boys to girls is 2:3. What fraction of the class are boys? Understanding

A. 2/3
B. 2/5
C. 3/5
D. 1/2

650. Which formula gives the conditional probability of A given that B has happened, when $P(B) > 0$? Understanding

A. $P(A \mid B) = P(A \text{ and } B) / P(A)$
B. $P(A \mid B) = P(A) \times P(B)$
C. $P(A \mid B) = P(A \text{ and } B) / P(B)$
D. $P(A \mid B) = P(A) + P(B) - P(A \text{ and } B)$

651. Events A and B can happen together. Which formula gives P(A or B)? Understanding

A. $P(A) + P(B)$
B. $P(A) \times P(B)$
C. $P(A) + P(B) - P(A \text{ and } B)$
D. $P(A) + P(B) + P(A \text{ and } B)$

652. The odds in favour of an event are 1:3. What is the probability of the event? Understanding

A. 1/3
B. 1/4
C. 3/4
D. 1/2

653. In how many distinct ways can n different people sit around a round table, if seatings that are rotations of each other count as the same? Understanding

A. n!
B. (n − 1)!
C. n! / 2
D. n − 1

654. Why does compound interest grow faster than simple interest at the same rate? Understanding

A. Compound interest uses a higher rate than the one stated
B. Each period's interest is added to the principal, so later interest is also earned on earlier interest
C. Simple interest is only paid at the end, while compound interest is paid at the start
D. Compound interest is calculated on the original principal every year

655. Every mark in a test is increased by 5. What happens to the mean and the range of the marks? Understanding

A. Both the mean and the range rise by 5
B. The mean rises by 5 and the range stays the same
C. The mean stays the same and the range rises by 5
D. Both stay the same

656. How many subsets (including the empty set and the set itself) does a set with n elements have? Understanding

A. 2^n
B. n^2
C. n!
D. 2n

657. Which formula gives the number of ordered arrangements of r items chosen from n different items, P(n, r)? Understanding

A. n! / (n − r)!
B. n! / (r! (n − r)!)
C. n^r
D. r! / (n − r)!

658. Two balls are drawn one after the other from a bag WITHOUT replacement. Why are the two draws not independent? Understanding

A. The two draws happen at different times
B. The first draw changes what is left in the bag, so it changes the probabilities for the second draw
C. Each draw has the same probability of every colour
D. Probabilities in a bag always add up to more than 1

659. A value changes from an old amount to a new amount. Which expression gives the percentage change? Understanding

A. (new − old) / old × 100
B. (new − old) / new × 100
C. new / old × 100
D. (old − new) / (old + new) × 100

660. A and B are complementary events. Which statement is true? Understanding

A. They are independent and $P(A) \times P(B) = 1$
B. They are mutually exclusive and $P(A) + P(B) = 1$
C. They can both happen together
D. P(A) must equal P(B)

661. A and B are mutually exclusive, and both P(A) and P(B) are greater than 0. Can A and B be independent? Understanding

A. Yes, mutually exclusive events are always independent
B. Yes, as long as $P(A) + P(B) \leq 1$
C. Only if P(A) = P(B)
D. No, because P(A and B) = 0 but $P(A) \times P(B) > 0$

662. Why does the combination formula $C(n, r) = n! / (r! (n - r)!)$ divide by r!? Understanding

A. To remove the items that were not chosen
B. Because r! is always larger than (n − r)!
C. To turn a probability into a count
D. Each unordered group of r items appears r! times among the ordered arrangements, so dividing removes the repeats

663. Which formula correctly gives $|A \cup B \cup C|$ for three finite sets (the inclusion-exclusion principle)? Understanding

A. $|A| + |B| + |C| - |A \cap B \cap C|$
B. $|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C|$
C. $|A| + |B| + |C| - 2|A \cap B \cap C|$
D. $|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$

664. $P(B) > 0$. Which condition is equivalent to A and B being independent? Understanding

A. $P(A \mid B) = P(B)$
B. P(A and B) = 0
C. $P(A \mid B) = P(A)$
D. $P(A) + P(B) = 1$

665. What is 12.5% of 640? Application

A. 64
B. 80
C. 75
D. 800

666. A bag costs 500 units. It is sold at a 15% discount. What is the sale price? Application

A. 425
B. 485
C. 75
D. 575

667. What is the mean of 4, 8, 15, 16, 23 and 42? Application

A. 18
B. 15.5
C. 16
D. 108

668. What is the median of 7, 3, 9, 1, 5 and 11? Application

A. 6
B. 5
C. 7
D. 9

669. A whole number is chosen at random from 1 to 20. What is the probability that it is a multiple of 4? Application

A. 1/4
B. 1/5
C. 3/10
D. 1/20

670. A map has a scale of 1:50,000. Two towns are 4 cm apart on the map. What is the real distance? Application

A. 20 km
B. 200 m
C. 12.5 km
D. 2 km

671. What is the value of 5! / 3!? Application

A. 2
B. 40
C. 120
D. 20

672. How many distinct arrangements are there of the letters of the word BANANA? Application

A. 720
B. 120
C. 60
D. 36

673. In how many distinct ways can 5 people sit around a round table, if rotations of the same seating count as one? Application

A. 120
B. 60
C. 5
D. 24

674. 10,000 units are invested at 10% per year, compounded annually. How much interest is earned after 2 years? Application

A. 2,100
B. 2,000
C. 12,100
D. 1,000

675. A society of 8 students must choose a president and a secretary, and one person cannot hold both posts. How many choices are possible? Application

A. 28
B. 64
C. 56
D. 16

676. One card is drawn from a standard 52-card deck. What is the probability that it is a king or a heart? Application

A. 4/13
B. 17/52
C. 1/4
D. 1/13

677. Ali and Babar share money in the ratio 4:7. Babar gets 21 units more than Ali. How much do they have altogether? Application

A. 33
B. 49
C. 84
D. 77

678. A can finish a job in 6 days and B can finish it in 3 days. Working together at these rates, how long do they take? Application

A. 4.5 days
B. 9 days
C. 3 days
D. 2 days

679. A car travels at 72 km/h. What is this speed in metres per second? Application

A. 72 m/s
B. 7.2 m/s
C. 259.2 m/s
D. 20 m/s

680. Selling a phone for 1,150 units gives a 15% profit on cost. At what price must it be sold to make a 25% profit? Application

A. 1,265
B. 1,322.5
C. 1,250
D. 1,437.5

681. How many 3-digit numbers (100 to 999) have three different digits? Application

A. 720
B. 504
C. 900
D. 648

682. A fair coin is tossed 4 times. What is the probability of getting exactly 2 heads? Application

A. 1/2
B. 1/4
C. 3/8
D. 1/16

683. A 3-person committee is chosen from 5 men and 4 women. How many committees have exactly 2 women? Application

A. 30
B. 60
C. 84
D. 40

684. How many litres of pure acid must be added to 20 litres of a 10% acid solution to make a 25% acid solution? Application

A. 3
B. 5
C. 4
D. 6

685. A fair coin has landed heads 5 times in a row. A student says tails is now 'due', so P(tails on the 6th toss) is more than 1/2. What is correct? Analysis

A. P(tails) is still 1/2, because tosses are independent
B. P(tails) is 5/6, because 5 heads have already happened
C. P(tails) is 1/64, because 6 heads in a row is rare
D. P(tails) is 1, because the results must balance out

686. Class A: 36 of 40 students passed. Class B: 45 of 60 passed. Which comparison is correct? Analysis

A. Class B has the higher pass rate, because 45 is more than 36
B. Both classes have the same pass rate
C. Class B has the higher pass rate: 75% against 60%
D. Class A has the higher pass rate: 90% against 75%

687. A shop's selling price is 25% more than its cost price. By what percentage is the cost price less than the selling price? Analysis

A. 25%
B. 75%
C. 80%
D. 20%

688. Two fair dice are rolled. A student says a total of 2 and a total of 7 are equally likely, since each is just one possible total. What is correct? Analysis

A. Both are 1/11
B. Both are 1/36
C. P(total 2) = 1/36 and P(total 7) = 1/6
D. P(total 2) = 1/6 and P(total 7) = 1/36

689. Two cards are drawn from a 52-card deck without replacement. A student says P(both aces) = (4/52) × (4/52). What is the correct probability? Analysis

A. 1/169
B. 1/221
C. 1/26
D. 2/13

690. Two fair coins are tossed. A student lists the outcomes as 'two heads, one head, no heads' and says P(one head and one tail) = 1/3. What is correct? Analysis

A. 1/3, because there are three possible results
B. 1/2, because HT and TH are two separate outcomes out of 4
C. 1/4, because only one outcome has exactly one head
D. 2/3, because two of the three results contain a head

691. Which trip has an average speed of exactly 50 km/h? Analysis

A. 60 km at 40 km/h, then 60 km at 60 km/h
B. 40 km at 40 km/h, then 40 km at 60 km/h
C. 1 hour at 40 km/h, then 1 hour at 60 km/h
D. 2 hours at 40 km/h, then 1 hour at 60 km/h

692. Shop A raises a price by 10% and later by another 10%. Shop B raises the same starting price once by 20%. Which statement is correct? Analysis

A. Both shops end at the same price, because 10% + 10% = 20%
B. Shop B ends higher, because one large rise beats two small ones
C. Shop A ends 1% of the original price higher, because its total rise is 21%
D. Shop A ends 2% higher, because each rise adds 1% extra

693. A student counts the diagonals of a hexagon as 6 × 3 = 18 (each vertex joins 3 non-adjacent vertices). What is the correct count, and what went wrong? Analysis

A. 18; the method is correct
B. 15; it is C(6, 2), every pair of vertices
C. 9; each diagonal was counted twice, once from each end
D. 12; only two diagonals leave each vertex

694. A 3-member committee is chosen from 8 people, but two particular people, X and Y, refuse to serve together. How many committees are possible? Analysis

A. 56
B. 20
C. 50
D. 36

695. 5 friends stand in a row for a photo. Two of them, Sara and Hina, must stand next to each other. How many arrangements are possible? Analysis

A. 24
B. 48
C. 120
D. 72

696. A box has 10 bulbs, of which 2 are faulty. 3 bulbs are picked at random without replacement. What is the probability that at least one is faulty? Analysis

A. 8/15
B. 7/15
C. 3/5
D. 1/5

697. A family has two children. You learn that at least one is a boy. Assuming boys and girls are equally likely and independent, what is the probability that both are boys? Analysis

A. 1/2
B. 1/4
C. 1/3
D. 2/3

698. In a group the ratio of boys to girls is 3:2. After 10 more girls join, the ratio becomes 1:1. How many people were in the group originally? Analysis

A. 25
B. 30
C. 60
D. 50

699. 1,000 units are invested for 3 years at 10% per year. Option A pays simple interest; option B compounds annually. How much more does option B pay? Analysis

A. 30
B. 31
C. 0
D. 331

Algorithmic Thinking MCQs for NSCT

700. Which statement best defines an algorithm? Understanding

A. Any program written in a high-level language
B. A diagram that shows the hardware of a computer
C. A finite sequence of precise steps that solves a problem and stops
D. A list of every possible input to a program

701. Which property of an algorithm requires it to stop after a finite number of steps? Understanding

A. Finiteness
B. Definiteness
C. Effectiveness
D. Generality

702. The step 'add a few of the numbers' violates which property of an algorithm? Understanding

A. Finiteness
B. Output
C. Input
D. Definiteness

703. What is a dry run? Understanding

A. Running a program without any input
B. Working through an algorithm by hand with sample values, recording what each step does
C. Compiling a program without executing it
D. Deleting unused variables from code

704. In a trace table, what do the columns and rows usually represent? Understanding

A. Columns are lines of code; rows are programmers
B. Columns are variables (and output); each row records their values after a step or iteration
C. Columns are inputs; rows are compile errors
D. Columns are loops; rows are functions

705. When swapping the values of two variables, why is a temporary variable normally used? Understanding

A. To make the program run faster
B. To keep one original value before it is overwritten
C. Because two variables cannot be compared directly
D. To free memory used by the first variable

706. In the step total = total + x inside a loop, what role does total play? Understanding

A. Accumulator
B. Counter
C. Flag
D. Sentinel

707. Top-down design (stepwise refinement) means: Understanding

A. Starting with the whole task and repeatedly breaking each step into smaller, more detailed steps
B. Writing the most detailed code first and grouping it later
C. Testing the program from the last line upwards
D. Designing only the user interface first

708. What is a loop invariant? Understanding

A. A variable whose value never changes inside the loop
B. The condition that makes the loop stop
C. A condition that is true before the loop starts and stays true after every iteration
D. The number of times the loop body executes

709. Which set of test inputs best checks the edge cases of a 'find the largest element' algorithm? Understanding

A. Three lists of five random positive numbers
B. An empty list, a one-element list and a list of all-negative numbers
C. One very long list of positive numbers
D. Lists that are already sorted in ascending order only

710. How do you normally argue that a loop terminates? Understanding

A. Show that the loop body contains no if statements
B. Run it once and see that it stops
C. Show that some quantity changes on every iteration towards a bound that ends the loop
D. Show that the loop variable is an integer

711. In bubble sort, a variable swapped is set to False before each pass and to True when a swap happens. What is it called and why is it used? Understanding

A. A counter; it counts the number of passes
B. A flag; if a pass makes no swaps, the list is sorted and the algorithm can stop early
C. An accumulator; it sums the swapped values
D. A sentinel; it marks the end of the list

712. Which three control structures are enough to express any algorithm, according to the structured programming theorem (Böhm and Jacopini)? Understanding

A. Input, output and storage
B. Sequence, selection and iteration
C. Recursion, arrays and pointers
D. Classes, objects and methods

713. An algorithm reads marks until the value −1 is entered. What is −1 called here? Understanding

A. A sentinel value
B. A flag
C. A counter
D. A loop invariant

714. In a loop-invariant proof (CLRS style), which part combines the invariant with the loop's exit condition to show the result is correct? Understanding

A. Initialisation
B. Maintenance
C. Termination
D. Induction base

715. An algorithm gives correct results in 20 dry runs. What can you conclude? Understanding

A. It is proved correct for all inputs
B. It must also be efficient
C. Any remaining bugs can only be syntax errors
D. It works for those inputs; testing can reveal bugs but cannot prove there are none

716. Dry-run this code. What does it print? Application

```
x = 5
y = 3
x = x + y
y = x - y
x = x - y
print(x, y)
```

A. 3 5
B. 5 3
C. 8 3
D. 8 5

717. What is the final value of total printed by this loop? Application

```
total = 0
for i in range(1, 5):
    total = total + i
print(total)
```

A. 10
B. 15
C. 4
D. 6

718. How many times does the loop body run (what is printed)? Application

```
i = 1
count = 0
while i < 100:
    i = i * 2
    count = count + 1
print(count)
```

A. 6
B. 7
C. 100
D. 50

719. How many times is m reassigned inside the loop? Application

```
a = [3, 8, 2, 9, 4]
m = a[0]
updates = 0
for x in a:
    if x > m:
        m = x
        updates = updates + 1
print(updates)
```

A. 1
B. 3
C. 5
D. 2

720. Trace n and s through this loop. What is printed? Application

```
n = 1234
s = 0
while n > 0:
    s = s + n % 10
    n = n // 10
print(s)
```

A. 1234
B. 4
C. 10
D. 24

721. This loop builds r from the digits of n. What is printed? Application

```
n = 1200
r = 0
while n > 0:
    r = r * 10 + n % 10
    n = n // 10
print(r)
```

A. 0021
B. 1200
C. 12
D. 21

722. This is Euclid's algorithm. What does it print? Application

```
a, b = 48, 18
steps = 0
while b != 0:
    a, b = b, a % b
    steps = steps + 1
print(a, steps)
```

A. 6 2
B. 18 3
C. 6 3
D. 6 4

723. How many times does the inner loop body run, as printed by c? Application

```
c = 0
for i in range(4):
    for j in range(i):
        c = c + 1
print(c)
```

A. 16
B. 10
C. 4
D. 6

724. This is one pass of bubble sort. What does it print? Application

```
a = [5, 1, 4, 2, 8]
for j in range(len(a) - 1):
    if a[j] > a[j + 1]:
        a[j], a[j + 1] = a[j + 1], a[j]
print(a)
```

A. [1, 4, 2, 5, 8]
B. [1, 2, 4, 5, 8]
C. [1, 5, 4, 2, 8]
D. [5, 1, 4, 2, 8]

725. Binary search looks for 23 in this sorted list. Which values are printed, in order? Application

```
a = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
low, high = 0, len(a) - 1
while low <= high:
    mid = (low + high) // 2
    print(a[mid])
    if a[mid] == 23:
        break
    elif a[mid] < 23:
        low = mid + 1
    else:
        high = mid - 1
```

A. 16, 23
B. 38, 23
C. 16, 38, 23
D. 16, 56, 23

726. An algorithm must print the average of N marks. Which step must come before 'divide by N'? Application

A. Add all N marks to a running total
B. Print the average
C. Sort the marks
D. Find the highest mark

727. This loop counts some multiples of 3. What does it print? Application

```
count = 0
for i in range(1, 16):
    if i % 3 == 0 and i % 5 == 0:
        pass
    elif i % 3 == 0:
        count = count + 1
print(count)
```

A. 5
B. 3
C. 4
D. 1

728. What is the value of x printed after this loop ends? Application

```
x = 10
while x > 0:
    x = x - 3
print(x)
```

A. 0
B. 1
C. -3
D. -2

729. This recursive function also counts its own calls. What is printed? Application

```
calls = 0
def f(n):
    global calls
    calls = calls + 1
    if n <= 1:
        return 1
    return f(n - 1) + f(n - 2)
print(f(5), calls)
```

A. 8 9
B. 5 15
C. 8 15
D. 8 8

730. This function should return the largest value in a non-empty list. Which input exposes its bug? Analysis

```
def largest(a):
    m = 0
    for x in a:
        if x > m:
            m = x
    return m
```

A. [3, 7, 1]
B. [0, 4, 4]
C. [10]
D. [-5, -2, -9]

731. This code should print the sum of the list (18) but prints 14. Which change fixes it? Analysis

```
a = [4, 7, 1, 6]
total = 0
for i in range(1, len(a)):
    total = total + a[i]
print(total)
```

A. Change range(1, len(a)) to range(len(a))
B. Change range(1, len(a)) to range(1, len(a) + 1)
C. Initialise total = 1
D. Change total + a[i] to total + a[i - 1]

732. contains(a, t) should return True if t appears anywhere in a. For which call does it return the WRONG result? Analysis

```
def contains(a, t):
    found = False
    for x in a:
        found = (x == t)
    return found
```

A. contains([1, 2, 3], 3)
B. contains([1, 2, 3], 5)
C. contains([1, 2, 3], 2)
D. contains([], 1)

733. What happens when this loop runs? Analysis

```
i = 0
while i != 10:
    i = i + 3
```

A. It stops when i equals 10
B. It never terminates, because i jumps from 9 to 12 and never equals 10
C. It runs exactly 3 times
D. It stops when i exceeds 10

734. A student tries to swap a and b. What does the code print? Analysis

```
a, b = 4, 7
a = b
b = a
print(a, b)
```

A. 7 7
B. 7 4
C. 4 4
D. 4 7

735. For this loop, which statement is a correct loop invariant at the start of each iteration? Analysis

```
a = [4, 7, 1, 6]
total = 0
k = 0
while k < len(a):
    total = total + a[k]
    k = k + 1
```

A. total equals the sum of a[0] to a[k]
B. total equals a[k]
C. k equals len(a)
D. total equals the sum of a[0] to a[k − 1]

736. To check a list of n items for duplicates, this code compares every pair once. How many comparisons does it make when n = 10? Analysis

```
n = 10
count = 0
for i in range(n):
    for j in range(i + 1, n):
        count = count + 1   # compare item i with item j
```

A. 100
B. 90
C. 10
D. 45

737. This function should return True only for prime numbers. Which call returns the WRONG result? Analysis

```
def is_prime(n):
    for d in range(2, n):
        if n % d == 0:
            return False
    return True
```

A. is_prime(2)
B. is_prime(1)
C. is_prime(9)
D. is_prime(13)

738. Which pair (a, b) is printed after the third iteration? Analysis

```
a, b = 1, 1
for _ in range(3):
    a, b = b, a + b
print(a, b)
```

A. 2 3
B. 3 5
C. 5 8
D. 3 4

739. is_pal is correct but does more work than needed. Which change keeps it correct and roughly halves the comparisons? Analysis

```
def is_pal(s):
    n = len(s)
    for i in range(n):
        if s[i] != s[n - 1 - i]:
            return False
    return True
```

A. Loop over range(n - 1) instead of range(n)
B. Compare s[i] with s[n - i] instead
C. Loop over range(n // 2) instead of range(n)
D. Return True as soon as one pair matches

740. According to Knuth's properties of an algorithm, which statement about input and output is correct? Understanding

A. It must take at least one input and may produce no output
B. It must take exactly one input and produce exactly one output
C. It needs no inputs or outputs as long as it terminates
D. It takes zero or more inputs and produces at least one output

741. What is the main purpose of writing an algorithm in pseudocode? Understanding

A. To make the algorithm run faster on a computer
B. To describe the steps clearly without being tied to the syntax of one programming language
C. To let the compiler find syntax errors early
D. To replace testing of the final program

742. In a standard flowchart, which symbol represents a decision such as 'Is x > 10?' Understanding

A. A rectangle
B. A diamond
C. A parallelogram
D. An oval

743. In the step count = count + 1, executed once each time a condition is met, what role does count play? Understanding

A. Accumulator
B. Counter
C. Flag
D. Sentinel

744. Which property of an algorithm requires every step to be basic enough to be carried out exactly, in finite time, with pencil and paper? Understanding

A. Finiteness
B. Definiteness
C. Effectiveness
D. Generality

745. An algorithm reads a rectangle's length and width and outputs its area. Which step is the 'process' part of input–process–output? Understanding

A. Read the length and the width
B. Display the area
C. Stop the algorithm
D. Multiply the length by the width

746. When should an algorithm use a selection (if/else) structure? Understanding

A. When it must repeat steps a number of times
B. When steps must run one after another in a fixed order
C. When it must choose between different steps depending on a condition
D. When it must store many values under one name

747. In algorithm design, what does abstraction mean? Understanding

A. Writing the program in as few lines as possible
B. Ignoring details that do not matter so you can focus on the essential parts of the problem
C. Repeating a step until a condition becomes false
D. Testing the algorithm with unusual inputs

748. What is a precondition of an algorithm or function? Understanding

A. A condition that is guaranteed to be true after it finishes
B. A condition that must be true before it runs for it to work correctly
C. A condition that stays true on every loop iteration
D. The first line of the algorithm

749. A program must keep asking for a password until the user types the correct one. Which kind of loop fits best? Understanding

A. A condition-controlled loop, because the number of repetitions is not known in advance
B. A count-controlled loop that runs exactly 3 times
C. No loop; a single if statement is enough
D. A nested loop, because there are two values to compare

750. How does a post-test loop (such as do…while or repeat…until) differ from a pre-test while loop? Understanding

A. Its body always runs at least once, because the condition is checked after the body
B. Its body may run zero times, because the condition is checked first
C. It can only be used with a counter variable
D. It never needs a stopping condition

751. A while loop runs forever. What is the most common cause? Understanding

A. The loop body contains an if statement
B. The loop variable was given a starting value
C. The loop uses the < operator instead of !=
D. Nothing inside the loop moves the controlling variable towards making the condition false

752. A function accepts whole-number marks from 1 to 100 inclusive. Which set of test inputs follows boundary value testing? Understanding

A. 25, 50 and 75
B. 1 and 100 only
C. −1000 and 1000
D. 0, 1, 100 and 101

753. Every correct recursive algorithm must include which of the following? Understanding

A. At least two recursive calls in each step
B. A global variable to store the result
C. A base case that is reached and returns without making a further recursive call
D. A loop inside the recursive function

754. In stepwise refinement, when should you stop breaking a step into smaller steps? Understanding

A. After exactly three levels of refinement
B. As soon as the first step has been written
C. When each step is simple enough to be written directly as code or carried out without further explanation
D. Only when the program has been fully tested

755. Why is the order of the tests important in if i < len(a) and a[i] > 0? Understanding

A. Python always evaluates both sides, so the order does not matter
B. Putting a[i] > 0 first makes the code faster and equally safe
C. The and operator checks the right-hand side first
D. The index is checked first, and short-circuit evaluation skips a[i] when i is out of range

756. In the worst case, how many comparisons does a linear search make on a list of n items? Understanding

A. n / 2
B. $\log_2 n$
C. 1
D. n

757. An outer loop runs n times and, on each of those runs, an inner loop runs m times. How many times does the inner loop body execute in total? Understanding

A. n × m
B. n + m
C. n
D. m

758. For a positive integer n, what do n % 10 and n // 10 give? Understanding

A. The first digit, and the number with its first digit removed
B. The number of digits, and the last digit
C. The last digit, and the number with its last digit removed
D. n divided by 10 as a decimal, and the remainder

759. In program correctness, what is the difference between partial and total correctness? Understanding

A. Partial correctness means it works on half of the test cases
B. Total correctness means it has been tested on every possible input
C. Total correctness is partial correctness plus a proof that the algorithm always terminates
D. They mean the same thing

760. What is a loop variant (also called a bound function) used for? Understanding

A. Proving the output is correct: it stays the same on every iteration
B. Proving termination: it is a non-negative integer quantity that strictly decreases on every iteration
C. Counting how many variables change inside the loop
D. Choosing between a for loop and a while loop

761. The halting problem (Turing, 1936) is undecidable. What does this mean for checking that loops terminate? Understanding

A. No loop can ever be proved to terminate
B. No single algorithm can decide termination for every program, so termination is argued for each loop specifically
C. Every loop terminates if it is run long enough
D. Compilers always reject programs with infinite loops

762. Which loop invariant is used to prove binary search correct on a sorted list a? Understanding

A. a[low] is always equal to the target
B. low and high always point to the first and last elements
C. The list a[low..high] is always unsorted
D. If the target is in a, it lies within a[low..high]

763. Dry-run this code. What does it print? Application

```
a = 4
b = a * 2
a = b - 1
print(a, b)
```

A. 7 8
B. 4 8
C. 7 4
D. 3 8

764. What does this loop print? Application

```
p = 1
for i in range(1, 5):
    p = p * i
print(p)
```

A. 10
B. 120
C. 24
D. 6

765. How many numbers does this code count? Application

```
nums = [3, 8, 6, 1, 4]
c = 0
for n in nums:
    if n % 2 == 0:
        c = c + 1
print(c)
```

A. 3
B. 2
C. 5
D. 18

766. Trace n and s. What is printed? Application

```
n = 5
s = ""
while n > 0:
    s = s + str(n)
    n = n - 2
print(s)
```

A. 531
B. 53
C. 5310
D. 135

767. What does this code print? Application

```
s = "code"
r = ""
for ch in s:
    r = ch + r
print(r)
```

A. code
B. edoc
C. c
D. e

768. What does this code print? Application

```
x = 10
y = 3
q = x // y
r = x % y
print(q, r)
```

A. 3.3333333333333335 1
B. 1 3
C. 3 0
D. 3 1

769. What does this code print? Application

```
total = 0
count = 0
for mark in [50, 70, 90]:
    total = total + mark
    count = count + 1
print(total / count)
```

A. 210
B. 3
C. 90
D. 70.0

770. What does this code print? Application

```
a = [6, 2, 9, 2, 5]
m = a[0]
pos = 0
for i in range(1, len(a)):
    if a[i] < m:
        m = a[i]
        pos = i
print(m, pos)
```

A. 2 3
B. 6 0
C. 2 4
D. 2 1

771. What does this code print? Application

```
a = [4, 9, 15, 7, 20]
i = 0
while i < len(a) and a[i] <= 10:
    i = i + 1
print(i)
```

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

772. This loop converts n to another form. What does it print? Application

```
n = 37
bits = ""
while n > 0:
    bits = str(n % 2) + bits
    n = n // 2
print(bits)
```

A. 101001
B. 100101
C. 37
D. 10010

773. What does this code print? Application

```
c = 0
for i in range(3):
    for j in range(5):
        if (i + j) % 2 == 0:
            c = c + 1
print(c)
```

A. 15
B. 7
C. 8
D. 6

774. This is the first two passes of insertion sort. What does it print? Application

```
a = [7, 3, 5, 1]
for i in range(1, 3):
    key = a[i]
    j = i - 1
    while j >= 0 and a[j] > key:
        a[j + 1] = a[j]
        j = j - 1
    a[j + 1] = key
print(a)
```

A. [1, 3, 5, 7]
B. [3, 7, 5, 1]
C. [3, 5, 7, 1]
D. [3, 5, 1, 7]

775. How many steps does this loop take before stopping? Application

```
n = 6
steps = 0
while n != 1:
    if n % 2 == 0:
        n = n // 2
    else:
        n = 3 * n + 1
    steps = steps + 1
print(steps)
```

A. 8
B. 7
C. 9
D. 6

776. What does this code print? Application

```
a = [5, 8, 2, 8, 6]
last = -1
for i in range(len(a)):
    if a[i] == 8:
        last = i
print(last)
```

A. 1
B. -1
C. 2
D. 3

777. What does this code print? Application

```
a = [3, 1, 4, 1, 5]
best = a[0]
out = []
for x in a:
    if x > best:
        best = x
    out.append(best)
print(out)
```

A. [3, 1, 4, 1, 5]
B. [3, 4, 5]
C. [5, 5, 5, 5, 5]
D. [3, 3, 4, 4, 5]

778. This finds the GCD by repeated subtraction. What does it print? Application

```
a, b = 20, 8
count = 0
while a != b:
    if a > b:
        a = a - b
    else:
        b = b - a
    count = count + 1
print(a, count)
```

A. 4 3
B. 4 2
C. 8 3
D. 2 3

779. Trace the recursion. What is printed? Application

```
def h(n):
    if n == 0:
        return ""
    return h(n // 2) + str(n % 2)
print(h(13))
```

A. 1011
B. 13
C. 1101
D. 110

780. What does this code print? Application

```
c = 0
for i in range(1, 5):
    for j in range(1, 5):
        if j > i:
            break
        c = c + j
print(c)
```

A. 40
B. 20
C. 10
D. 30

781. What does this code print? Application

```
a = [1, 2, 3]
b = a
b.append(4)
a = a + [5]
print(len(a), len(b))
```

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

782. This two-pointer loop searches a sorted list for a pair that sums to 13. What does it print? Application

```
a = [1, 3, 4, 6, 9]
i, j = 0, len(a) - 1
steps = 0
while i < j:
    s = a[i] + a[j]
    if s == 13:
        break
    elif s < 13:
        i = i + 1
    else:
        j = j - 1
    steps = steps + 1
print(a[i], a[j], steps)
```

A. 4 9 2
B. 4 9 3
C. 3 9 1
D. 1 9 0

783. The two assignments run one after the other, not at the same time. What is printed? Application

```
a, b = 2, 5
for _ in range(2):
    a = a + b
    b = a - b
print(a, b)
```

A. 9 7
B. 12 7
C. 9 2
D. 7 2

784. Which input makes this function crash? Analysis

```
def average(a):
    return sum(a) / len(a)
```

A. [5]
B. [0, 0, 3]
C. []
D. [1, 2]

785. This loop should print the numbers 1 to 10. What actually happens, and what is the fix? Analysis

```
for i in range(1, 10):
    print(i)
```

A. It prints 0 to 9; change the range to range(1, 10)
B. It prints 1 to 10; no change is needed
C. It prints 2 to 10; change the range to range(0, 10)
D. It prints 1 to 9; change the range to range(1, 11)

786. count_vowels should count the vowels in any English word. For which call does it return the WRONG answer? Analysis

```
def count_vowels(s):
    c = 0
    for ch in s:
        if ch in "aeiou":
            c = c + 1
    return c
```

A. count_vowels("banana")
B. count_vowels("sky")
C. count_vowels("Apple")
D. count_vowels("queue")

787. This code should print the sum of the list (12) but prints 6. What is the cause? Analysis

```
a = [2, 4, 6]
for x in a:
    total = 0
    total = total + x
print(total)
```

A. range is missing, so the loop skips the first element
B. The + operator joins the numbers as text
C. total is reset to 0 inside the loop, so only the last element is kept
D. The loop stops after the first element

788. In Euclid's loop below, which quantity proves that the loop terminates for positive integers a and b? Analysis

```
while b != 0:
    a, b = b, a % b
```

A. b, because it is a non-negative integer that strictly decreases on every iteration
B. a, because it strictly increases on every iteration
C. a + b, because it stays the same on every iteration
D. a % b, because it is always equal to 1

789. The programmer wanted to print every element of a = [10, 20, 30]. What happens? Analysis

```
a = [10, 20, 30]
i = 0
while i < len(a):
    i = i + 1
    print(a[i])
```

A. It prints 10, 20 and 30
B. It prints 10 and 20 only
C. It prints 20 and 30, then raises an IndexError
D. It loops forever

790. This code should print the average of the marks before the sentinel −1 (75). It prints about 49.67. What is wrong? Analysis

```
data = [70, 80, -1]
total, count = 0, 0
for x in data:
    total = total + x
    count = count + 1
    if x == -1:
        break
print(total / count)
```

A. The break should be continue
B. count should start at 1
C. total / count should use integer division
D. The sentinel is added to total and count before the check, so −1 is treated as a mark

791. Which condition is a loop invariant, true at the start of every iteration of this loop? Analysis

```
def power(x, n):
    result = 1
    i = 0
    while i < n:
        result = result * x
        i = i + 1
    return result
```

A. result == x ** n
B. result == x ** i
C. result == x ** (i + 1)
D. result == i * x

792. digits(n) should return how many digits a non-negative integer n has. For which call is the result WRONG? Analysis

```
def digits(n):
    c = 0
    while n > 0:
        n = n // 10
        c = c + 1
    return c
```

A. digits(7)
B. digits(0)
C. digits(10)
D. digits(999)

793. This binary search has a bug: high = mid instead of high = mid − 1. Which call never terminates? Analysis

```
def bsearch(a, t):
    low, high = 0, len(a) - 1
    while low <= high:
        mid = (low + high) // 2
        if a[mid] == t:
            return mid
        elif a[mid] < t:
            low = mid + 1
        else:
            high = mid
    return -1
```

A. bsearch([1, 3, 5], 3)
B. bsearch([1, 3, 5], 5)
C. bsearch([1, 3, 5], 6)
D. bsearch([1, 3, 5], 0)

794. second_largest should return the second largest distinct value. For which input is the answer WRONG? Analysis

```
def second_largest(a):
    first = second = a[0]
    for x in a:
        if x > first:
            second = first
            first = x
        elif x > second:
            second = x
    return second
```

A. [9, 4, 7]
B. [4, 7, 9]
C. [4, 9, 7]
D. [1, 2]

795. Which loop is guaranteed to terminate for EVERY positive integer starting value of n? Analysis

A. while n > 0: n = n // 2
B. while n != 0: n = n - 2
C. while n != 10: n = n + 3
D. while n != 1: n = n // 2 if n % 2 == 0 else 3 * n + 1

796. is_sorted should return True only if the list is in non-decreasing order. For which call does it return the WRONG result? Analysis

```
def is_sorted(a):
    for i in range(len(a) - 1):
        if a[i] > a[i + 1]:
            return False
        else:
            return True
```

A. is_sorted([1, 3, 2])
B. is_sorted([1, 2, 3])
C. is_sorted([3, 1, 2])
D. is_sorted([2, 2, 5])

797. Three variables must be rotated so that a gets b's value, b gets c's value and c gets a's old value. Which sequence does this? Analysis

A. a = b; b = c; c = a
B. t = a; a = c; c = b; b = t
C. t = a; a = b; b = c; c = t
D. t = c; c = b; b = a; a = t

798. The programmer wants to remove every 2 from the list. What does the code print? Analysis

```
a = [1, 2, 2, 3]
for x in a:
    if x == 2:
        a.remove(x)
print(a)
```

A. [1, 2, 3]
B. [1, 3]
C. [1, 2, 2, 3]
D. [3]

Critical Thinking & Decision Making MCQs for N...

799. In an argument, which word most often signals the conclusion? Understanding

A. Because
B. Since
C. Therefore
D. Given that

800. In critical thinking, what is an assumption in an argument? Understanding

A. The final claim the argument is trying to prove
B. A fact that has been independently verified
C. An unstated premise the conclusion depends on
D. A counter-argument raised by an opponent

801. A deductive argument is valid when: Understanding

A. The conclusion must be true if all the premises are true
B. All of its premises are actually true
C. Most people agree with its conclusion
D. Its conclusion is probably true

802. What makes a deductive argument sound? Understanding

A. It is valid, whether or not its premises are true
B. It is valid and all its premises are true
C. Its conclusion is popular
D. It has at least three premises

803. 'Ice-cream sales and drowning cases both rise in summer, so ice cream causes drowning.' Which error is this? Understanding

A. Treating correlation as causation
B. Ad hominem
C. Circular reasoning
D. Appeal to authority

804. Which fallacy attacks the person making an argument instead of the argument itself? Understanding
    A. Straw man
    B. False dilemma
    C. Slippery slope
    D. Ad hominem

805. A team lead says 'we should add tests to the payment module', and a colleague replies 'so you want us to stop shipping features and test everything forever?' What fallacy is this? Understanding
    A. Ad hominem
    B. Hasty generalisation
    C. Straw man
    D. Appeal to popularity

806. 'Either we rewrite the whole system in a new language, or we accept that it will never be fast.' What is the main flaw? Understanding
    A. Circular reasoning
    B. Appeal to authority
    C. Post hoc reasoning
    D. False dilemma: it presents two options when others exist

807. A student uses two Android phones that both overheat and concludes 'all Android phones overheat.' What is this? Understanding
    A. Hasty generalisation
    B. Straw man
    C. Ad hominem
    D. False dilemma

808. What is the sunk cost fallacy? Understanding
    A. Refusing to spend any money on a project
    B. Choosing the cheapest option every time
    C. Ignoring the cost of future maintenance
    D. Continuing with a choice because of past, unrecoverable investment rather than future costs and benefits

809. You choose to spend Saturday on a paid freelance task instead of revising for an exam. In decision-making terms, what is the lost revision time? Understanding
    A. A sunk cost
    B. A fixed cost
    C. The opportunity cost of the freelance task
    D. A dominated option

810. Caching results to make a program faster while using more memory is an example of: Understanding
    A. A sunk cost
    B. A logical fallacy
    C. A circular argument
    D. A trade-off

811. A developer believes a library is buggy, then notices only the bug reports and ignores the many successful uses. Which bias is this? Understanding
    A. Confirmation bias
    B. Anchoring bias
    C. Survivorship bias
    D. Sunk cost bias

812. How is a good inductive argument described, compared with a valid deductive one? Understanding

A. It is valid: its conclusion is guaranteed
B. It is circular: the conclusion is assumed in the premises
C. It is strong: its premises make the conclusion probable, not certain
D. It is unsound: its premises are false

813. 'If the server is down, the site shows an error. The site shows an error. Therefore the server is down.' Why is this invalid? Understanding

A. It denies the antecedent
B. It is modus tollens, which is always invalid
C. It has a false premise
D. It affirms the consequent: the error could have another cause

814. When comparing options on several criteria, when can an option be eliminated without choosing any weights? Understanding

A. When another option is at least as good on every criterion and strictly better on at least one
B. When it is the most expensive option
C. When it scores lowest on the single most important criterion
D. When it has the fewest features

815. 'Our app crashes on older phones. Most of our users in rural areas have older phones. So we should fix the crash before adding new features.' What is the conclusion? Application

A. Our app crashes on older phones
B. Most rural users have older phones
C. Older phones are slow
D. We should fix the crash before adding new features

816. 'Everyone in our batch uses this framework, so it must be the best one for our FYP.' Which fallacy is this? Application

A. Straw man
B. Slippery slope
C. Appeal to popularity (bandwagon)
D. Circular reasoning

817. Two laptops are scored out of 10. Weights: price 0.5, battery 0.3, weight 0.2. Laptop A scores (6, 9, 7) and Laptop B scores (8, 6, 8). Which wins, and with what weighted score? Application

A. A, 7.1
B. B, 7.4
C. A, 7.4
D. B, 7.1

818. Project X has a 60% chance of earning 100 (thousand units) and a 40% chance of losing 20. What is its expected value? Application

A. 60
B. 52
C. 80
D. 40

819. 'Our website is slow, so we need a bigger server.' Which assumption does this argument depend on? Application

A. Bigger servers are always cheaper
B. Users do not care about speed
C. The website was fast last year
D. The slowness is caused by a lack of server capacity

820. A team claims a new caching layer cut page load time. Which evidence would most strengthen this claim? Application

A. An A/B test in which only the cached group's load times fell, with traffic otherwise equal
B. The team that built the cache says it works
C. Load times fell during a public holiday when traffic was low
D. Several developers feel the site is faster

821. 'Students who use our revision app score higher in exams, so the app improves scores.' Which fact most weakens this? Application

A. The app is available on Android and iOS
B. Mostly students who were already top performers chose to use the app
C. The app has thousands of downloads
D. Some app users also attend tuition

822. 'If we allow one late submission, soon everyone will submit late and deadlines will mean nothing.' Which fallacy is this? Application

A. Slippery slope
B. Ad hominem
C. False dilemma
D. Hasty generalisation

823. Your team spent 6 months on a custom library for its FYP. A mature open-source library now does the same job and would save 2 months. Which reasoning is sound? Application

A. Keep the custom library because 6 months would otherwise be wasted
B. Switch only if the open-source library is also 6 months old
C. Compare only the future time and quality of each option; the 6 months are spent either way
D. Average the two options' past costs

824. A startup idea has a 50% chance of a gain of 40, a 30% chance of a gain of 10 and a 20% chance of a loss of 30 (thousand units). What is the expected value? Application

A. 29
B. 17
C. 6.67
D. 20

825. An app needs fast access to the i-th element and rarely inserts in the middle. Which choice best fits the trade-off? Application

A. A linked list, because insertion is O(1) at a known node
B. A linked list, because it uses less memory per element
C. A stack, because it supports random access
D. An array, because indexing is O(1) and middle inserts are rare

826. A product has fixed costs of 50,000 units, sells for 500 per item and costs 300 per item to make. How many items must be sold to break even? Application

A. 250
B. 100
C. 167
D. 500

827. 'This source is reliable because it says it is reliable.' What is wrong with this reasoning? Application

A. It is a straw man
B. It is circular: the conclusion is used as its own premise
C. It is a false dilemma
D. It is a hasty generalisation

828. 'All CS students take DSA. Sara takes DSA. So Sara is a CS student.' How should this argument be classified? Application

A. Valid and sound
B. Invalid: other students could also take DSA
C. Valid but unsound
D. Inductive and strong

829. 'Several famous tech founders dropped out of university, so dropping out increases your chance of success.' What is the main flaw? Analysis

A. Ad hominem against the founders
B. A false dilemma between studying and working
C. Circular reasoning
D. Survivorship bias: it ignores the many dropouts who did not succeed

830. Bug reports fell from 50 a month to 30 after code reviews began, and the manager says reviews cut bugs by 40%. In the same period active users fell from 10,000 to 5,000. What happened to bug reports per 1,000 users? Analysis

A. They fell from 5 to 3
B. They rose from 5 to 6
C. They fell by 40%
D. They stayed at 5

831. Algorithm A passes 90 of 100 easy tests and 10 of 50 hard tests. Algorithm B passes 170 of 200 easy tests and 5 of 50 hard tests. Which conclusion is justified? Analysis

A. B is better because its overall pass rate is higher
B. A and B perform equally
C. A has the higher pass rate in both groups; B's higher overall rate comes from being tested on more easy cases
D. B is better on hard tests

832. Laptop A scores (price 6, battery 9, weight 7) and B scores (8, 6, 8). If the weights change to price 0.2, battery 0.5, weight 0.3, what happens? Analysis

A. B still wins, 7.4 to 7.1
B. A wins, 7.8 to 7.0
C. B wins, 7.8 to 7.0
D. They tie at 7.4

833. Premises: 'Every deployment on a Friday needs manager approval.' and 'Deployment X did not get manager approval.' Which conclusion follows necessarily? Analysis

A. Deployment X was not made on a Friday
B. Deployment X was made on a Friday
C. Deployment X failed
D. Deployments on other days do not need approval

834. Three hosting vendors: V1 costs 100 with 99.9% uptime and 24/7 support; V2 costs 120 with 99.5% uptime and business-hours support; V3 costs 90 with 99.5% uptime and 24/7 support. Which can be eliminated before choosing any weights? Analysis

A. V3 only
B. V1 and V2
C. None of them can be eliminated
D. V2 only

835. 'Since the new hire joined, we have missed two deadlines, so the new hire is causing delays.' Which is the best critique? Analysis

A. It attacks the new hire's character
B. It generalises from too many examples
C. It assumes that because one event followed another, it caused it, without ruling out other causes
D. It is valid because the timing matches

836. Option A pays 40 for certain. Option B pays 80 with probability 0.5 and 0 otherwise. Which statement is correct? Analysis

A. Both have an expected value of 40; the choice depends on how much risk the decision-maker accepts
B. B is better because its best outcome is higher
C. A is better because its expected value is higher
D. B's expected value is 80

837. 'Passing the final project is necessary to graduate.' Which statement must be true? Analysis

A. Anyone who passes the final project will graduate
B. Anyone who graduated passed the final project
C. Passing the final project is enough to graduate
D. Students who fail the project can still graduate

838. 'Algorithm X is O(n log n) and Y is $O(n^2)$, so X will always be faster on our data.' X actually takes $100 \cdot n \cdot \log_2 n$ steps and Y takes n^2 steps. For n = 100, which is true? Analysis

A. X is faster: about 664 steps against 10,000 for Y
B. They take the same number of steps
C. Y is faster: about 10,000 steps against about 66,439 for X
D. X is faster because O(n log n) is always better

839. In the argument 'We should cache this query, since it runs on every page load', what role does the word 'since' play? Understanding

A. It introduces a premise (a reason)
B. It marks an unstated assumption
C. It introduces the conclusion
D. It signals a counter-argument

840. 'If the tests pass, the build is released. The tests pass. Therefore the build is released.' Which valid argument form is this? Understanding

A. Denying the antecedent
B. Modus ponens
C. Modus tollens
D. Affirming the consequent

841. 'If a student cheats, they fail. Hamza did not cheat. Therefore Hamza did not fail.' Why is this argument invalid? Understanding

A. It is a valid use of modus tollens
B. It is circular, because the conclusion repeats a premise
C. It affirms the consequent: failing has many causes
D. It denies the antecedent: Hamza could fail for another reason

842. 'A famous cricketer says this laptop is the best for programming, so it must be.' Which fallacy is this? Understanding

A. Appeal to an irrelevant authority
B. Straw man
C. Slippery slope
D. Ad hominem

843. In a meeting about a slow database, someone says 'Let's not forget how much the cloud bill went up last year.' The bill is unrelated to the slowness. What is this? Understanding

A. Begging the question
B. A false dilemma
C. A red herring
D. A hasty generalisation

844. A reviewer points out a bug in Bilal's code. Bilal replies, 'Your code had bugs last month too.' Which fallacy is this? Understanding

A. Tu quoque (appeal to hypocrisy)
B. Post hoc
C. Appeal to popularity
D. Straw man

845. 'A bug is a kind of insect. Our code is full of bugs. So our code needs pest control.' What flaw does this show? Understanding

A. Equivocation: one word is used in two different senses
B. Appeal to tradition
C. Affirming the consequent
D. Hasty generalisation from a small sample

846. A client's first remark is 'this should take about two weeks', and the team's own estimate then lands close to two weeks without much analysis. Which bias is most likely at work? Understanding

A. Survivorship bias
B. Sunk cost fallacy
C. Anchoring
D. Bandwagon effect

847. After reading news about a major data breach, a manager rates breaches as the company's biggest risk, though the logs show outages are far more frequent. Which bias is this? Understanding

A. Availability heuristic
B. Anchoring
C. Confirmation bias
D. Planning fallacy

848. A developer estimates a feature will take 3 days. Similar features have always taken him 6 to 8 days, but he believes 'this one is simpler'. Which bias does this show? Understanding

A. Planning fallacy
B. Bandwagon effect
C. Sunk cost fallacy
D. Survivorship bias

849. A project team agrees quickly on a design because nobody wants to disagree with the lead, and no alternatives are discussed. What is this called? Understanding

A. Satisficing
B. Brainstorming
C. Opportunity cost
D. Groupthink

850. Which of these statements is a fact rather than an opinion? Understanding

A. Java code is ugly
B. Python is the best first programming language
C. Python was first released in 1991
D. Every student should learn C before Python

851. 'If a number is divisible by 4, it is even.' For a number being even, what is 'divisible by 4'? Understanding

A. A necessary condition
B. A sufficient condition
C. Unrelated to being even
D. Both necessary and sufficient

852. Which statement is logically equivalent to 'If the input is valid, the request succeeds'? Understanding

A. If the request succeeds, the input is valid
B. The input is valid and the request succeeds
C. If the request does not succeed, the input is not valid
D. If the input is not valid, the request does not succeed

853. A team stops looking at hosting options once it finds one that meets every minimum requirement, rather than searching for the best possible one. What is this strategy called? Understanding

A. Satisficing
B. Anchoring
C. Groupthink
D. Maximising

854. A study finds that developers who drink more coffee make more commits. Both heavy coffee drinking and frequent commits are more common among night-shift workers. In this study, what is night-shift work? Understanding

A. A sufficient condition for coffee drinking
B. Evidence that coffee causes commits
C. A possible confounding variable
D. The conclusion of the study

855. Which of these claims is falsifiable, meaning some observation could show it to be wrong? Understanding

A. Our code is elegant in a deep sense
B. Some unknown bug might exist somewhere in the system
C. The new index cuts this query's average time below 50 ms
D. This framework will feel right to good developers

856. What does a cost-benefit analysis do when deciding between options? Understanding

A. It lists the options in the order they were suggested
B. It compares each option's expected benefits with its costs, in common units where possible
C. It rejects any option that carries a risk
D. It picks the option with the lowest cost, whatever its benefits

857. In a decision tree, what does a chance node (usually drawn as a circle) represent? Understanding

A. A sunk cost that must be ignored
B. The final payoff of a path
C. A point where an uncertain event decides the outcome, with a probability on each branch
D. A point where the decision-maker chooses between options

858. Two explanations fit the facts equally well: a user mistyped a password, or attackers are running a hidden campaign against one account. Which principle favours the first? Understanding

A. The sunk cost fallacy
B. Appeal to popularity
C. Occam's razor: prefer the explanation with fewer assumptions
D. Confirmation bias

859. You want to know the exact rules for a programming language feature. Which source is the most reliable? Understanding

A. An anonymous forum answer with no example
B. The language's official specification or documentation
C. A comment on a social-media post
D. A friend's memory of a lecture

860. In decision theory, what distinguishes a decision under risk from a decision under uncertainty? Understanding

A. Under risk, the probabilities of outcomes are known; under uncertainty they are not
B. There is no difference; the terms are synonyms
C. Under risk, there is only one option; under uncertainty there are several
D. Under risk, losses are possible; under uncertainty only gains are

861. A patch is described two ways: '90% of servers stay online' and '10% of servers go offline'. People approve the first wording far more often. Which effect is this? Understanding

A. Survivorship bias
B. Framing effect
C. Regression to the mean
D. Anchoring

862. 'A login succeeds if and only if the password matches.' Which statement captures this exactly? Understanding

A. A matching password is necessary but not sufficient
B. A successful login implies nothing about the password
C. A matching password is sufficient but not necessary
D. A matching password is both necessary and sufficient for a successful login

863. 'Because the old server is out of warranty and repairs now cost more than a new one, we should replace it.' Which part is a premise? Application

A. The new server will never fail
B. Replacing the server is the only option
C. We should replace the server
D. Repairs now cost more than a new server

864. 'Every module passed its unit tests, so the whole system will work correctly.' Which flaw is this? Application

A. Ad hominem
B. Fallacy of composition: what is true of each part is assumed true of the whole
C. Appeal to popularity
D. Sunk cost fallacy

865. A bug-bounty submission has a 25% chance of being accepted for a reward of 200 and earns nothing otherwise. What is its expected value? Application

A. 200
B. 25
C. 150
D. 50

866. 'If the cache is cold, the first request is slow. If the first request is slow, the health check times out.' What follows? Application

A. If the cache is cold, the health check times out
B. If the health check times out, the cache is cold
C. If the first request is fast, the cache is warm
D. The cache is cold

867. A framework scores 7 for performance, 5 for documentation and 10 for community, with weights 0.5, 0.3 and 0.2. What is its weighted score? Application

A. 7.0
B. 7.3
C. 6.5
D. 22

868. Vendor X charges 500 with a 10% chance of a 2,000 penalty for downtime. Vendor Y charges 600 with a 2% chance of a 3,000 penalty. Judged on expected cost, which is cheaper? Application

A. X: its fixed price of 500 is lower
B. They are equal at 700
C. Y: 660 against 700 for X
D. Y: 600 against 700, because its penalty can be ignored

869. An automation tool costs 120,000 up front, saves 15,000 a month in labour and costs 3,000 a month to maintain. After how many months does it pay for itself? Application

A. 40
B. 10
C. 8
D. 6.7

870. A graduate has three offers paying 80,000, 75,000 and 70,000 a year and chooses the 80,000 job. In money terms, what is the opportunity cost of that choice? Application

A. 0, because the best-paid job was chosen
B. 145,000, the sum of the offers given up
C. 75,000, the value of the best option given up
D. 70,000, the lowest offer

871. 'A rival app added dark mode and its users grew 30%, so adding dark mode will grow our users by 30%.' Which assumption does the argument rely on? Application

A. Our app already has dark mode
B. The rival's growth figure is too low
C. Users dislike dark mode
D. Dark mode caused the rival's growth, and it would affect our users the same way

872. 'Since the team went remote, commits per week rose 20%, so remote work made the team more productive.' Which fact most weakens this? Application

A. The team's office had poor Wi-Fi
B. Remote work is popular in many companies
C. At the same time, the team adopted a rule to make smaller, more frequent commits
D. The team's manager prefers remote work

873. 'We should hire Ayesha because she scored highest on our coding test.' Which assumption must hold for this argument to work? Application

A. Coding tests are used by most companies
B. Ayesha has never taken a coding test before
C. Scores on the coding test predict how well someone will do the job
D. The other candidates did not prepare

874. 'Teams that use more monitoring tools report more incidents, so monitoring tools cause incidents.' What is the most likely flaw? Application

    A. Reverse causation: teams with many incidents adopt more monitoring, and monitoring also detects more incidents
    B. Circular reasoning
    C. Ad hominem against the teams
    D. A false dilemma between monitoring and no monitoring

875. Three laptops score 8.2, 7.9 and 7.5 on a weighted matrix. The budget is a hard limit of 100,000, and they cost 115,000, 95,000 and 90,000 in that order. Which should be chosen? Application

    A. The 7.5 laptop, because it is the cheapest
    B. The 7.9 laptop: the highest score among options within budget
    C. The 8.2 laptop, because it has the highest score
    D. None, because the top scorer is over budget

876. A release note says 'crashes are cut by 50%': the crash rate fell from 0.2% of sessions to 0.1%. What is the absolute reduction? Application

    A. 0.2 percentage points
    B. 0.1 percentage points
    C. 1 percentage point
    D. 50 percentage points

877. Which survey gives the most trustworthy picture of how all students at a university rate its LMS? Application

    A. 5,000 votes in an open online poll
    B. A poll of the computing society's members
    C. Interviews with the 30 students who complained most
    D. 400 students chosen at random from the enrolment list

878. A coding contest costs 10 to enter. One entrant in five wins a prize of 40; the rest get nothing. What is the expected value of entering? Application

    A. 8
    B. 30
    C. 6
    D. -2

879. 10% of emails are spam. A filter flags 95% of spam and also wrongly flags 5% of genuine emails. If an email is flagged, about how likely is it to be spam? Application

    A. 81%
    B. 68%
    C. 10%
    D. 95%

880. Launching now: 30% chance of a gain of 100, 70% chance of a loss of 40. Or build a prototype first for 10: it reveals with certainty whether the product will succeed, and you launch only if it will. Which has the higher expected value? Application

    A. Prototype first: EV 30 against 2
    B. Launch now: both have EV 2, and it is faster
    C. Launch now: EV 30 against 20
    D. Prototype first: EV 20 against 2

881. Weights: cost 0.4, performance 0.35, support 0.25. Scores out of 10: C (9, 5, 6), D (6, 8, 7), E (7, 7, 6). Which option wins? Application

A. C, because it is best on the heaviest criterion
B. D, with 6.95
C. E, with 6.75
D. C, with 6.85

882. Payoffs (low demand, high demand): Small server (50, 60), Medium (40, 100), Large (20, 120). Demand probabilities are unknown. Which server does a cautious manager using the maximin rule pick? Application

A. Small, because its worst case of 50 is the best worst case
B. Large, because its best case of 120 is highest
C. Medium, because it has the smallest maximum regret
D. Medium, because its average payoff is highest

883. Same table: Small (50, 60), Medium (40, 100), Large (20, 120). Which option does the minimax regret rule choose? Application

A. Small, with a maximum regret of 60
B. Medium, with a maximum regret of 20
C. Large, with a maximum regret of 30
D. Small, with a maximum regret of 0

884. A team wants to know whether a new search feature is used. Which evidence is strongest? Analysis

A. A tweet from one user who loves it
B. The product manager's belief that it is popular
C. The fact that a competitor has a similar feature
D. Usage logs from 10,000 sessions showing how often the feature is opened

885. 'Bugs in module A doubled, from 2 to 4, while bugs in module B rose only 10%, from 50 to 55. So A's quality is falling faster and needs attention first.' What does the argument overlook? Analysis

A. Percentages can never be compared
B. B's bug count actually fell
C. A's figure is wrong: 2 to 4 is a 50% rise
D. B gained more new bugs (5) than A (2) and has far more bugs overall

886. Which of these arguments is valid? Analysis

A. If the key is wrong, decryption fails. Decryption failed. So the key is wrong.
B. Some keys are wrong. Some keys are old. So some old keys are wrong.
C. If the key is wrong, decryption fails. The key is not wrong. So decryption did not fail.
D. If the key is wrong, decryption fails. Decryption did not fail. So the key is not wrong.

887. A fraud model is '95% accurate' on a dataset where 95% of transactions are genuine. Which is the most important critique? Analysis

A. A model that labels every transaction genuine would also be 95% accurate, so accuracy alone shows nothing here
B. The figure must be wrong, because no model reaches 95%
C. Accuracy is the best measure when classes are imbalanced
D. 95% accuracy proves the model catches most fraud

888. A company starts rewarding developers for lines of code written. Six months later code volume has doubled, but releases are slower. What is the best explanation? Analysis
    A. Once lines of code became a target, developers wrote more verbose code, so the measure stopped reflecting useful output
    B. The developers became less skilled
    C. The data must be wrong, because more code means more features
    D. Releasing code is not related to writing code

889. Option A pays 100 with probability p and 0 otherwise. Option B pays 45 for certain. The team estimates p = 0.5. Below what value of p would B have the higher expected value? Analysis
    A. 0.9
    B. 0.5
    C. 0.55
    D. 0.45

890. Students who attend extra tutoring score lower on average than students who do not. Someone concludes that tutoring harms results. What is the most likely explanation? Analysis
    A. Tutoring is assigned to students who are already struggling, so the groups differed before tutoring began
    B. Tutoring always lowers scores
    C. The tutored students were the top performers
    D. The scores were added up incorrectly

891. 'Our tests cover only 40% of the code. Untested code is where most bugs hide. So most of our future bugs will come from the untested 60%, and we should write tests for it first.' What role does 'most of our future bugs will come from the untested 60%' play? Analysis
    A. A premise taken as given without support
    B. The final conclusion
    C. An intermediate conclusion that supports the final conclusion
    D. An irrelevant aside

892. The 10 slowest servers last week were given a patch. This week their response times improved, and the team credits the patch. Other servers were not tracked. What is the strongest critique? Analysis
    A. The patch obviously worked, because times improved
    B. Only fast servers should have been patched
    C. Extreme results tend to move back towards the average by themselves, so a control group of unpatched slow servers is needed
    D. Response time is not a valid measure of performance

893. Landing page A converted 300 of 10,000 visitors (3%). Page B, tested for one hour, converted 2 of 20 visitors (10%). A manager says B 'more than triples conversions'. What is the best assessment? Analysis
    A. A is better because it produced more conversions in total
    B. B's sample is too small: one conversion more or less moves its rate by 5 points, so the result is not reliable
    C. The claim is sound, because 10% is more than three times 3%
    D. B is worse, because its rate is really 1%

894. A startup has 50 in reserve. Plan P: 90% chance of +30, 10% chance of −80. Plan Q: +15 for certain. Which assessment is soundest? Analysis
    A. P is better: its EV is 27 against 15
    B. Q must be chosen, because P's expected value is negative
    C. P must be chosen, because the higher expected value always wins
    D. P has the higher EV (19 against 15), but its 10% chance of a loss larger than the reserve could end the company, so Q is defensible

895. Premises: 'No untested code is deployed.' and 'Some deployed code has bugs.' Which conclusion follows necessarily? Analysis
    A. All buggy code is deployed
    B. No untested code has bugs
    C. All tested code is deployed
    D. Some tested code has bugs

896. Five benchmark runs. Algorithm A: 10, 10, 10, 10, 100 ms. Algorithm B: 20, 20, 20, 20, 20 ms. A report says 'B is faster on average, so B is better for every workload.' Which conclusion is justified? Analysis
    A. A and B are equivalent, because their medians are equal
    B. B's mean (20) beats A's (28), but A is faster in 4 of 5 runs; one outlier drives A's mean, so the choice depends on how much rare slow runs matter
    C. B is faster in every run
    D. A is better in every respect, because its median is lower

897. 'Every module that handles payments needs a security review. So the refund module needs a security review.' Which added premise makes the argument valid? Analysis
    A. Every module that needs a security review handles payments
    B. The refund module handles payments
    C. The refund module has had a security review before
    D. Some modules that handle payments are refund modules

898. A team chose a migration plan with an 80% estimated chance of success, based on solid data. It failed. The manager says 'that was a bad decision'. What is the best critique? Analysis
    A. The team should always choose plans with a 100% chance of success
    B. The manager is right, because any decision that fails is bad
    C. It judges the decision by its outcome; a sound decision can still turn out badly, so the reasoning and information at the time should be judged
    D. The estimate must have been 20%, since the plan failed

Debugging & Error Analysis MCQs for NSCT

899. A program runs to completion without any error message but prints the wrong total. What kind of error is this? Understanding
    A. Runtime error
    B. Logic error
    C. Syntax error
    D. Linker error

900. In Python, the line `if x > 0` is missing its colon. When is this error reported? Understanding
    A. Before the program starts running, as a SyntaxError
    B. Never; Python silently adds the colon
    C. Only when the if statement is reached, as a RuntimeError
    D. After the program ends, as a logic error

901. Which exception does Python raise when this line runs? Understanding

```
print(10 / 0)
```

A. TypeError
B. ZeroDivisionError
C. SyntaxError
D. ValueError

902. What is an off-by-one error? Understanding

A. Any error that appears only on the first run of a program
B. Using one variable where two are needed
C. A loop or index boundary that goes one step too far or stops one step too early
D. A rounding error in the last decimal place of a float

903. When a Python program crashes, what does the traceback (stack trace) show? Understanding

A. The values of all variables in the program
B. The chain of function calls active at the crash, with the file and line of each
C. A suggested fix for the bug
D. Every line the program has executed since it started

904. What does setting a breakpoint in a debugger do? Understanding

A. Makes the line run twice to check for consistency
B. Pauses execution at that line so you can inspect variables and step through the code
C. Stops the program permanently and reports an error
D. Deletes the line from the program

905. A student explains her code line by line to a rubber duck on her desk and spots the bug halfway through. What is this technique called? Understanding

A. Rubber duck debugging
B. Static analysis
C. Pair programming
D. Unit testing

906. A user reports that a form "sometimes" saves the wrong date. What should a developer do first? Understanding

A. Reproduce the bug reliably with a known input
B. Rewrite the date-handling module from scratch
C. Add try/except around the save call to hide the error
D. Close the report because it is not consistent

907. A feature that worked in version 1.2 stops working after an unrelated change in version 1.3. What is this kind of bug called? Understanding

A. A syntax error
B. A boundary value
C. A regression
D. A compile-time warning

908. A function accepts ages from 18 to 60 inclusive. Which set of test inputs best follows boundary value analysis? Understanding

A. 17, 18, 60, 61
B. 20, 30, 40, 50
C. 18, 19, 20, 21
D. 0, 100, 1000, -1

909. What does an IndexError in Python tell you? Understanding

A. The code used a list or string index outside the valid range
B. The program ran out of memory
C. Two values of incompatible types were combined
D. A variable was used before it was defined

910. Which statement about logic errors is correct? Understanding

A. They only occur in languages without type checking
B. They are caught by the parser before the program runs
C. They always crash the program with an exception
D. The compiler or interpreter cannot detect them, so you find them through testing and tracing

911. `git bisect` helps find the commit that introduced a bug. Which strategy does it use? Understanding

A. Linear search from the newest commit backwards
B. Random sampling of commits
C. Comparing every pair of commits
D. Binary search over the commit history

912. Adding a print statement to investigate a bug makes the bug disappear. What is this kind of bug usually called? Understanding

A. An off-by-one error
B. A syntax error
C. A Heisenbug
D. A regression

913. Why does this Python code print False, and what is the correct way to compare the values? Understanding

```
print(0.1 + 0.2 == 0.3)
```

A. Python rounds 0.3 down to 0; use the decimal point twice
B. The + operator concatenates the values as strings; convert them to int first
C. == compares object identity for floats; use the is operator instead
D. 0.1 and 0.2 cannot be stored exactly in binary floating point; compare with a tolerance such as math.isclose

914. What is the most common cause of an infinite while loop? Understanding

A. The loop variable is never updated so that the condition can become false
B. The loop is placed inside a function
C. The condition uses the < operator
D. The loop body contains a print statement

915. This function should return 1 + 2 + ... + n. What does it print for n = 5? Application

```
def total(n):
    s = 0
    for i in range(1, n):
        s += i
    return s

print(total(5))
```

A. 5
B. 15
C. 14
D. 10

916. What happens when this loop over a three-element list runs? Application

```
a = [3, 6, 9]
for i in range(len(a) + 1):
    print(a[i])
```

A. It prints 3, 6, 9 and 0
B. It prints 3, 6 and 9 and ends normally
C. It raises IndexError before printing anything
D. It prints 3, 6 and 9, then raises IndexError

917. The function should return the mean of the list. What does it print? Application

```
def mean(a):
    return sum(a) // len(a)

print(mean([1, 2, 2]))
```

A. 1.6666666666666667
B. 1.67
C. 2
D. 1

918. What do the two calls to add print? Application

```
def add(item, bucket=[]):
    bucket.append(item)
    return bucket

print(add(1))
print(add(2))
```

A. [1] then [2]
B. [1] then [1, 2]
C. [] then [1]
D. [1, 2] then [1, 2]

919. What does this JavaScript if statement print? Application

```
let x = 0;
if (x = 0) {
  console.log("zero");
} else {
  console.log("non-zero");
}
```

A. Nothing; it throws a SyntaxError
B. zero
C. non-zero
D. undefined

920. This function should return the largest value in a non-empty list. Which test input exposes its bug? Application

```
def find_max(a):
    m = 0
    for v in a:
        if v > m:
            m = v
    return m
```

A. [-4, -2, -7]
B. [7]
C. [0, 5, 5]
D. [3, 1, 2]

921. What does this code print after changing one cell of the grid? Application

```
a = [[0] * 2] * 2
a[0][0] = 1
print(a)
```

A. [[1, 1], [1, 1]]
B. [[1, 0], [0, 0]]
C. [[1, 0], [1, 0]]
D. [[1, 1], [0, 0]]

922. This function should count the vowels in a word. What does it print? Application

```
def vowels(word):
    n = 0
    for ch in word:
        if ch in "aeiou":
            n += 1
    return n

print(vowels("Education"))
```

A. 3
B. 9
C. 5
D. 4

923. A form reads a quantity as text and adds 3 to it. What does this JavaScript print? Application

```
let a = "5";
let b = 3;
console.log(a + b);
```

A. TypeError
B. 8
C. 53
D. NaN

924. How many times does the loop body run? Application

```
count = 0
for i in range(0, 10, 3):
    count += 1
print(count)
```

A. 3
B. 5
C. 4
D. 10

925. A 20 m fence needs a post at each end and one every 2 m in between. A program computes the number of posts as 20 // 2. By how much is it wrong? Application

A. It is two posts short: 10 instead of 12
B. It is one post short: 10 instead of 11
C. It is one post too many: 10 instead of 9
D. It is correct: 10 posts

926. Trace this code. What does it print? Application

```
x = 3
for i in range(3):
    x = x * 2 - i
print(x)
```

A. 20
B. 24
C. 11
D. 18

927. What happens when f(-1) is called? Application

```
def f(n):
    if n == 0:
        return 1
    return n * f(n - 1)

print(f(-1))
```

A. It prints 1
B. It prints 0
C. It prints -1
D. It raises RecursionError

928. What does this condition check print when x is 5? Application

```
x = 5
if x == 1 or 2:
    print("match")
else:
    print("no match")
```

A. Nothing
B. match
C. SyntaxError
D. no match

929. This binary search fails to find some values that are in the sorted list, for example 9 in [1, 3, 5, 7, 9]. What is the bug? Analysis

```
def bs(a, t):
    lo, hi = 0, len(a) - 1
    while lo < hi:
        mid = (lo + hi) // 2
        if a[mid] == t:
            return mid
        elif a[mid] < t:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1
```

A. mid should be computed as (lo + hi + 1) // 2
B. lo = mid + 1 should be lo = mid
C. return -1 should be return lo
D. The loop condition should be lo <= hi

930. The code should print the sum of every window of 2 adjacent values. What does it print, and why? Analysis

```
a = [2, 4, 6, 8]
k = 2
sums = [sum(a[i:i + k]) for i in range(len(a) - k)]
print(sums)
```

A. [6, 10, 14]: the code is correct
B. [6, 10, 14, 8]: the range goes one window too far
C. [6, 10]: the range stops one window early
D. IndexError: the slice runs past the end

931. average() passes its tests on [2, 4, 6], but production logs show a ZeroDivisionError from it. Which input most likely caused the crash? Analysis

```
def average(a):
    return sum(a) / len(a)
```

A. [0]
B. []
C. [0, 0, 0]
D. [-1, 1]

932. This code was meant to swap x and y. What does it print? Analysis

```
x, y = 3, 7
x = y
y = x
print(x, y)
```

A. 7 7
B. 3 3
C. 3 7
D. 7 3

933. The code should print each row's sum of the matrix: 3 and then 7. It prints 3 and then 10. What is the cause? Analysis

```
m = [[1, 2], [3, 4]]
total = 0
for row in m:
    for v in row:
        total += v
    print(total)
```

A. The inner loop skips the last element of each row
B. The matrix rows are aliases of the same list
C. print is inside the inner loop
D. total is set to 0 once, outside the outer loop, so it keeps the first row's sum

934. The code should remove every 2 from the list. What does it print? Analysis

```
a = [1, 2, 2, 3]
for x in a:
    if x == 2:
        a.remove(x)
print(a)
```

A. [1, 3]
B. [1, 2, 3]
C. [1, 2, 2, 3]
D. RuntimeError

935. What does this JavaScript print once all the timers have fired? Analysis

```
const out = [];
for (var i = 0; i < 3; i++) {
  setTimeout(() => {
    out.push(i);
    if (out.length === 3) console.log(out.join(" "));
  }, 0);
}
```

A. 0 1 2
B. undefined undefined undefined
C. 3 3 3
D. 0 0 0

936. This line raises TypeError: can only concatenate str (not "int") to str. Which change fixes it? Analysis

```
total = 42
print("Total: " + total)
```

A. Declare total as global before the print
B. Convert "Total: " to an int
C. Convert total to a string: "Total: " + str(total)
D. Change + to *

937. The loop should print the even numbers 2, 4, 6, 8 and 10. Which single change makes it correct? Analysis

```
i = 2
while i < 10:
    print(i)
    i += 2
```

A. Change i < 10 to i <= 10
B. Change i += 2 to i += 1
C. Move print(i) to after i += 2
D. Start i at 0 instead of 2

938. Wrong data appears at the end of a pipeline of 1,024 stages. You can inspect the output after any stage. Using bisection, what is the most checks needed to find the first faulty stage? Analysis

A. 512
B. 10
C. 11
D. 32

939. Which exception does Python raise when this code runs? Understanding

```
d = {"a": 1}
print(d["b"])
```

A. IndexError
B. ValueError
C. KeyError
D. NameError

940. Which exception does Python raise when this code runs? Understanding

```
n = int("abc")
print(n)
```

A. TypeError
B. ValueError
C. SyntaxError
D. KeyError

941. Which exception does Python raise when this code runs? Understanding

```
items = None
items.append(5)
```

A. ZeroDivisionError
B. AttributeError
C. NameError
D. IndexError

942. A Python function has one line indented with 4 spaces and the next line in the same block with 3. What happens? Understanding

A. A logic error occurs that only testing can find
B. The program runs, and the line with 3 spaces is skipped
C. Python reports an IndentationError before the program runs
D. Python silently re-indents the line

943. Paused on a line that calls compute(x), you want to go inside compute and watch it run line by line. Which debugger command do you use? Understanding

A. Step into
B. Continue
C. Step over
D. Step out

944. What is a watch expression in a debugger used for? Understanding

A. Preventing a variable from being changed
B. Timing how long the program takes to run
C. Showing the value of a variable or expression each time execution pauses
D. Automatically fixing the line that crashed

945. What happens when Python runs the statement assert count >= 0 and count is −1? Understanding

A. It sets count to 0
B. It prints a warning and continues
C. Nothing; asserts are only comments
D. It raises AssertionError

946. In Java, the line int x = "hello"; appears in a method that is never called. When is the error reported? Understanding

A. Never, because the method is unused
B. Only when the program exits
C. At compile time, before the program runs
D. Only if the method is called at run time

947. What does a linter such as pylint or ESLint do? Understanding

A. Compiles code into machine code
B. Runs the program and records every variable's value
C. Analyses source code without running it and flags likely mistakes and style problems
D. Deletes code that is never executed

948. Which standard logging level is meant for detailed information that is useful only when diagnosing a problem? Understanding

A. ERROR
B. CRITICAL
C. WARNING
D. DEBUG

949. A function accepts marks from 0 to 100. Testing one mark below 0, one from 0 to 100 and one above 100 is an example of which technique? Understanding

A. Boundary value analysis
B. Equivalence partitioning
C. Rubber duck debugging
D. Bisection

950. In C, an array is declared int a[5]; and the code reads a[5]. What does the C standard say happens? Understanding

A. The behaviour is undefined: it may print garbage, crash or seem to work
B. a[5] always returns 0
C. The compiler always refuses to compile it
D. The program always stops with an IndexError

951. In standard testing terms, a programmer types < instead of <=, the code then contains a defect, and a user sees a wrong total. Which sequence names these correctly? Understanding

A. Error (mistake), then fault (defect), then failure
B. Fault, then error, then failure
C. Failure, then fault, then error
D. Bug, then crash, then exception

952. Two threads both read a counter, add 1 and write it back. Sometimes the final count is lower than expected. What is this kind of bug called? Understanding

A. A race condition
B. A memory leak
C. A syntax error
D. An off-by-one error

953. A long-running server uses more and more memory each day until it crashes, though its workload stays the same. What is the most likely bug? Understanding

A. A memory leak: memory is allocated but never released
B. An off-by-one error in a loop
C. A division by zero
D. A syntax error in the startup code

954. In Java, which situation causes a NullPointerException? Understanding

A. Reading index 10 of a 5-element array
B. Converting the string "abc" to an int
C. Calling a method on a reference variable whose value is null
D. Dividing an integer by zero

955. When asking for help with a bug, what is a minimal reproducible example? Understanding

A. The whole project zipped up
B. A screenshot of the error message
C. A description of the bug with no code
D. The smallest complete piece of code that still shows the bug when run

956. Why is wrapping code in try: ... except: pass considered a bad debugging practice? Understanding

A. It only catches syntax errors
B. It is a syntax error in Python 3
C. It makes the program run slower in all cases
D. It hides every error, so the program carries on with bad data and the real cause is lost

957. In Python, how do == and is differ? Understanding

A. == compares values; is checks whether both names refer to the same object
B. == works only on numbers; is works only on strings
C. They are identical in every case
D. is compares values; == checks object identity

958. A bug appears every time a specific input is given, under the same conditions, and adding print statements does not change it. Which term describes it? Understanding

A. A regression
B. A Bohrbug
C. A race condition
D. A Heisenbug

959. A program passes all 200 of its tests. What can you conclude? Understanding

A. The program has no bugs
B. The program has no logic errors, though runtime errors are possible
C. The tests must be wrong, since all of them pass
D. The tested cases work, but untested inputs may still contain bugs

960. What does this JavaScript print? Understanding

```
console.log(2 ** 53 + 1);
```

A. It throws a RangeError
B. Infinity
C. 9007199254740992
D. 9007199254740993

961. This code should count how many times each word appears. What happens when it runs? Application

```
words = ["to", "be", "to"]
counts = {}
for w in words:
    counts[w] += 1
print(counts)
```

A. It prints {'to': 1, 'be': 1}
B. It prints {}
C. It raises KeyError on the first word
D. It prints {'to': 2, 'be': 1}

962. This code tries to capitalise the first letter of a name. What happens? Application

```
s = "hello"
s[0] = "H"
print(s)
```

A. It raises IndexError
B. It raises TypeError, because strings cannot be changed in place
C. It prints hello
D. It prints Hello

963. This countdown should print 5 4 3 2 1 0. What does it print? Application

```
for i in range(5, 0, -1):
    print(i, end=" ")
```

A. 4 3 2 1 0
B. 5 4 3 2 1
C. Nothing
D. 5 4 3 2 1 0

964. What does this JavaScript print? Application

```
console.log("10" - 3);
```

A. 7
B. NaN
C. 103
D. "103"

965. This function should return True for even numbers. What does it print? Application

```
def is_even(n):
    return n % 2 == 1

print(is_even(4))
```

A. It raises TypeError
B. 0
C. False
D. True

966. What happens when this code runs? Application

```
list = [1, 2]
letters = list("abc")
print(letters)
```

A. It raises TypeError, because the name list now refers to a list, not the built-in
B. It raises NameError
C. It prints [1, 2]
D. It prints ['a', 'b', 'c']

967. This Java code should print 2.5. What does it print? Application

```
int a = 5, b = 2;
double r = a / b;
System.out.println(r);
```

A. 2.0
B. 2
C. 2.5
D. 3.0

968. This code should print the name in capitals. What does it print? Application

```
s = "abc"
s.upper()
print(s)
```

A. ABC
B. None
C. abc
D. It raises AttributeError

969. A copy of the list was meant to be changed while the original stayed the same. What does this code print? Application

```
a = [1, 2, 3]
b = a
b.append(4)
print(a)
```

A. [1, 2, 3]
B. It raises TypeError
C. [1, 2, 3, 4]
D. [4]

970. This function should report whether any number in the list is negative. What does it print? Application

```
def has_negative(a):
    for v in a:
        if v < 0:
            return True
        else:
            return False

print(has_negative([3, -1]))
```

A. It raises IndexError
B. False
C. None
D. True

971. This code should sort the list. What does it print? Application

```
a = [3, 1, 2]
a = a.sort()
print(a)
```

A. [1, 2, 3]
B. [3, 1, 2]
C. None
D. It raises TypeError

972. This function should return n factorial. What does it print for n = 4? Application

```
def fact(n):
    r = 1
    for i in range(n):
        r *= i
    return r

print(fact(4))
```

A. 1
B. 24
C. 6
D. 0

973. A grading function maps 0-49 to F, 50-69 to C and 70-100 to A. Which test set covers every valid equivalence class with one value each and nothing more? Application

A. 49, 50, 69, 70
B. 0, 100
C. 10, 20, 30
D. 25, 60, 85

974. The leap-year rule is: divisible by 4, except century years, which must be divisible by 400. Which input exposes the bug in this function? Application

```
def leap(y):
    return y % 4 == 0
```

A. 2024
B. 1900
C. 2023
D. 2000

975. This JavaScript should sort the numbers in ascending order. What does it print? Application

```
const a = [10, 9, 1];
a.sort();
console.log(a);
```

A. [9, 10, 1]
B. [1, 10, 9]
C. [1, 9, 10]
D. [10, 9, 1]

976. What does this JavaScript print? Application

```
console.log(0 == "", 0 === "");
```

A. false true
B. true false
C. true true
D. false false

977. A report rounds average scores. What does this Python 3 code print? Application

```
print(round(2.5), round(3.5))
```

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

978. Each function was meant to return its own index. What does this code print? Application

```
fs = [lambda: i for i in range(3)]
print([f() for f in fs])
```

A. It raises NameError
B. [2, 2, 2]
C. [0, 1, 2]
D. [0, 0, 0]

979. What does this code print? Application

```
found = 0
for i in range(3):
    for j in range(3):
        if j == 1:
            break
        found += 1
print(found)
```

A. 1
B. 6
C. 9
D. 3

980. Version numbers are read from a file as strings. This code should print the highest version. What does it print? Application

```
versions = ["9", "10", "100"]
print(max(versions))
```

A. 10
B. 9
C. It raises TypeError
D. 100

981. b was meant to be an independent copy of a. What does this code print? Application

```
a = [[1], [2]]
b = a.copy()
b[0].append(9)
print(a)
```

A. [[1, 9], [2]]
B. [[1], [2]]
C. [[1, 9], [2, 9]]
D. [[1], [2], 9]

982. People aged 18 and over count as adults. Which single test input is most likely to reveal the bug in this function? Analysis

```
def is_adult(age):
    return age > 18
```

A. 65
B. 5
C. 18
D. 30

983. This function should count how often a character appears. count('banana', 'b') returns 0. Why? Analysis

```
def count(s, c):
    n = 0
    for i in range(1, len(s)):
        if s[i] == c:
            n += 1
    return n

print(count("banana", "b"))
```

A. The comparison should use is instead of ==
B. The loop starts at index 1, so the first character is never checked
C. n is reset to 0 on every pass
D. range(len(s)) would skip the last character

984. This palindrome check is called with 'abba'. What happens, and what is the fix? Analysis

```
def is_pal(s):
    for i in range(len(s) // 2):
        if s[i] != s[len(s) - i]:
            return False
    return True

print(is_pal("abba"))
```

A. IndexError on the first pass; compare with s[len(s) - 1 - i]
B. IndexError; the loop should be range(len(s) // 2 + 1)
C. It returns True correctly
D. It returns False; the loop should run to len(s)

985. This program crashes with IndexError: list index out of range. The traceback lists main, then load, then parse. Where is the root cause? Analysis

```
def parse(row):
    return int(row[1])

def load(rows):
    return [parse(r) for r in rows]

def main():
    data = load([["a", "5"], ["b"]])
    print(sum(data))

main()
```

A. int() cannot convert the string "5"
B. The second row has one field, and parse reads row[1] without checking the row's length
C. sum cannot add a list of integers
D. main should not call load

986. This function should return the average of the positive numbers only. avg_pos([4, -2, 8]) returns 4.0 instead of 6.0. Which single change fixes it? Analysis

```
def avg_pos(a):
    total = 0
    for v in a:
        if v > 0:
            total += v
    return total / len(a)

print(avg_pos([4, -2, 8]))
```

A. Change v > 0 to v >= 0
B. Start total at 1 instead of 0
C. Use // instead of /
D. Divide by the number of positive values counted in the loop, not len(a)

987. Using git bisect, commit 1 is known good and commit 200 is bad. You test commit 100 (good), then commit 150 (bad). Which commit should be tested next? Analysis

A. 75
B. 175
C. 101
D. 125

988. A duplicate-check function gives correct results in all tests, but users report the page freezing with 50,000 items. It compares every item with every other item. What is the best diagnosis? Analysis

A. An off-by-one error in the inner loop
B. A syntax error that only appears with large inputs
C. A performance bug: about $n^2 / 2$ comparisons is over a billion for 50,000 items; use a set for O(n)
D. A logic error, because the results must be wrong

989. This function should count the digits of a non-negative integer. Which input exposes a bug, and why? Analysis

```
def digits(n):
    c = 0
    while n > 0:
        n //= 10
        c += 1
    return c
```

A. 9, because one-digit numbers return 2
B. 100, because trailing zeros are skipped
C. 12345, because large numbers overflow
D. 0, because the loop never runs and it returns 0 instead of 1

990. A dashboard suddenly shows a 0% conversion rate for one region, although sales there look normal. What is the most likely cause in this code? Analysis

```
def conversion(sales, visits):
    try:
        return sales / visits
    except:
        return 0
```

A. sales and visits are swapped
B. visits is 0 for that region, and the bare except turns the ZeroDivisionError into a silent 0
C. Python cannot divide integers to get a float
D. The division should use // instead of /

991. Why does this loop never end? Analysis

```
x = 0.0
while x != 1.0:
    x += 0.1
```

A. x is never updated inside the loop
B. The condition should use is not instead of !=
C. Repeated adding of 0.1 never gives exactly 1.0 in binary floating point, so x != 1.0 stays true
D. 0.1 is rounded to 0 on each addition

992. This bubble sort is called with [4, 3, 2, 1]. What does it return, and which change fixes it? Analysis

```
def bubble(a):
    n = len(a)
    for i in range(n):
        for j in range(n - i - 2):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
    return a

print(bubble([4, 3, 2, 1]))
```

A. [1, 2, 3, 4]; the code is correct
B. [3, 2, 1, 4]; the outer loop should be range(n - 1)
C. It raises IndexError; the bound should be n - i
D. [2, 3, 4, 1]; the inner loop bound should be n - i - 1

993. This code should remove users with no orders. It raises RuntimeError: dictionary changed size during iteration. Which fix is correct? Analysis

```
orders = {"ali": 3, "sara": 0, "omar": 0}
for name in orders:
    if orders[name] == 0:
        del orders[name]
print(orders)
```

A. Wrap the del line in try/except and ignore the error
B. Replace del orders[name] with orders[name] = None
C. Loop over orders.values() instead
D. Loop over a copy of the keys: for name in list(orders):

994. A developer expects [1, 2, 3]. What does this JavaScript actually print, and why? Analysis

```
console.log(['1', '2', '3'].map(parseInt));
```

A. [1, NaN, NaN]: map passes the index as parseInt's radix argument
B. [NaN, NaN, NaN]: parseInt cannot read strings in an array
C. [1, 2, 3]: parseInt converts each string
D. ['1', '2', '3']: map returns the strings unchanged

995. This digit-sum function works for positive numbers. What happens when it is called with -17 in Python? Analysis

```
def digit_sum(n):
    s = 0
    while n != 0:
        s += n % 10
        n //= 10
    return s
```

A. It never ends: n becomes -1, and -1 // 10 is still -1
B. It returns -8, as it would in C
C. It raises ValueError for negative input
D. It returns 8

996. Users say the first 10 products never appear and the last page is always empty. Pages are numbered from 1 and hold 10 items. What is wrong? Analysis

```
def get_page(items, page, size=10):
    return items[page * size:(page + 1) * size]

print(get_page(list(range(25)), 1))
```

A. The slice treats page as 0-based; page 1 shows items 10-19. It should be items[(page - 1) * size : page * size]
B. size should be 11
C. Python slices exclude the start index
D. The slice end should be (page + 1) * size + 1

Complexity & Efficiency Awareness MCQs for NSCT

997. What does Big-O notation describe about an algorithm? Understanding

A. The number of lines of code it contains
B. An upper bound on how its running time (or memory) grows as the input size grows
C. The smallest input it can handle
D. The exact number of seconds it takes on a given computer

998. An algorithm takes $3n^2 + 5n + 20$ steps. What is its Big-O complexity? Understanding

A. $O(3n^2)$
B. $O(n^2)$
C. $O(n^2 + n)$
D. $O(n)$

999. What is the time complexity of reading a[i] from an array by its index? Understanding

A. O(n)
B. O(i)
C. O(log n)
D. O(1)

1000. Which of these grows fastest as n becomes large? Understanding

A. $O(2^n)$
B. $O(n^3)$
C. $O(n^2)$
D. O(n log n)

1001. What is the worst-case time complexity of binary search on a sorted array of n elements? Understanding

A. O(n)
B. O(log n)
C. O(n log n)
D. O(1)

1002. What does $f(n) = \Theta(g(n))$ mean? Understanding

A. f grows strictly slower than g
B. f grows at the same rate as g: g is both an upper and a lower bound, up to constants
C. f is always smaller than g for every n
D. f is an upper bound on g

1003. What does $f(n) = \Omega(g(n))$ mean? Understanding

A. f is the average-case cost of g
B. f grows at least as fast as g for large n (a lower bound)
C. f and g are always equal
D. f grows no faster than g (an upper bound)

1004. What is the worst-case time complexity of linear search in an unsorted list of n items? Understanding

A. O(1)
B. O(n)
C. $O(n^2)$
D. O(log n)

1005. A loop adds up the n numbers in an array using one variable, total. What is its extra space complexity? Understanding

A. O(1)
B. O(n)
C. $O(n^2)$
D. O(log n)

1006. Which of these is an example of a time-space trade-off? Understanding

A. Caching computed results in a dictionary so repeated calls return instantly
B. Using a faster CPU
C. Renaming variables to shorter names
D. Removing comments from source code

1007. What are the average-case and worst-case lookup times for a hash table with n keys? Understanding

A. O(1) average and worst case
B. O(1) average, O(n) worst case
C. O(n) average, $O(n^2)$ worst case
D. O(log n) average, O(n) worst case

1008. Appending to a Python list is described as amortised O(1). What does amortised mean here? Understanding

A. Appends are O(1) only for lists under 1,000 items
B. The list never needs to grow its storage
C. Every single append takes exactly the same time
D. Occasional expensive resizes are spread over many cheap appends, so the average per append is constant

1009. Why is $O(\log_2 n)$ usually written simply as $O(\log n)$? Understanding

A. Because computers only use base-10 logarithms
B. Because the base is always e in algorithm analysis
C. Because log n is always smaller than 1
D. Logarithms in different bases differ only by a constant factor, which Big-O ignores

1010. What is the worst-case lower bound on the number of comparisons for any comparison-based sorting algorithm? Understanding

A. $\Omega(n)$
B. $\Omega(\log n)$
C. $\Omega(n \log n)$
D. $\Omega(n^2)$

1011. What is the best-case time complexity of insertion sort, and when does it happen? Understanding

A. $O(n \log n)$, when the input is random
B. $O(n^2)$, when the input is already sorted
C. $O(n)$, when the input is already sorted
D. $O(1)$, when the input has one element only

1012. What extra space does a standard merge sort of an array need? Understanding

A. $O(1)$
B. $O(n^2)$
C. $O(\log n)$ only
D. $O(n)$

1013. With the inner loop bounded by i, how many times does count += 1 run when n = 10? Application

```
n = 10
count = 0
for i in range(n):
    for j in range(n):
        count += 1
print(count)
```

A. 20
B. 100
C. 10
D. 55

1014. How many times does count += 1 run when n = 10? Application

```
n = 10
count = 0
for i in range(n):
    for j in range(i):
        count += 1
print(count)
```

A. 50
B. 55
C. 100
D. 45

1015. How many times does the loop body run when n = 64? Application

```
n = 64
i = 1
count = 0
while i < n:
    i *= 2
    count += 1
print(count)
```

A. 32
B. 7
C. 6
D. 64

1016. In this shrinking nested loop, how many times does count += 1 run for n = 16, and what is the growth rate? Application

```
n = 16
count = 0
for i in range(n):
    j = n
    while j > 1:
        j //= 2
        count += 1
print(count)
```

A. 80 times; O(n log n)
B. 256 times; $O(n^2)$
C. 64 times; O(n log n)
D. 16 times; O(n)

1017. The inner loop steps by i. For n = 12 the body runs 35 times. What is the growth rate of this code? Application

```
n = 12
count = 0
for i in range(1, n + 1):
    for j in range(i, n + 1, i):
        count += 1
print(count)
```

A. $O(n^2)$
B. O(n)
C. O(n log n)
D. $O(n\sqrt{n})$

1018. A function runs three separate (not nested) loops one after another, each over the same n items. What is its time complexity? Application

A. $O(3^n)$
B. O(n log n)
C. $O(n^3)$
D. O(n)

1019. An $O(n^2)$ algorithm takes 2 seconds for n = 1,000. Roughly how long should it take for n = 2,000? Application

A. 8 seconds
B. 16 seconds
C. 4 seconds
D. 2 seconds

1020. Algorithm A takes 100n steps and algorithm B takes n^2 steps. For which input sizes is A strictly faster? Application

A. All $n \geq 1$
B. No n, because A's constant is larger
C. $n < 100$
D. $n > 100$

1021. How many calls to fib are made in total when this code computes fib(5)? Application

```
calls = 0
def fib(n):
    global calls
    calls += 1
    if n < 2:
        return n
    return fib(n - 1) + fib(n - 2)

fib(5)
print(calls)
```

A. 5
B. 8
C. 31
D. 15

1022. How many times does count += 1 run when n = 16, and what is the growth rate? Application

```
n = 16
count = 0
i = n
while i > 0:
    for j in range(i):
        count += 1
    i //= 2
print(count)
```

A. 256 times; $O(n^2)$
B. 31 times; O(n log n)
C. 31 times; O(n)
D. 64 times; O(n log n)

1023. How many times does the loop body run? Application

```
count = 0
for i in range(1, 20, 3):
    count += 1
print(count)
```

A. 6
B. 10
C. 7
D. 20

1024. A duplicate check compares every pair of items once using the loops below. How many comparisons does it make for n = 100? Application

```
def has_duplicate(a):
    n = len(a)
    for i in range(n):
        for j in range(i + 1, n):
            if a[i] == a[j]:
                return True
    return False
```

A. 4,950
B. 5,050
C. 9,900
D. 10,000

1025. You must answer 1,000 membership queries on an unsorted list of 1,000,000 items. Which approach needs the least total work on average? Application

A. Compare every pair of list items first, then search
B. Build a set from the list once, then test each query against the set
C. Sort the list for each query, then binary search
D. Run a linear search of the list for each query

1026. In Python, what are the average time complexities of `x in my_list` and `x in my_set`? Application

A. O(1) for both
B. O(1) for the list, O(n) for the set
C. O(n) for the list, O(1) for the set
D. O(log n) for both

1027. A student says "two nested loops always mean $O(n^2)$". What is the actual complexity of this code? Analysis

```
def weighted(a):
    n = len(a)
    total = 0
    for i in range(n):
        for j in range(5):
            total += a[i] * j
    return total
```

A. O(n)
B. $O(n^2)$
C. O(log n)
D. $O(5n^2)$

1028. What is the time complexity of this loop? Analysis

```
def first_root_above(n):
    i = 1
    while i * i <= n:
        i += 1
    return i
```

A. O(√n)
B. O(log n)
C. O(n)
D. O(n^2)

1029. A divide-and-conquer algorithm splits the input into two halves, solves each recursively and merges the results in linear time: T(n) = 2T(n/2) + n. What is T(n)? Analysis

A. O(n)
B. O(log n)
C. O(n^2)
D. O(n log n)

1030. How many calls to f are made when f(4) is called, and what does that suggest about its growth? Analysis

```
def f(n):
    if n == 0:
        return
    f(n - 1)
    f(n - 1)
```

A. 16 calls; O(n^2)
B. 8 calls; O(n)
C. 5 calls; O(n)
D. 31 calls; O(2^n)

1031. Naive recursive fib(n) is exponential. After adding memoisation (a dictionary of computed results), what is its time complexity? Analysis

A. O(n^2)
B. O(n)
C. O(log n)
D. O(2^n)

1032. A machine performs about 10^8 simple operations per second. Roughly how long does an O(n^2) algorithm take for n = 10^6? Analysis

A. About 10,000 seconds (close to 3 hours)
B. About 1 second
C. About 100 seconds
D. About 0.01 seconds

1033. This code builds a list of n items by inserting each one at the front. What is the total time, and how can it be improved? Analysis

```
def build_reversed(data):
    result = []
    for x in data:
        result.insert(0, x)
    return result
```

A. $O(n^2)$; append to the end and reverse once, or use collections.deque
B. O(n); it is already optimal
C. O(1); insert is always constant time
D. O(n log n); sort the list afterwards

1034. A target is equally likely to be at any of n = 9 positions in a list. On average, how many comparisons does a successful linear search make? Analysis

A. 9
B. 1
C. 5
D. 4.5

1035. Algorithm X takes 1,000n steps, which is O(n). Algorithm Y takes n^2 steps, which is $O(n^2)$. Which is faster for n = 500? Analysis

A. X, because 1,000n is 500 steps
B. X, because O(n) is always faster than $O(n^2)$
C. Y, because 250,000 steps is fewer than 500,000
D. They take the same number of steps

1036. A recursive function sums a list of n numbers by adding the first element to the sum of the rest. Compared with a simple loop, what extra space does the recursive version use? Analysis

```
def rsum(a, i=0):
    if i == len(a):
        return 0
    return a[i] + rsum(a, i + 1)
```

A. O(n) for the call stack, versus O(1) for the loop
B. O(1), the same as the loop
C. O(log n) for the call stack
D. $O(n^2)$, because each call copies its list slice

1037. Which list orders these growth rates from slowest-growing to fastest-growing? Understanding

A. O(1), O(log n), O(n), O(n log n)
B. O(log n), O(1), O(n), O(n log n)
C. O(1), O(n), O(log n), O(n log n)
D. O(1), O(log n), O(n log n), O(n)

1038. An algorithm takes 7n + 3 log n + 12 steps. What is its Big-O complexity? Understanding

A. O(log n)
B. O(n)
C. O(n log n)
D. O(7n)

1039. What is the time complexity of pushing an item onto, or popping it from, the top of a stack? Understanding

A. O(log n)
B. O(n)
C. O(1)
D. $O(n^2)$

1040. What does the space complexity of an algorithm measure? Understanding

A. How many lines of code the algorithm contains
B. How much disk space the compiled program file takes
C. How many CPU cores the algorithm can use at once
D. How the extra memory it needs grows with the input size

1041. Which of these sorting algorithms runs in O(n log n) time even in the worst case? Understanding

A. Merge sort
B. Bubble sort
C. Selection sort
D. Insertion sort

1042. Why is a running time of 5n steps written simply as O(n)? Understanding

A. Because the constant 5 is always smaller than n
B. Big-O describes how time grows with n, and a constant factor does not change that growth rate
C. Because Big-O only counts loops and ignores arithmetic
D. Because every algorithm takes about 5 steps per item anyway

1043. For large n, which term dominates $n^3 + 1000n^2$? Understanding

A. $1000n^2$
B. Both terms count equally
C. n^3
D. Neither; the sum is O(1)

1044. In complexity analysis, what does n usually stand for? Understanding

A. The number of seconds the program runs
B. The number of variables declared in the code
C. The speed of the processor in operations per second
D. The size of the input, such as the number of items in a list

1045. When does linear search hit its worst case on a list of n items? Understanding

A. When the target is in the last position or not in the list at all
B. When the target is the first element
C. When the list is already sorted
D. When the list contains duplicate values

1046. Given only the head pointer of a singly linked list, how long does it take to reach the head node and the k-th node? Understanding

A. O(1) for both
B. O(1) for the head, O(k) for the k-th node
C. O(k) for both
D. O(log k) for the k-th node

1047. Given a pointer to the head, what is the time complexity of inserting a new node at the front of a singly linked list of n nodes? Understanding

A. O(n)
B. O(log n)
C. O(1)
D. $O(n^2)$

1048. For n = 1,000,000, roughly how do $n \log_2 n$ and n^2 compare? Understanding

A. About 10^6 versus 10^6
B. About 2×10^7 versus 2×10^7
C. About 10^{12} versus 2×10^7
D. About 2×10^7 versus 10^{12}

1049. What is the best-case time complexity of linear search, and when does it occur? Understanding

A. O(1), when the target is the first element
B. O(n), when the target is the first element
C. O(log n), when the list is sorted
D. O(1), when the target is not in the list

1050. What is the time complexity of breadth-first search on a graph with V vertices and E edges stored as adjacency lists? Understanding

A. O(V × E)
B. O(V + E)
C. $O(V^2)$
D. O(E log V)

1051. Quick sort that always picks the first element as the pivot is given an already sorted array. What is its running time? Understanding

A. O(n log n), as in the average case
B. O(n), because the array is already sorted
C. $O(n^2)$, because every partition splits off only one element
D. O(log n), because each step halves the array

1052. If an algorithm is O(n), is it also correct to say it is $O(n^2)$? Understanding

A. No, because each algorithm belongs to exactly one Big-O class
B. No, because $O(n^2)$ means it must take n^2 steps
C. Only if the algorithm contains a nested loop
D. Yes, because Big-O is an upper bound and n^2 bounds n from above

1053. An algorithm's running time doubles every time the input grows by one element. Which class does it belong to? Understanding

A. $O(2^n)$
B. $O(n^2)$
C. O(2n)
D. O(n log n)

1054. In a binary min-heap with n elements, what are the times for inserting an element and for removing the minimum? Understanding

A. O(1) for both
B. O(log n) for both
C. O(n) for both
D. O(1) to insert, O(n) to remove the minimum

1055. What does it mean when a sorting algorithm is described as in-place? Understanding

A. It never moves any element
B. It runs in O(n) time
C. It needs only O(1) extra memory beyond the input array
D. It keeps equal elements in their original order

1056. A program lists every possible ordering of n distinct items. Which complexity describes the number of orderings it produces? Understanding

A. $O(2^n)$
B. $O(n^2)$
C. O(n log n)
D. O(n!)

1057. Which statement about $\log_2(n!)$ is correct? Understanding

A. It is Θ(n log n)
B. It is Θ(n)
C. It is Θ(n!)
D. It is Θ(log n)

1058. Which recurrence describes binary search, and what does it solve to? Understanding

A. T(n) = 2T(n/2) + O(1), which is O(log n)
B. T(n) = T(n/2) + O(1), which is O(log n)
C. T(n) = T(n/2) + O(n), which is O(n log n)
D. T(n) = T(n − 1) + O(1), which is O(log n)

1059. Which statement is correct? Understanding

A. Both 2^{n+1} and 2^{2n} are $O(2^n)$
B. Neither 2^{n+1} nor 2^{2n} is $O(2^n)$
C. 2^{n+1} is $O(2^n)$, but 2^{2n} is not $O(2^n)$
D. 2^{2n} is $O(2^n)$, but 2^{n+1} is not

1060. Which pair of functions belongs to the same Big-O class? Understanding

A. n^2 and n^3
B. 2^n and 3^n
C. n and n log n
D. $\log(n^2)$ and log n

1061. How many times does count += 1 run when n = 20? Application

```
n = 20
count = 0
for i in range(0, n, 2):
    count += 1
print(count)
```

A. 10
B. 20
C. 9
D. 11

1062. An O(n) algorithm takes 3 seconds for n = 10,000. Roughly how long should it take for n = 40,000? Application

A. 48 seconds
B. 12 seconds
C. 6 seconds
D. 3 seconds

1063. At most how many elements does binary search examine in a sorted array of 1,000 elements? Application

A. 500
B. 1,000
C. 10
D. 100

1064. How many times does count += 1 run when n = 5? Application

```
n = 5
count = 0
for i in range(n):
    for j in range(n):
        for k in range(n):
            count += 1
print(count)
```

A. 15
B. 25
C. 75
D. 125

1065. List a has n items and list b has m items. What is the time complexity of this function? Application

```
def total_both(a, b):
    s = 0
    for x in a:
        s += x
    for y in b:
        s += y
    return s
```

A. O(n + m)
B. O(n × m)
C. O(n)
D. O(max(n, m)^2)

1066. An O(log n) search takes about 10 steps for n = 1,024. About how many steps will it take for n = 1,048,576 (2^{20})? Application

A. 10,240
B. 20
C. 1,024
D. 100

1067. How many times does count += 1 run when n = 10? Application

```
n = 10
count = 0
for i in range(n):
    for j in range(i, n):
        count += 1
print(count)
```

A. 45
B. 100
C. 55
D. 50

1068. How many times does the loop body run when n = 81? Application

```
n = 81
i = n
count = 0
while i > 1:
    i //= 3
    count += 1
print(count)
```

A. 27
B. 3
C. 6
D. 4

1069. The outer loop doubles i and the inner loop runs n times. How many times does count += 1 run for n = 32, and what is the growth rate? Application

```
n = 32
count = 0
i = 1
while i < n:
    for j in range(n):
        count += 1
    i *= 2
print(count)
```

A. 160 times; O(n log n)
B. 1,024 times; $O(n^2)$
C. 63 times; O(n)
D. 32 times; O(n)

1070. A program stores an n × n grid of 4-byte integers. How much memory does the grid need for n = 1,000, and what is its space complexity? Application

A. About 4 KB; O(n)
B. About 4 MB; $O(n^2)$
C. About 4 GB; $O(n^3)$
D. About 4 MB; O(n)

1071. You must answer k membership queries on an unsorted list of n numbers. You sort the list once and then binary search for each query. What is the total time? Application

A. O(k × n)
B. O(n + k)
C. O(n log n + k log n)
D. O(n log n × k)

1072. You need to know whether any two numbers in a list add up to a target. Which approach does this in O(n) average time? Application

A. Compare every pair of numbers with two nested loops
B. Sort the list, then binary search for target − x for each x
C. Check every subset of the list
D. Scan once, checking whether target − x is already in a hash set, then adding x to the set

1073. An array of n numbers receives q queries, each asking for the sum of elements from index l to r. You first build a prefix-sum array. What is the total time? Application

A. O(n + q)
B. O(n × q)
C. O(q log n)
D. O(n^2)

1074. How many calls to f are made in total when f(8) is called? Application

```
calls = 0
def f(n):
    global calls
    calls += 1
    if n <= 1:
        return 1
    return f(n // 2) + f(n // 2)

f(8)
print(calls)
```

A. 8
B. 15
C. 4
D. 16

1075. In Java, a loop builds a string of n characters with s = s + c, where strings are immutable. What is the total cost, and what is the usual fix? Application

A. O(n); no fix is needed
B. O(n log n); sort the characters first
C. O(n^2); use a StringBuilder (or ''.join in Python)
D. O(n^2); use a longer string variable

1076. To find the largest sum of k consecutive elements, a sliding window adds the new element and subtracts the one that leaves. What is the time for an array of n elements? Application

A. O(n × k)
B. O(k)
C. O(n^2)
D. O(n)

1077. How many times does the loop body run when n = 65,536, and what is the growth rate? Application

```
n = 65536
i = 2
count = 0
while i < n:
    i = i * i
    count += 1
print(count)
```

A. 4 times; O(log log n)
B. 16 times; O(log n)
C. 256 times; O(√n)
D. 8 times; O(log n)

1078. A recursive selection sort finds the minimum of n items with a linear scan, then sorts the remaining n − 1 items recursively: $T(n) = T(n - 1) + n$. What is $T(n)$? Application

A. $O(n \log n)$
B. $O(n^2)$
C. $O(n)$
D. $O(2^n)$

1079. How many times does `count += 1` run when `n = 8`, and what is the growth rate? Application

```
n = 8
count = 0
for i in range(n):
    j = 1
    while j < i:
        j *= 2
        count += 1
print(count)
```

A. 28 times; $O(n^2)$
B. 8 times; $O(n)$
C. 14 times; $O(n \log n)$
D. 24 times; $O(n \log n)$

1080. A machine does about 10^8 simple operations per second. Roughly how long does an $O(n \log n)$ sort take for $n = 10^6$, counting $n \log_2 n$ operations? Application

A. About 2 hours
B. About 20 seconds
C. About 0.002 seconds
D. About 0.2 seconds

1081. A naive $n \times n$ matrix multiplication takes 1 second for $n = 200$. Roughly how long should it take for $n = 600$? Application

A. 27 seconds
B. 3 seconds
C. 9 seconds
D. 81 seconds

1082. A function makes one pass over a list of n items, then calls `sorted()` on the list once. What is its overall time complexity? Analysis

A. $O(n)$
B. $O(n \log n)$
C. $O(n^2)$
D. $O(n + \log n)$

1083. The inner loop sits inside an if that is true only when i == 0. What is the time complexity of this function? Analysis

```
def f(a):
    n = len(a)
    count = 0
    for i in range(n):
        if i == 0:
            for j in range(n):
                count += 1
        count += 1
    return count
```

A. $O(n^2)$
B. O(1)
C. O(n)
D. O(n log n)

1084. lst and queries each hold n items. What is the time complexity of this function, and why? Analysis

```
def count_hits(lst, queries):
    hits = 0
    for x in queries:
        if x in lst:
            hits += 1
    return hits
```

A. O(n), because there is only one loop
B. O(n log n), because in uses binary search
C. O(1), because in is a built-in operator
D. $O(n^2)$, because x in lst scans the list each time

1085. A student runs binary search on a sorted singly linked list of n nodes. Why is it not O(log n)? Analysis

A. Reaching the middle node needs O(n) link-following, so the search costs O(n) overall
B. Binary search only works on numbers, not on nodes
C. Linked lists cannot be sorted
D. It is O(log n); linked lists support binary search as well as arrays do

1086. A program repeatedly removes the smallest task and adds new tasks to a collection of n tasks. Which design gives the best time per operation? Analysis

A. An unsorted list, re-sorted before every remove: O(n log n) per remove
B. A binary heap: O(log n) per insert and per remove-min
C. A sorted array: O(1) to remove the minimum and O(1) to insert
D. A stack: O(1) for both operations

1087. A memoised recursive function solve(i, j) is called with $0 \le i, j < n$. Each distinct (i, j) is computed once and does O(1) work apart from its recursive calls. What is the total time? Analysis

A. O(n)
B. $O(2^n)$
C. $O(n^2)$
D. O(n log n)

1088. A device has very little spare memory and must sort a large array in O(n log n) worst-case time. Which choice fits best? Analysis

A. Merge sort, which needs O(n) extra space
B. Quick sort, because its worst case is O(n log n)
C. Insertion sort, because it uses O(1) extra space
D. Heap sort, which is O(n log n) and uses O(1) extra space

1089. This function removes duplicates while keeping order. What is its worst-case time for n items, and what is a better approach? Analysis

```
def dedupe(a):
    result = []
    for x in a:
        if x not in result:
            result.append(x)
    return result
```

A. $O(n^2)$; also track seen items in a set so each check is O(1)
B. O(n); it is already optimal
C. O(n log n); sort the result at the end
D. $O(n^2)$; use a second list instead of result

1090. This duplicate check returns as soon as it finds a match. A student says the early return makes it O(n). What is the worst-case complexity? Analysis

```
def has_dup(a):
    for i in range(len(a)):
        for j in range(i + 1, len(a)):
            if a[i] == a[j]:
                return True
    return False
```

A. O(n), because of the early return
B. $O(n^2)$, because with no duplicates every pair is still compared
C. O(1), because it can stop at the first pair
D. O(n log n), because the inner loop shrinks

1091. If i is odd the loop subtracts 1, otherwise it halves i. How many iterations does it make for n = 100, and what is the growth rate? Analysis

```
n = 100
i = n
count = 0
while i > 1:
    if i % 2 == 0:
        i //= 2
    else:
        i -= 1
    count += 1
print(count)
```

A. 50 iterations; O(n)
B. 6 iterations; O(log n)
C. 8 iterations; O(log n)
D. 99 iterations; O(n)

1092. For each i, the inner loop runs while j * j <= i. How many times does count += 1 run for n = 16, and what is the growth rate? Analysis

```
n = 16
count = 0
for i in range(1, n + 1):
    j = 1
    while j * j <= i:
        count += 1
        j += 1
print(count)
```

A. 256 times; $O(n^2)$
B. 64 times; O(n log n)
C. 16 times; O(n)
D. 38 times; $O(n\sqrt{n})$

1093. A function visits every node of a binary tree by recursing into both halves and doing constant work per node: T(n) = 2T(n/2) + O(1). A student answers O(n log n). What is correct? Analysis

A. O(n), because the work per level doubles and sums to about 2n
B. O(n log n), as the student says
C. O(log n), because the problem halves each time
D. $O(n^2)$, because there are two recursive calls

1094. An algorithm does linear work and then recurses on one half of the input: T(n) = T(n/2) + n. What is T(n)? Analysis

A. O(n log n)
B. O(n)
C. O(log n)
D. $O(n^2)$

1095. Both versions of binary search are O(log n) in time. How do their extra-space costs compare? Analysis

A. Both use O(1) extra space
B. Recursive uses O(n) stack space; iterative uses O(log n)
C. Recursive uses O(log n) stack space; iterative uses O(1)
D. Both use O(log n) extra space

1096. A student timed insertion sort on already sorted arrays of growing size, saw linear growth and concluded insertion sort is O(n). What is wrong with the conclusion? Analysis

A. Nothing; measured times prove the Big-O class
B. Insertion sort is actually O(n log n) on every input
C. The timer was wrong, because insertion sort cannot be linear on any input
D. Sorted input is the best case; reverse-sorted input gives $\Theta(n^2)$, so the worst case is $O(n^2)$

Problem Solving Using Programming MCQs for NSCT

1097. When you dry-run code by hand, what is a trace table used for? Understanding

A. Recording the value of each variable after every step
B. Storing the final output only
C. Listing the syntax errors in the code
D. Measuring how long each line takes to run

1098. Which Python operator gives the remainder after integer division? Understanding

A. //
B. /
C. **
D. %

1099. You need to keep asking for a password until the user types the correct one. Which loop fits best? Understanding

A. No loop; one if statement
B. A while loop
C. A for loop over range(10)
D. A for loop over the password's characters

1100. In the accumulator pattern, where is the accumulator variable (for example total = 0) usually initialised? Understanding

A. Once, before the loop starts
B. Inside the loop, at the start of each pass
C. After the loop ends
D. It never needs initialising

1101. In Python, when is b evaluated in the expression a and b? Understanding

A. Never, because and only looks at a
B. Only when a is falsy
C. Always, before a
D. Only when a is truthy

1102. A global variable x is 10. A function then runs x = 5 inside its body, with no global statement. What is the global x after the call? Understanding

```
x = 10
def f():
    x = 5
f()
print(x)
```

A. None
B. 5
C. It raises NameError
D. 10

1103. How do you get the last element of a non-empty Python list a? Understanding

A. a[last]
B. a[0]
C. a[-1]
D. a[len(a)]

1104. What is the difference between return and print inside a function? Understanding

A. return sends a value back to the caller; print only displays text
B. They are interchangeable
C. return also prints the value to the screen
D. print sends a value back; return only displays it

1105. In a search loop, a variable found = False is set to True when the target is seen. What is this variable called? Understanding

A. An accumulator
B. A flag
C. A constant
D. A loop counter

1106. Which two parts must every correct recursive function have? Understanding

A. A loop and a global variable
B. Two recursive calls and no base case
C. A print statement and a return statement
D. A base case, and a recursive call that moves towards it

1107. What does -7 // 2 evaluate to in Python? Understanding

```
print(-7 // 2)
```

A. 3
B. -3
C. -4
D. -3.5

1108. What does 2 ** 3 ** 2 evaluate to in Python? Understanding

```
print(2 ** 3 ** 2)
```

A. 512
B. 64
C. 36
D. 12

1109. A function receives a list and calls append on it. What happens to the caller's list? Understanding

```
def add_one(lst):
    lst.append(1)

nums = [0]
add_one(nums)
print(nums)
```

A. It changes too, because both names refer to the same list object
B. It changes only if the function returns the list
C. It stays the same, because Python always copies arguments
D. It raises an error, because parameters are read-only

1110. What happens when this code runs? Understanding

```
s = "hello"
s[0] = "H"
print(s)
```

A. It raises TypeError, because strings are immutable
B. It prints hello
C. It prints Hello
D. It prints Hhello

1111. What is a dry run? Understanding

A. Running code with all print statements removed
B. Running code without saving it
C. Working through code by hand with sample input to predict what it does
D. Compiling code without running it

1112. What is stepwise refinement? Understanding

A. Rewriting the same code repeatedly to make it shorter
B. Breaking a problem into sub-problems and refining each until it maps to simple code
C. Running a program step by step in a debugger
D. Testing each line of code once

1113. What total does this loop print? Application

```
total = 0
for i in range(1, 6):
    if i % 2 == 0:
        total += i
print(total)
```

A. 9
B. 4
C. 6
D. 15

1114. What does this string slice print? Application

```
print("NSCT"[::-1])
```

A. TCSN
B. NSCT
C. TSCN
D. N

1115. What does this digit loop print for n = 4096? Application

```
n = 4096
s = 0
while n > 0:
    s += n % 10
    n //= 10
print(s)
```

A. 4096
B. 19
C. 21
D. 4

1116. What does this number-reversing loop print for n = 1203? Application

```
n = 1203
rev = 0
while n > 0:
    rev = rev * 10 + n % 10
    n //= 10
print(rev)
```

A. 321
B. 30210
C. 1203
D. 3021

1117. What does this list comprehension print? Application

```
print([x * x for x in range(5) if x % 2])
```

A. [0, 4, 16]
B. [0, 1, 4, 9, 16]
C. [1, 4, 9, 16]
D. [1, 9]

1118. What does the call f(1, c=4) print? Application

```
def f(a, b=2, c=3):
    return a + b * c

print(f(1, c=4))
```

A. 12
B. 9
C. 7
D. 13

1119. What string do these nested loops build and print? Application

```
s = ""
for i in range(3):
    for j in range(i + 1):
        s += str(j)
print(s)
```

A. 0010120123
B. 001012
C. 000111222
D. 012

1120. What does g(7) print? Application

```
def g(n):
    if n <= 1:
        return n
    return g(n - 1) + g(n - 2)

print(g(7))
```

A. 21
B. 13
C. 7
D. 8

1121. What does the call p(3) print? Application

```
def p(n):
    if n == 0:
        return
    p(n - 1)
    print(n, end=" ")

p(3)
```

A. 0 1 2 3
B. 3 2 1
C. 3 2 1 0
D. 1 2 3

1122. After pushing through b, what does this JavaScript print? Application

```
let a = [1, 2, 3];
let b = a;
b.push(4);
console.log(a.length);
```

A. 4
B. 3
C. undefined
D. TypeError

1123. What does this word-counting code print? Application

```
words = "a b a c b a".split()
counts = {}
for w in words:
    counts[w] = counts.get(w, 0) + 1
print(counts["a"])
```

A. 6
B. 2
C. 1
D. 3

1124. What does this JavaScript sort print? Application

```
console.log([10, 1, 2].sort().join(","));
```

A. 1,2,10
B. 10,2,1
C. 2,1,10
D. 1,10,2

1125. After x = x + [4], what does print(y) show? Application

```
x = [1, 2, 3]
y = x
x = x + [4]
print(y)
```

A. [4]
B. [1, 2, 3, 4]
C. [1, 2, 3]
D. [1, 2, 3, [4]]

1126. What does this for-else code print? Application

```
for i in range(3):
    if i == 5:
        break
else:
    print("done")
```

A. 0 1 2 done
B. SyntaxError
C. done
D. Nothing

1127. Task: count how many numbers in list a are strictly greater than the list's average. Which expression is correct? Analysis

A. len(a) - sum(a) / len(a)
B. sum(x for x in a if x > sum(a) / len(a))
C. sum(1 for x in a if x > sum(a) / len(a))
D. len([x for x in a if x >= sum(a) / len(a)])

1128. For which input does this is_prime function give the wrong answer? Analysis

```
def is_prime(n):
    for d in range(2, n):
        if n % d == 0:
            return False
    return True
```

A. 13
B. 2
C. 1
D. 9

1129. A faster prime test only tries divisors up to √n: for d in range(2, BOUND). Which BOUND is correct for every n ≥ 2? Analysis

A. n // 2
B. n ** 0.5
C. int(n ** 0.5) + 1
D. int(n ** 0.5)

1130. How many character comparisons does this function make for "racecar"? Analysis

```
def is_pal(s):
    i, j = 0, len(s) - 1
    count = 0
    while i < j:
        count += 1
        if s[i] != s[j]:
            return False, count
        i += 1
        j -= 1
    return True, count

print(is_pal("racecar"))
```

A. 4
B. 3
C. 7
D. 6

1131. A student meant to generate Fibonacci numbers but replaced the tuple assignment with two separate lines. What does this code print? Analysis

```
a, b = 0, 1
for _ in range(6):
    a = b
    b = a + b
print(a)
```

A. 8
B. 32
C. 64
D. 13

1132. The Collatz rule halves an even n and turns an odd n into 3n + 1. How many steps does this code count before reaching 1, starting from 6? Analysis

```
n = 6
steps = 0
while n != 1:
    if n % 2 == 0:
        n = n // 2
    else:
        n = 3 * n + 1
    steps += 1
print(steps)
```

A. 8
B. 16
C. 9
D. 6

1133. What does this JavaScript loop with continue and break print? Analysis

```
let s = 0;
for (let i = 0; i < 5; i++) {
  if (i === 2) continue;
  if (i === 4) break;
  s += i;
}
console.log(s);
```

A. 10
B. 4
C. 1
D. 6

1134. What does this list of lambdas print when each is called? Analysis

```
fs = [lambda: i for i in range(3)]
print([f() for f in fs])
```

A. [2, 2, 2]
B. [0, 0, 0]
C. NameError
D. [0, 1, 2]

1135. What does this count call print? Analysis

```
print("banana".count("ana"))
```

A. 0
B. 2
C. 3
D. 1

1136. What does h(9875) print? Analysis

```
def h(n):
    if n == 0:
        return 0
    return n % 10 + h(n // 10)

print(h(9875))
```

A. 5789
B. 20
C. 29
D. 9875

1137. Before writing any code for a problem, what should you identify first? Understanding

A. The inputs, the required outputs and the rules connecting them
B. The fastest programming language for the task
C. The variable names you will use
D. The number of lines the program will need

1138. A program reads marks until the user enters -1, which is not a real mark. What is -1 called here? Understanding

A. A flag
B. An accumulator
C. A sentinel value
D. A counter

1139. In the input-process-output model of a program that averages a student's marks, which part is computing the average? Understanding

A. Input
B. Output
C. Storage
D. Process

1140. In a language without tuple assignment, what do you need to swap the values of a and b correctly? Understanding

A. A temporary variable to hold one value
B. A second loop
C. A recursive function
D. A global variable

1141. Which Python call converts the string "42" into the integer 42? Understanding

A. str(42)
B. int("42")
C. float("42")
D. len("42")

1142. When testing a function, what is an edge case? Understanding

A. Any input that makes the program run slowly
B. The most common input users will give
C. An input at the boundary of what is allowed, such as an empty list or the largest value
D. An input that contains a syntax error

1143. What does break do inside a loop? Understanding

A. Skips to the next iteration of the loop
B. Stops the whole program
C. Restarts the loop from the beginning
D. Exits the innermost loop immediately

1144. Which of these is a syntax error rather than a logic error? Understanding

A. Leaving out the colon after an if condition in Python
B. Using + where - was needed in a formula
C. Starting a loop counter at 1 instead of 0
D. Dividing by the wrong variable

1145. What type does Python 3's input() always return? Understanding

A. An integer (int)
B. A string (str)
C. Whatever type the user typed
D. A list of words

1146. What is the difference between = and == in Python? Understanding

A. = compares values; == assigns them
B. They are interchangeable in Python
C. = assigns a value to a variable; == compares two values for equality
D. == assigns a value only inside if statements

1147. Which set of test inputs best checks a function that returns the largest number in a list? Understanding

A. [1, 2, 3], [4, 5, 6], [7, 8, 9]
B. [10, 20, 30], [10, 20, 30], [10, 20, 30]
C. [100], [200], [300]
D. [7], [-5, -2, -9], [3, 8, 8, 1]

1148. With int operands, what do 7 / 2 give in C and in Python 3? Understanding

A. 3 in C and 3.5 in Python 3
B. 3.5 in both
C. 3 in both
D. 3.5 in C and 3 in Python 3

1149. In Python, which of these values is falsy? Understanding

A. "0"
B. []
C. [0]
D. " "

1150. What does a Python function return if it finishes without reaching a return statement? Understanding

A. 0
B. An empty string
C. None
D. The value of its last expression

1151. Binary search assumes its input list is sorted. What is this assumption called? Understanding

A. A postcondition
B. A loop invariant
C. A side effect
D. A precondition

1152. In a loop that sums a list, the statement "total equals the sum of the items processed so far" is true before and after every iteration. What is it called? Understanding

A. A loop invariant
B. A sentinel
C. A base case
D. A precondition

1153. Why does 0.1 + 0.2 == 0.3 evaluate to False in Python? Understanding

A. Python rounds every float to one decimal place
B. 0.1 and 0.2 cannot be stored exactly in binary floating point, so the sum is slightly off
C. == cannot compare float values
D. Python adds floats as strings

1154. In Python, after for i in range(3): finishes, can you still read i? Understanding

A. No; i is deleted when the loop ends
B. Yes; it holds 3, one past the last value
C. Yes; it holds 2, because Python loops do not create a new scope
D. No; reading it raises a SyntaxError

1155. What does it mean that a sorting function is stable? Understanding

A. It never crashes on any input
B. It always takes the same time
C. It sorts without using extra memory
D. Items with equal keys keep their original relative order

1156. In Java, what does Integer.MAX_VALUE + 1 evaluate to? Understanding

A. Integer.MIN_VALUE (-2147483648), because int arithmetic wraps around
B. 2147483648, because Java widens the result automatically
C. An ArithmeticException is thrown
D. Integer.MAX_VALUE, because the value saturates

1157. What do round(2.5) and round(3.5) return in Python 3? Understanding

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

1158. In Java, why can s1 == s2 be false even when two strings contain the same characters? Understanding

A. == compares only the first character of each string
B. == ignores letter case
C. == compares object references, so it checks whether both refer to the same object; use equals()
D. Java strings cannot be compared at all

1159. A program sorts students by name with a stable sort, then sorts the result by grade with a stable sort. How are students ordered within the same grade? Understanding

A. In their original input order
B. In reverse alphabetical order
C. In random order
D. Alphabetically by name

1160. What does this print? Application

```
print(list(range(2, 10, 3)))
```

A. [2, 5, 8]
B. [2, 5, 8, 11]
C. [2, 3, 4]
D. [3, 6, 9]

1161. This loop finds the position of the largest value. What does it print? Application

```
a = [3, 9, 2, 9, 4]
m = a[0]
idx = 0
for i in range(len(a)):
    if a[i] > m:
        m = a[i]
        idx = i
print(idx)
```

A. 3
B. 1
C. 9
D. 0

1162. What does this print? Application

```
x = 7
y = x % 3 + x // 3
print(y)
```

A. 2
B. 1
C. 3
D. 4

1163. How many vowels does this loop count in "education"? Application

```
count = 0
for ch in "education":
    if ch in "aeiou":
        count += 1
print(count)
```

A. 4
B. 9
C. 6
D. 5

1164. What does this JavaScript print? Application

```
console.log("5" + 3, "5" - 3);
```

A. 53 2
B. 8 2
C. 53 53
D. 8 NaN

1165. What does this print? Application

```
print("ab" * 3)
```

A. ab3
B. abababab
C. ab ab ab
D. aabbab

1166. What does the second call print? Application

```
def f(x, acc=[]):
    acc.append(x)
    return acc

f(1)
print(f(2))
```

A. [2]
B. [1]
C. [1, 2]
D. [2, 1]

1167. This loop is Euclid's algorithm. What does it print? Application

```
a, b = 48, 18
while b != 0:
    a, b = b, a % b
print(a)
```

A. 18
B. 3
C. 2
D. 6

1168. What does this loop print for n = 13? Application

```
n = 13
bits = ""
while n > 0:
    bits = str(n % 2) + bits
    n //= 2
print(bits)
```

A. 1101
B. 1011
C. 13
D. 110

1169. What does outer() print? Application

```
def outer():
    x = 1
    def inner():
        return x + 1
    x = 5
    return inner()

print(outer())
```

A. 2
B. 6
C. 1
D. It raises a NameError

1170. What does this chain of string methods print? Application

```
s = "  NSCT 2026 "
print(s.strip().lower().split())
```

A. ['NSCT', '2026']
B. ['', 'nsct', '2026', '']
C. ['nsct', '2026']
D. 'nsct 2026'

1171. What does this print? Application

```
a = [1, 2, 3, 4, 5]
print(a[1:4], a[-2:])
```

A. [2, 3, 4, 5] [4, 5]
B. [1, 2, 3] [5]
C. [2, 3, 4] [5, 4]
D. [2, 3, 4] [4, 5]

1172. This code runs one pass of bubble sort. What does it print? Application

```
a = [5, 1, 4, 2, 8]
for i in range(len(a) - 1):
    if a[i] > a[i + 1]:
        a[i], a[i + 1] = a[i + 1], a[i]
print(a)
```

A. [1, 4, 2, 5, 8]
B. [1, 2, 4, 5, 8]
C. [1, 5, 4, 2, 8]
D. [5, 1, 4, 2, 8]

1173. What does this C program print? Application

```
#include <stdio.h>
int main(void) {
    int s = 0;
    for (int i = 1; i <= 10; i++) {
        if (i % 3 == 0) continue;
        s += i;
    }
    printf("%d\n", s);
    return 0;
}
```

A. 55
B. 37
C. 18
D. 40

1174. What does this print? Application

```
print(bool("False"), bool(0))
```

A. False False
B. False True
C. True False
D. True True

1175. What does this f-string print? Application

```
a, b = 10, 3
print(f"{a / b:.2f}")
```

A. 3
B. 3.3333333333333335
C. 3.34
D. 3.33

1176. What does this print? Application

```
grid = [[0] * 3] * 3
grid[0][0] = 1
print(grid)
```

A. [[1, 0, 0], [1, 0, 0], [1, 0, 0]]
B. [[1, 0, 0], [0, 0, 0], [0, 0, 0]]
C. [[1, 1, 1], [0, 0, 0], [0, 0, 0]]
D. [[1, 1, 1], [1, 1, 1], [1, 1, 1]]

1177. What does this JavaScript print? Application

```
for (var i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 0);
}
```

A. 0 1 2 (one per line)
B. 3 3 3 (one per line)
C. 2 2 2 (one per line)
D. undefined three times

1178. What does this Java snippet print? Application

```
int x = 10;
x = x++ + ++x;
System.out.println(x);
```

A. 21
B. 20
C. 22
D. 23

1179. What does this print? Application

```
def f():
    try:
        return 1
    finally:
        print("F", end=" ")

print(f())
```

A. 1 F
B. 1
C. F
D. F 1

1180. What does r(3) print? Application

```
def r(n):
    if n == 0:
        return []
    return r(n - 1) + [n] + r(n - 1)

print(r(3))
```

A. [1, 2, 1, 3, 1, 2, 1]
B. [1, 2, 3]
C. [3, 2, 1]
D. [1, 1, 2, 1, 1, 2, 3]

1181. This code should print the sum 1 + 2 + … + n but prints 10 for n = 5. Which fix is correct? Analysis

```
n = 5
total = 0
for i in range(1, n):
    total += i
print(total)
```

A. Change total = 0 to total = 1
B. Change range(1, n) to range(0, n)
C. Change range(1, n) to range(1, n + 1)
D. Change total += i to total += i + 1

1182. For which input does this function return the wrong maximum? Analysis

```
def biggest(a):
    m = 0
    for x in a:
        if x > m:
            m = x
    return m
```

A. [4, 9, 2]
B. [0, 0, 0]
C. [7]
D. [-3, -1, -2]

1183. What does this print, and what would make the word count correct? Analysis

```
s = "a  b"
print(len(s.split(" ")))
```

A. 3; use s.split() with no argument
B. 2; the code is already correct
C. 3; use s.split(" ") with two spaces
D. 4; use len(s) instead

1184. Which condition correctly tests whether year y is a Gregorian leap year? Analysis

A. y % 4 == 0
B. (y % 4 == 0 and y % 100 != 0) or y % 400 == 0
C. y % 4 == 0 and y % 100 != 0
D. y % 4 == 0 or y % 400 == 0

1185. What does find([4, 7, 9], 7) return, and why? Analysis

```
def find(a, t):
    for i in range(len(a)):
        if a[i] == t:
            return i
        return -1

print(find([4, 7, 9], 7))
```

A. 1, the index of 7
B. 0, because the loop stops at the first element
C. -1, because return -1 is inside the loop and runs after checking only the first item
D. None, because no return is reached

1186. This code is meant to reverse the list in place. What does it print? Analysis

```
a = [1, 2, 3, 4]
n = len(a)
for i in range(n):
    a[i], a[n - 1 - i] = a[n - 1 - i], a[i]
print(a)
```

A. [4, 3, 2, 1]
B. [4, 2, 3, 1]
C. [1, 3, 2, 4]
D. [1, 2, 3, 4]

1187. Why might this loop never stop? Analysis

```
x = 0.0
while x != 1.0:
    x += 0.1
```

A. Repeated additions of 0.1 never give exactly 1.0, so x != 1.0 stays true
B. x is never changed inside the loop
C. 0.1 is rounded to 0 in Python
D. while loops cannot compare floats

1188. This code tries to remove every 2 from the list while looping over it. What does it print? Analysis

```
a = [1, 2, 2, 3]
for x in a:
    if x == 2:
        a.remove(x)
print(a)
```

A. [1, 3]
B. [1, 2, 3]
C. [1, 2, 2, 3]
D. [3]

1189. This function counts the trailing zeros of n!. What does it print for n = 100? Analysis

```
n = 100
zeros = 0
while n > 0:
    n //= 5
    zeros += n
print(zeros)
```

A. 20
B. 2
C. 24
D. 25

1190. The two-pointer function below searches a sorted list for a pair summing to 10. What does it print? Analysis

```
def pair(a, t):
    i, j = 0, len(a) - 1
    while i < j:
        s = a[i] + a[j]
        if s == t:
            return (a[i], a[j])
        if s < t:
            i += 1
        else:
            j -= 1
    return None

print(pair([1, 3, 4, 6, 8, 11], 10))
```

A. (1, 9)
B. (3, 8)
C. None
D. (4, 6)

1191. What does fact(5) return, and where is the bug? Analysis

```
def fact(n):
    if n == 0:
        return 0
    return n * fact(n - 1)

print(fact(5))
```

A. 0; the base case should return 1
B. 120; there is no bug
C. 1; the recursive call is never reached
D. It recurses forever; there is no base case

1192. This binary search uses while lo < hi with hi = len(a) - 1. For a = [1, 3, 5, 7], which call returns the wrong result? Analysis

```
def bs(a, t):
    lo, hi = 0, len(a) - 1
    while lo < hi:
        mid = (lo + hi) // 2
        if a[mid] == t:
            return mid
        elif a[mid] < t:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1
```

A. bs(a, 3)
B. bs(a, 7)
C. bs(a, 5)
D. bs(a, 4)

1193. In this pseudocode, FOR i ← 1 TO 4 includes both 1 and 4. What is output? Analysis

```
x ← 1
FOR i ← 1 TO 4
    x ← x * i + 1
ENDFOR
OUTPUT x
```

A. 16
B. 24
C. 65
D. 41

1194. b is a shallow copy of a. What does print(a) show? Analysis

```
a = [[1, 2], [3]]
b = a.copy()
b[0].append(9)
b[1] = [0]
print(a)
```

A. [[1, 2], [3]]
B. [[1, 2, 9], [0]]
C. [[1, 2], [0]]
D. [[1, 2, 9], [3]]

Data-Driven Problem Solving MCQs for the NSCT

1195. A department's NSCT practice-test pass rate rose from 40% to 50%. Which statement describes the change correctly? Understanding

A. It rose by 10%
B. It rose by 10 percentage points, a 25% relative increase
C. It rose by 25 percentage points
D. It rose by 50%

1196. You need to show how the number of daily active users of a campus app changed over 12 months. Which chart is the best fit? Understanding

A. Line chart
B. Pie chart
C. Histogram
D. Scatter plot of month against user ID

1197. A society's annual budget is split across four categories that add up to 100%. Which chart shows each category's share of the whole most directly? Understanding

A. Line chart
B. Scatter plot
C. Box plot
D. Pie chart (or a single 100% stacked bar)

1198. Which chart is best for checking whether students' weekly study hours are related to their test marks? Understanding

A. Pie chart
B. Stacked area chart
C. Bar chart of average marks per student name
D. Scatter plot

1199. Which measure of central tendency is least affected by a few extreme values? Understanding

A. Mean
B. Range
C. Median
D. Sum

1200. Over a summer, ice-cream sales and the number of swimming accidents in a city rise together. What is the most likely explanation? Understanding

A. A third factor, hot weather, increases both
B. Eating ice-cream causes swimming accidents
C. Swimming accidents cause people to buy ice-cream
D. The data must be wrong, because unrelated things cannot correlate

1201. How does a histogram differ from an ordinary bar chart? Understanding

A. A histogram can only show percentages
B. A histogram compares separate categories such as cities or brands
C. A histogram shows the distribution of a numeric variable grouped into ranges (bins), with adjacent bars
D. A histogram always uses horizontal bars

1202. A bar chart compares two exam averages, 96 and 98, but its y-axis starts at 95. Why can this mislead readers? Understanding

A. The truncated axis makes a small difference look several times larger than it is
B. Bar charts cannot show averages
C. The y-axis must always start at 100 for marks
D. The bars should be replaced by a pie chart

1203. Under the common 1.5 × IQR rule, when is a data point treated as an outlier? Understanding

A. When it is more than 1.5 away from the mean
B. When it is the largest value in the dataset
C. When it appears only once in the dataset
D. When it is below Q1 - 1.5 × IQR or above Q3 + 1.5 × IQR

1204. To estimate how many final-year students across Pakistan feel ready for the NSCT, a student surveys only members of one online study group. What is the main weakness? Understanding

A. The sample is too random
B. Surveys can never measure opinions
C. The result will be exactly correct if the group is large
D. The sample is not representative, so the result is biased

1205. In the data set 4, 7, 7, 2, 9, 7, 4, what is the mode? Understanding

A. 4
B. 7
C. 6
D. 9

1206. Two classes have the same average mark of 60. Class P has a standard deviation of 3 and class Q has a standard deviation of 15. What does this tell you? Understanding

A. Class Q performed better overall
B. Class P has more students
C. Marks in class Q are far more spread out around 60 than marks in class P
D. Class Q's median must be higher than 60

1207. What is Simpson's paradox? Understanding

A. A chart whose axis does not start at zero
B. A correlation between two variables caused by a third variable
C. A trend that appears in every subgroup reverses or disappears when the groups are combined
D. A dataset whose mean and median are equal

1208. A dataset of screen time against sleep hours gives a Pearson correlation coefficient r = -0.9. Which conclusion is justified? Understanding

A. There is a strong negative linear relationship: more screen time tends to go with less sleep
B. Screen time causes 90% of sleep loss
C. There is almost no relationship, because r is negative
D. Each extra hour of screen time reduces sleep by exactly 0.9 hours

1209. Before averaging a column of marks, you notice several blank cells. What is the best first step? Understanding

A. Replace every blank with 0 and carry on
B. Find out why the values are missing and decide how to handle them, for example by excluding them
C. Delete the whole column
D. Replace blanks with the highest mark so the average is not lowered

1210. For a set of monthly salaries, the mean is clearly larger than the median. What does this most likely indicate? Understanding

A. The distribution is perfectly symmetric
B. Most salaries are above the mean
C. The distribution is right-skewed, with a long tail of high salaries
D. The data contains a calculation error

1211. Quarterly sales of a small software firm (units): Q1 120, Q2 150, Q3 135, Q4 180. By what percentage did sales grow from Q1 to Q4? Application

A. 33.3%
B. 60%
C. 50%
D. 150%

1212. Five students scored 62, 75, 81, 75 and 90. What is the mean score? Application

A. 75
B. 77.5
C. 76.6
D. 81

1213. A website's average daily visitors fell from 80 thousand to 68 thousand. What is the percentage change? Application

A. -12%
B. -15%
C. -17.6%
D. -1.2%

1214. Section A has 30 students with an average of 60 marks. Section B has 20 students with an average of 70. What is the combined average? Application

A. 64
B. 65
C. 62
D. 66

1215. Six API response times (ms) were logged. What is the median response time? Application

```
request : 1    2    3    4    5    6
time_ms : 120  95   300  110  105  98
```

A. 138 ms
B. 107.5 ms
C. 110 ms
D. 105 ms

1216. In a pie chart of a project budget, the 'Cloud hosting' slice is 35% of the total. What is its central angle? Application

A. 35°
B. 105°
C. 140°
D. 126°

1217. An app had 10,000 users. Users grew by 20% in March and then fell by 20% in April. How many users did it have at the end of April? Application

A. 10,000
B. 9,600
C. 9,800
D. 10,400

1218. A startup's revenue grew from 1,000 thousand rupees to 1,331 thousand rupees over 3 years. What is the compound annual growth rate (CAGR)? Application

A. 11.03%
B. 33.1%
C. 9%
D. 10%

1219. The table shows practice-test pass rates (%) by department for two years. Which department had the largest gain in percentage points? Application

```
dept | 2025 | 2026
CS   |  45  |  52
IT   |  38  |  47
SE   |  50  |  56
AI   |  41  |  49
```

A. CS
B. IT
C. SE
D. AI

1220. For a set of lab marks, Q1 = 40 and Q3 = 60. Using the 1.5 × IQR rule, which of these marks is an outlier? Application

A. 95
B. 85
C. 88
D. 89

1221. Which landing page has the highest conversion rate (orders ÷ visits)? Application

```
page | visits | orders
A    | 1200   | 54
B    |  800   | 40
C    | 1500   | 60
D    |  600   | 27
```

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

1222. A bar chart of visitors to a university stall reads: Mon 40, Tue 55, Wed 38, Thu 61, Fri 47. What is the range? Application

A. 21
B. 14
C. 61
D. 23

1223. Daily support tickets for five days were 10, 12, 15, 11, 16. What is the 3-day moving average on day 5? Application

A. 12.8
B. 13
C. 14
D. 16

1224. An A/B test gives: version A, 2,000 visitors and 100 sign-ups; version B, 2,500 visitors and 140 sign-ups. How does B compare with A? Application

A. B is 40% better because it has 40 more sign-ups
B. B converts 12 percentage points better than A
C. B is 0.6% better in relative terms
D. B converts 5.6% vs 5.0%: a 0.6 percentage-point gain, or a 12% relative lift

1225. Pass counts for two tutors are shown. Tutor X has the higher overall pass rate. Which conclusion is best supported? Analysis

```
tutor | morning (pass/total) | evening (pass/total)
X     | 81 / 90              |  2 / 10
Y     | 19 / 20              | 24 / 80
```

A. Tutor X is the better tutor because 83% beats 43%
B. Tutor Y has the higher pass rate in both batches; X's overall lead comes from teaching mostly the easier morning batch
C. The data is inconsistent, because a tutor cannot be worse overall but better in every batch
D. Tutor X is better in the evening batch only

1226. Data shows that students who visit the library more often score higher in exams. The administration plans to make library visits compulsory to raise marks. What is the strongest criticism? Analysis

A. The link may be due to a confounder such as motivation, so forcing visits may not raise marks; a controlled trial would test it
B. The data is useless, because libraries have no effect on learning
C. The plan is certain to work, because the correlation is positive
D. The sample is too small, whatever its size

1227. A slide's bar chart shows product B's rating as 52 and product A's as 49. The y-axis starts at 48, so B's bar looks four times as tall as A's. What is B's actual relative advantage? Analysis

A. About 6%
B. About 300%
C. About 400%
D. About 3%

1228. Branch P converted 10% of 100 leads; branch Q converted 2% of 900 leads. A manager reports the company's overall conversion rate as 6%. What is the correct overall rate? Analysis

A. 6%
B. 2.8%
C. 12%
D. 5.5%

1229. A plagiarism checker flags 95% of copied assignments and wrongly flags 5% of original ones. 2% of all assignments are copied. If an assignment is flagged, roughly how likely is it to be copied? Analysis

A. About 28%
B. About 95%
C. About 50%
D. About 2%

1230. A blog studies 20 famous tech founders, finds that 15 dropped out of university, and concludes that dropping out helps you succeed. What is the main flaw? Analysis

A. The sample should have included only graduates
B. Survivorship bias: it ignores the many dropouts who did not succeed
C. 15 out of 20 is too low to show any pattern
D. Founders' education is a categorical variable, so it cannot be analysed

1231. A dashboard reports an average quiz mark of 50 for these six records. Two students had not yet sat the quiz. What explains the figure, and what is the average of students who sat it? Analysis

```
student | S1 | S2 | S3 | S4 | S5 | S6
mark    | 70 | -- | 80 | 90 | -- | 60
```

A. The dashboard is right; the average is 50
B. Blanks were ignored; the true average is 50
C. Blanks were counted as 0; the average of the four who sat it is 75
D. The median should be used; it is 85

1232. A marketing campaign ran in December 2025. Sales fell 16% from November to December, so a manager calls it a failure. Using the table, which conclusion is best supported? Analysis

```
month    | 2024 | 2025
November | 480  | 500
December | 360  | 420
```

A. The campaign failed, because December sales were lower than November
B. The campaign had no effect, because both years show a dip
C. Nothing can be concluded without monthly data for ten years
D. December always dips; December sales grew 16.7% year on year versus 4.2% for November, so the campaign probably helped

1233. A small firm's monthly salaries (thousand rupees) are 40, 45, 50, 55, 60 and 400 (the owner). Which figure best describes a typical employee's salary? Analysis

A. The mean, about 108.3 thousand
B. The mode, 400 thousand
C. The median, 52.5 thousand
D. The range, 360 thousand

1234. A service promise says 95% of requests must finish within 200 ms. Which configuration meets it, based on these measurements? Analysis

```
config | mean_ms | p95_ms | max_ms
A      | 120     | 260    | 900
B      | 140     | 190    | 750
```

A. Configuration B, because its 95th-percentile latency is under 200 ms
B. Configuration A, because its mean latency is lower
C. Both, because both means are under 200 ms
D. Neither, because both maximum latencies exceed 200 ms

1235. A university wants to compare how many students enrolled in each of its five degree programmes this year. Which chart fits best? Understanding

A. Line chart
B. Scatter plot
C. Histogram
D. Bar chart

1236. What does a pivot table in a spreadsheet mainly do? Understanding

A. Rotates a chart by 90 degrees
B. Sorts a single column alphabetically
C. Groups rows by a field and summarises them, for example total sales per region
D. Locks cells so they cannot be edited

1237. In data-driven problem solving, what should you do first? Understanding

A. Collect every dataset the organisation holds
B. Build a dashboard with as many charts as possible
C. Pick the chart colours and layout
D. State the question clearly and decide which metric would answer it

1238. In a standard box plot, what does the line inside the box show? Understanding

A. The mean
B. The median
C. The mode
D. The midpoint of the minimum and maximum

1239. A table of average response time by weekday and hour is shaded so that slower cells are darker. What is this chart called? Understanding

A. Heatmap
B. Pie chart
C. Histogram
D. Scatter plot

1240. A student's mock-test result is reported at the 90th percentile. What does this mean? Understanding

A. The student scored 90% of the marks
B. The student answered 90 questions correctly
C. The student scored as well as or better than about 90% of test-takers
D. The student is among the top 90 candidates

1241. Between which values can Pearson's correlation coefficient r lie? Understanding

A. 0 and 1
B. -1 and +1
C. 0 and 100
D. -100 and +100

1242. After a course ends, an optional online feedback form is completed by only 12% of students, mostly those who were very happy or very unhappy. Which bias is most likely? Understanding

A. Non-response (voluntary response) bias
B. Survivorship bias
C. Confirmation bias
D. Recall bias from a long time gap

1243. A software house starts rewarding developers by the number of lines of code they write. Soon code becomes longer and harder to maintain. Which principle does this illustrate? Understanding

A. The Pareto principle: 80% of effects come from 20% of causes
B. Simpson's paradox: aggregated trends can reverse
C. Goodhart's law: when a measure becomes a target, it stops being a good measure
D. Survivorship bias: only successes are observed

1244. Which chart shows categories as bars sorted from largest to smallest together with a line for the cumulative percentage, to find the 'vital few' causes? Understanding

A. Stacked area chart
B. Pareto chart
C. Box plot
D. Radar chart

1245. In an A/B test of two sign-up pages, why are visitors randomly assigned to version A or version B? Understanding

A. So that each version gets exactly the same number of visitors
B. So the two groups are alike on average, and a difference in results can be attributed to the page
C. So the test finishes faster
D. So that the better version is shown to more people

1246. On a chart with a logarithmic y-axis (1, 10, 100, 1,000), what do equal vertical distances represent? Understanding

A. Equal absolute differences, such as adding 10
B. Equal ratios, such as multiplying by 10
C. Equal percentages of the total
D. Equal time intervals

1247. The standard deviation of a set of recorded server temperatures is 0. What must be true? Understanding

A. Every recorded value is the same
B. The mean temperature is 0
C. There is exactly one recorded value
D. The data contains no outliers but may still vary

1248. City A reported 500 mobile-phone thefts last year and City B reported 300. Why can you not yet say City B is safer? Understanding

A. The counts must be compared as rates, for example thefts per 1,000 residents
B. 500 and 300 are too close to compare
C. Theft data is always unreliable
D. You must first calculate the median number of thefts

1249. A date column mixes '03/04/2026' entries typed day-first and others typed month-first. What is the best fix before analysis? Understanding

A. Convert every date to one unambiguous format such as ISO 8601 (YYYY-MM-DD) using the known source of each entry
B. Delete all dates that contain a slash
C. Sort the column so the mixed entries end up together
D. Keep the text as it is, since spreadsheets guess the format correctly

1250. A quiz's marks are left-skewed: most students scored high, with a long tail of a few very low marks. How do the mean and median usually compare? Understanding

A. The mean is higher than the median
B. The mean equals the median
C. The median is always zero in a skewed set
D. The mean is lower than the median

1251. A learning app groups users by the month they signed up and tracks what share of each group is still active after 1, 2 and 3 months. What is this analysis called? Understanding

A. Pareto analysis
B. Regression analysis
C. Cohort analysis
D. Sensitivity analysis

1252. Which metric is a leading indicator of a batch's final NSCT practice results? Understanding

A. The batch's final pass rate
B. The weekly share of assigned practice questions completed
C. Last year's graduation rate
D. The number of degrees the department awarded

1253. An A/B test of two checkout buttons reports a p-value of 0.03 for the difference in conversion rate. What does this mean? Understanding

A. There is a 3% chance that version B is better
B. If there were truly no difference, a gap at least this large would appear by chance about 3% of the time
C. There is a 97% chance that version B is better
D. Version B converts 3% better than version A

1254. A team tests 20 different button colours, each against the original at a 5% significance level, and celebrates the one colour that 'wins'. Assuming no colour really matters and the tests are independent, roughly how likely is at least one false win? Understanding

A. About 5%
B. About 20%
C. About 64%
D. About 100%

1255. In a random survey, the margin of error for a proportion is roughly proportional to 1/sqrt(n). If the sample size rises from 100 to 400, what happens to the margin of error? Understanding

A. It falls to a quarter
B. It doubles
C. It halves
D. It stays the same

1256. A straight trend line fits a startup's user numbers well for months 1 to 12. Why is using it to predict month 60 risky? Understanding

A. A trend line can only be used for interpolation inside the first month
B. Five years of data would make the line steeper
C. Trend lines can only be drawn on scatter plots, not over time
D. It is extrapolation: it assumes the same pattern continues far outside the observed range

1257. Countries with more internet users per capita have higher average incomes. A report concludes that any individual who gets internet access will earn more. What is the flaw called? Understanding

A. Survivorship bias
B. The ecological fallacy: inferring individual behaviour from group-level data
C. Simpson's paradox
D. Sampling error from a small sample

1258. The number of bugs found in five code reviews was 3, 8, 5, 10 and 4. What is the median? Application

A. 6
B. 8
C. 5
D. 4

1259. A course has 250 students and 18% of them failed. How many students passed? Application

A. 45
B. 232
C. 205
D. 218

1260. A laptop's price rose from Rs 80,000 to Rs 100,000, then later fell back to Rs 80,000. What was the percentage decrease in the fall? Application

A. 25%
B. 18%
C. 22.5%
D. 20%

1261. Commits pushed to a project over one week are shown. Which day accounted for exactly 25% of the week's commits? Application

```
day | commits
Mon | 12
Tue | 18
Wed | 20
Thu | 30
Fri | 40
```

A. Thu
B. Fri
C. Wed
D. Tue

1262. A town of 48,000 people recorded 120 dengue cases this season. What is the rate per 1,000 people? Application

A. 2.5
B. 25
C. 0.25
D. 4.0

1263. In a pie chart of a lab's spending, the 'Networking equipment' slice has a central angle of 72 degrees. What share of the total does it represent? Application

A. 72%
B. 20%
C. 25%
D. 7.2%

1264. A student's semester results are: a 3-credit course at grade point 4.0, a 3-credit course at 3.0, a 2-credit course at 2.0 and a 1-credit course at 3.0. What is the credit-weighted GPA, to two decimals? Application

A. 3.11
B. 3.00
C. 3.25
D. 2.89

1265. A delivery van drives 60 km to a warehouse at 30 km/h and returns the same 60 km at 60 km/h. What is its average speed for the round trip? Application

A. 45 km/h
B. 40 km/h
C. 42.5 km/h
D. 50 km/h

1266. Treating these 8 values as the whole population, what is their standard deviation: 2, 4, 4, 4, 5, 5, 7, 9? Application

A. 2
B. 4
C. 1.5
D. 2.14

1267. Mock-test marks have a mean of 60 and a standard deviation of 8. What is the z-score of a student who scored 76? Application

A. 16
B. 2
C. 1.27
D. 0.5

1268. A box plot of assignment marks shows: minimum 20, Q1 35, median 50, Q3 70, maximum 95. What is the interquartile range (IQR)? Application

A. 75
B. 50
C. 35
D. 20

1269. A marketing team stores totals per channel. What does this SQLite query output? Application

```
CREATE TABLE stats(channel TEXT, visits INT, conversions INT);
INSERT INTO stats VALUES ('ads',200,20),('email',40,6),
  ('social',100,12),('search',50,7);
SELECT channel, ROUND(100.0 * conversions / visits, 1) AS rate
FROM stats ORDER BY rate DESC LIMIT 1;
```

A. ads|10.0
B. search|14.0
C. email|0
D. email|15.0

1270. A course website's funnel last month: 10,000 visitors, 3,000 created accounts, 600 started the free trial and 150 paid. Which step has the lowest step-to-step conversion rate? Application

A. Account to free trial (20%)
B. Visitor to account (30%)
C. Free trial to paid (25%)
D. Visitor to paid (1.5%)

1271. A cost-of-living index uses 2023 = 100. It is 118 in 2025 and 129.8 in 2026. By what percentage did costs rise from 2025 to 2026? Application

A. 11.8%
B. 10%
C. 29.8%
D. 18%

1272. After a 20% discount, an online course costs Rs 6,400. What was the original price? Application

A. Rs 7,680
B. Rs 7,800
C. Rs 8,400
D. Rs 8,000

1273. Cloud costs rose 10% in the first year and 20% in the second year. What is the total increase over the two years? Application

A. 32%
B. 30%
C. 31%
D. 15%

1274. For study hours x = 1, 2, 3, 4, 5 and quiz marks y = 2, 4, 5, 4, 5, what is Pearson's correlation coefficient r, to two decimals? Application

A. 0.60
B. 0.90
C. -0.77
D. 0.77

1275. Monthly paid users (hundreds) for months 1 to 4 were 3, 5, 7 and 8. Using the least-squares line y = a + bx, what is the predicted value for month 6? Application

A. 11.7
B. 10
C. 12
D. 13.25

1276. A random survey of 400 students finds 50% prefer online classes. Using margin of error = 1.96 x sqrt(p(1 - p) / n) at 95% confidence, what is the margin of error? Application

A. About +/- 2.5 percentage points
B. About +/- 4.9 percentage points
C. About +/- 9.8 percentage points
D. About +/- 0.5 percentage points

1277. These are 20 API response times (ms). Using the nearest-rank method (rank = ceiling(p x n)), what is the 90th-percentile response time? Application

```
110, 95, 130, 102, 98, 250, 105, 99, 120, 101,
97, 140, 103, 96, 115, 108, 180, 100, 112, 104
```

A. 180 ms
B. 120 ms
C. 250 ms
D. 140 ms

1278. 1,000 users sign up in January. 60% are still active after month 1, and in each later month 80% of the previous month's active users remain. How many are active at the end of month 3? Application

A. 400
B. 480
C. 360
D. 384

1279. A slide shows total app downloads since launch as a line that rises every month, and the presenter says growth is accelerating. What is the main problem? Analysis

A. Line charts cannot show download data
B. The chart should be a pie chart
C. The y-axis should start at 1,000
D. A cumulative total always rises; you need new downloads per month to judge whether growth is speeding up

1280. Shop P has a 4.9-star average from 8 reviews. Shop Q has 4.6 stars from 2,300 reviews. Which reading is best supported? Analysis

A. P is clearly better because 4.9 is higher
B. Both ratings are equally reliable because they are averages
C. Q must have more unhappy customers in proportion than P
D. Q's rating is the more reliable estimate, because P's is based on a very small sample

1281. A student society's survey asks: 'Don't you agree that the new timetable is much better than the old one?' 85% say yes. What is the main problem with this result? Analysis

A. 85% is too high to be a real survey result
B. Yes/no questions can never be analysed
C. The leading wording pushes respondents towards 'yes', so the 85% is inflated
D. The sample should be smaller

1282. A line chart of a university's enrolments places the years 2010, 2020, 2024, 2025 and 2026 equally spaced on the x-axis. Why can this mislead? Analysis

A. Line charts should not show years
B. Enrolment should always be shown as a percentage
C. Five data points are too many for a line chart
D. Unequal time gaps drawn as equal make recent change look as fast as a decade of change

1283. A firm must choose: Option A gives a certain profit of Rs 50 thousand. Option B gives a 60% chance of Rs 100 thousand profit and a 40% chance of a Rs 20 thousand loss. Which statement is correct? Analysis

A. A has the higher expected value because it is certain
B. B has the higher expected value (Rs 52 thousand), but A has no risk, so the choice depends on how much risk the firm can bear
C. B's expected value is Rs 60 thousand, so B is clearly better
D. Both have an expected value of Rs 50 thousand

1284. A new task dashboard is rolled out. Tasks closed per week jump 25% in the first two weeks, then return to the old level by week six. What is the best interpretation? Analysis

A. The dashboard permanently improved productivity by 25%
B. A novelty effect: early behaviour changed because the tool was new, so short tests overstate the long-term benefit
C. The week-six data must be wrong and should be removed
D. The dashboard caused productivity to fall

1285. Users could opt in to a beta version of a study app. Beta users spent 40% more time in the app than other users, so the team plans to launch the beta to everyone. What is the main weakness of the evidence? Analysis

A. Self-selection: keen users chose the beta, so they may have used the app more anyway
B. 40% is too small a difference to matter
C. Time in app cannot be measured accurately
D. The beta group was probably larger than the other group

1286. A student-records extract has ages that include 0 and 230 among otherwise normal values of 17 to 30. What is the best action? Analysis

A. Treat them as likely entry errors: check them against the source and correct or flag them, and record what you did
B. Delete the two rows silently so the averages look right
C. Keep them, because removing any value is always wrong
D. Replace them with the mean age of the dataset without noting it

1287. A call centre rewards agents on calls handled per hour. Calls per hour rise, but complaints and repeat calls also rise. Which change to the metrics best fixes this? Analysis

A. Raise the calls-per-hour target further
B. Measure average call length only
C. Stop measuring agents completely
D. Pair calls per hour with first-contact resolution rate, so speed cannot be gained by leaving problems unsolved

1288. Feature X's users grew 400% (from 5 to 25). Feature Y's users grew 20% (from 10,000 to 12,000). A report says X is the fastest-growing and most important feature. What is the best response? Analysis

A. X is more important because 400% is bigger than 20%
B. Y grew faster because 12,000 is bigger than 25
C. Percentage changes cannot be calculated from such small numbers
D. X has the larger percentage growth, but Y added 2,000 users against X's 20, so report both relative and absolute change

1289. After a city launched an online form for reporting street crime, reported crimes rose 40% the following year. A news site says crime is soaring. Which conclusion is best supported? Analysis

A. The rise may largely reflect easier reporting rather than more crime; compare with a source not affected by the change, such as victim surveys
B. Crime definitely rose 40%
C. The online form caused criminals to commit more crime
D. The data should be ignored because online forms are unreliable

1290. A chart plots monthly revenue (left axis, Rs 0 to 50 million) and server outages (right axis, 0 to 6) as two lines that rise together and cross. A manager says outages are 'catching up with' revenue. What is wrong with this reading? Analysis

A. Revenue and outages cannot be plotted on the same chart
B. The right axis should also be in rupees
C. With two independent y-axes, where lines cross depends only on how each axis was scaled, so the crossing has no meaning
D. Two lines rising together proves outages cause revenue

1291. An online shop's data: March had 1,000 orders at an average order value of Rs 2,000; April had 800 orders at Rs 2,300. The team celebrates the 15% rise in average order value. Which conclusion is best supported? Analysis

A. Revenue rose 15% because average order value rose 15%
B. Revenue was unchanged because the two effects cancel out
C. Revenue fell 8% (Rs 2.0m to Rs 1.84m) because order count dropped, so the higher average does not mean the business did better
D. Average order value cannot rise when orders fall

1292. Speed cameras were installed at the 10 junctions with the most accidents in 2025. In 2026 accidents there fell 30%. Which conclusion is best supported? Analysis

A. The cameras caused the full 30% fall
B. The cameras had no effect at all
C. Part of the fall may be regression to the mean, since sites picked for an extreme year tend to improve anyway; compare with similar high-accident sites without cameras
D. The 2026 data must contain errors because the fall is too large

1293. A company hires only applicants who scored above 80 on its aptitude test. Among the people hired, test score shows almost no correlation with job performance. HR proposes dropping the test. What is the main flaw? Analysis

A. The correlation proves the test is useless
B. Job performance cannot be measured
C. Restriction of range: only high scorers were hired, so the data cannot show how low scorers would have performed
D. A correlation near zero means the test predicts performance perfectly

Creative & Innovative Thinking MCQs for the NSCT

1294. Edward de Bono's term 'lateral thinking' refers to which approach? Understanding

A. Following one logical chain of steps from premises to conclusion
B. Solving problems by deliberately looking at them from indirect, non-obvious angles instead of only step-by-step logic
C. Copying a known solution exactly as it was used before
D. Choosing the option with the lowest cost

1295. Which kind of thinking aims to generate as many different possible solutions as you can? Understanding

A. Convergent thinking
B. Divergent thinking
C. Deductive thinking
D. Linear thinking

1296. After a brainstorm produced 30 ideas, a team scores each idea against criteria and picks the best one. Which kind of thinking is this? Understanding

A. Divergent thinking
B. Lateral thinking
C. Convergent thinking
D. Random thinking

1297. Which rule is central to a classic brainstorming session? Understanding

A. Only senior members may suggest ideas
B. Every idea must be fully costed before it is said aloud
C. Stop as soon as the first workable idea appears
D. Defer judgement: no criticism while ideas are being generated

1298. Using SCAMPER, a student turns an old smartphone into a home security camera. Which SCAMPER prompt is this? Understanding

A. Substitute
B. Put to another use
C. Combine
D. Eliminate

1299. In de Bono's Six Thinking Hats, which hat is used for new ideas and alternatives? Understanding

A. Black hat
B. White hat
C. Green hat
D. Red hat

1300. During a Six Thinking Hats review of a new app idea, a member lists risks: privacy leaks, server costs and app-store rejection. Which hat are they wearing? Understanding

A. Yellow hat
B. Green hat
C. Black hat
D. Blue hat

1301. What does 'reverse brainstorming' (inversion) involve? Understanding

A. Brainstorming the solution backwards from the last step
B. Letting one person brainstorm while others criticise
C. Asking how to cause or worsen the problem, then turning each answer into a fix
D. Repeating an old brainstorm in reverse order

1302. An engineer notices burrs sticking to a dog's fur and designs a hook-and-loop fastener. Which creative technique does this illustrate? Understanding

A. Brute-force search
B. Analogical thinking (biomimicry): borrowing a solution from another domain
C. Convergent evaluation
D. Functional fixedness

1303. A student cannot think of using a coin to turn a slotted screw because 'coins are for paying'. What is this barrier called? Understanding

A. Functional fixedness
B. Confirmation bias
C. Divergent thinking
D. Abstraction

1304. What is first-principles thinking? Understanding

A. Using the first solution that comes to mind
B. Copying the approach of the market leader
C. Breaking a problem down to basic facts that must be true, then rebuilding a solution from them
D. Following company rules in the order they were written

1305. Tenants complain that the lift is slow. Instead of a new lift, the owner installs mirrors beside the lift doors and complaints fall. Which technique was used? Understanding

A. Reframing the problem from 'slow lift' to 'boring wait'
B. Brute force
C. Convergent thinking
D. Root-cause elimination of the lift motor fault

1306. Why do innovative teams build a quick prototype or minimum viable product (MVP) early? Understanding

A. Because a prototype replaces the need for any final product
B. To test key assumptions with real users cheaply before investing heavily
C. To avoid collecting user feedback
D. Because prototypes must include every planned feature

1307. Which is the standard order of the five design-thinking stages (Stanford d.school model)? Understanding

A. Define, Empathise, Prototype, Ideate, Test
B. Empathise, Define, Ideate, Prototype, Test
C. Ideate, Empathise, Define, Test, Prototype
D. Prototype, Test, Define, Ideate, Empathise

1308. What is a mind map mainly used for? Understanding

A. Calculating the cost of each idea
B. Proving that a solution is correct
C. Recording meeting attendance
D. Laying out ideas around a central topic so branches and links between them are visible

1309. Which statement best separates innovation from invention? Understanding

A. Innovation always requires a patent
B. Innovation is putting a new or improved idea into practical use so that it creates value
C. Invention and innovation mean exactly the same thing
D. Innovation only applies to physical products

1310. Instead of adding 1 + 2 + ... + 100 one by one, you pair the first and last terms (1 + 100, 2 + 99, ...). What total does this give? Application

A. 5000
B. 10100
C. 5050
D. 5100

1311. A knockout tournament has 64 teams. Each match eliminates exactly one team. How many matches are needed to find the champion? Application

A. 32
B. 64
C. 126
D. 63

1312. A drawer holds 10 black, 10 white and 10 blue socks. In the dark, what is the minimum number you must take out to be certain of a matching pair? Application

A. 4
B. 3
C. 2
D. 11

1313. Three switches outside a closed room control one ordinary filament bulb inside. You may enter the room only once. Which plan identifies the correct switch? Application

A. Flip all three switches on together, then enter
B. Flip each switch once quickly, then enter
C. Turn switch 1 on, enter, and guess between 2 and 3 if the bulb is off
D. Turn switch 1 on for ten minutes, turn it off, turn switch 2 on, then enter: lit = 2, warm and off = 1, cold and off = 3

1314. A list should hold every integer from 1 to 100 once, but one is missing. The 99 numbers present add up to 4977. Which number is missing? Application

A. 27
B. 77
C. 50
D. 73

1315. You have a 3-litre jug and a 5-litre jug, unmarked, and a tap. Each fill, empty or pour from one jug into the other counts as one step. What is the fewest number of steps to have exactly 4 litres in a jug? Application

A. 4
B. 5
C. 6
D. 8

1316. Eight coins look identical, but one is slightly heavier. Using a balance scale, what is the minimum number of weighings that always finds the heavy coin? Application

A. 3
B. 4
C. 1
D. 2

1317. You have two ropes; each takes exactly 60 minutes to burn, but unevenly. How can you measure exactly 45 minutes? Application

A. Light rope A at both ends and rope B at one end; when A burns out (30 min), light B's other end; B finishes 15 min later
B. Cut each rope at its midpoint and burn the pieces one after another
C. Light both ropes at one end and stop when they look three-quarters burnt
D. Burn rope A fully, then burn three-quarters of rope B by length

1318. Two opposite corner squares are removed from an 8 × 8 chessboard, leaving 62 squares. Can 31 dominoes, each covering two adjacent squares, tile it exactly? Application

A. Yes: 62 squares divided by 2 is exactly 31 dominoes
B. Yes, but only if the dominoes are placed diagonally
C. No: 62 is not divisible by 4
D. No: both removed squares are the same colour, leaving 32 of one colour and 30 of the other, but each domino covers one of each

1319. A program checks whether a list of n student IDs has any duplicate by comparing every pair, which is $O(n^2)$. Which alternative is fundamentally faster on average? Application

A. Insert each ID into a hash set and stop if it is already there: O(n) expected time
B. Compare every pair twice to be safe
C. Reverse the list and compare every pair again
D. Use two nested loops but start the inner loop at 0

1320. To answer thousands of range-sum queries quickly, you precompute prefix sums P, where P[i] is the sum of the first i elements. For A = [3, 1, 4, 1, 5, 9, 2, 6] (0-indexed), what is the sum of A[2] to A[5] inclusive? Application

A. 19
B. 20
C. 14
D. 28

1321. Ten people at a meeting each shake hands once with every other person. How many handshakes take place? Application

A. 45
B. 90
C. 100
D. 50

1322. What is the smallest group size that is certain to contain two people born in the same calendar month? Application

A. 12
B. 24
C. 7
D. 13

1323. Twelve coins include one counterfeit that may be heavier or lighter. What is the minimum number of balance weighings that always identifies it and says whether it is heavier or lighter? Application

    A. 2
    B. 3
    C. 4
    D. 12

1324. Measurements show the average wait for the hostel lift is about 40 seconds, yet residents say it 'takes forever'. The budget cannot fund a new lift. Which idea addresses the complaint best? Analysis

    A. Install a second lift shaft
    B. Ban residents of the first two floors from using the lift
    C. Add a display showing the lift's floor and estimated arrival time, plus a notice board in the waiting area
    D. Speed up the lift motor beyond its safety rating

1325. A team runs reverse brainstorming: 'How could we make sure first-year students never use our tutoring app?' They list: slow sign-up, English-only screens, no offline mode. What should they do next? Analysis

    A. Publish the list as the app's feature plan
    B. Stop, because reverse brainstorming only produces negative ideas
    C. Run a normal brainstorm and discard this list
    D. Turn each cause into a fix, for example one-tap sign-up, Urdu screens and offline notes, then prioritise them

1326. Instead of testing every pair, you find two numbers summing to 10 in the sorted array [1, 3, 4, 6, 8, 11] with two pointers: start at both ends; if the sum is too small move the left pointer right, if too large move the right pointer left. Which pair is found, after how many sum checks? Analysis

    A. (4, 6) after 5 checks
    B. (1, 8) after 2 checks
    C. (4, 6) after 3 checks
    D. (3, 8) after 4 checks

1327. The classic nine-dot puzzle asks you to join a 3 × 3 grid of dots with four straight lines without lifting your pen. Most people fail at first. Which self-imposed assumption blocks them? Analysis

    A. That the lines must stay inside the square formed by the dots
    B. That the lines must be straight
    C. That the pen cannot be lifted
    D. That every dot must be crossed

1328. A team scores three ideas from 1 to 10 (higher is better) on cost-friendliness (weight 0.5), impact (0.3) and speed (0.2). Which idea has the highest weighted score? Analysis

```
idea | cost-friendly | impact | speed
A    | 4             | 9      | 3
B    | 7             | 8      | 7
C    | 9             | 5      | 7
```

    A. Idea B, with 7.3
    B. Idea A, because it has the highest impact
    C. Idea C, with 7.4
    D. Idea B, because its unweighted total of 22 is highest

1329. A bank switches from one queue per counter to a single shared queue that feeds whichever counter is free. Which computing design uses the same idea? Analysis

A. A thread pool where idle workers take jobs from one shared work queue
B. Giving each CPU core its own fixed list of jobs assigned in advance
C. A stack where the newest job is always served first
D. Caching recent results to avoid repeating work

1330. Four people must cross a bridge at night with one torch. At most two cross at a time, at the slower person's pace, and the torch must be carried each way. Crossing times: 1, 2, 5 and 10 minutes. What is the minimum total time? Analysis

A. 19 minutes
B. 17 minutes
C. 18 minutes
D. 21 minutes

1331. In a brainstorming meeting, the team lead criticises each idea as soon as it is suggested, and after ten minutes people stop speaking. What is the best fix? Analysis

A. Let the lead suggest all the ideas
B. Shorten the meeting to five minutes
C. Ask people to send only fully tested ideas
D. Separate idea generation from evaluation: collect ideas first without comment, then judge them

1332. A team spent six months building every feature of a revision app before showing it to any student, then found students wanted only past-topic quizzes. Which principle did they skip? Analysis

A. Write more documentation before coding
B. Prototype early and test assumptions with real users
C. Add even more features before launch
D. Use a different programming language

1333. A manager says, 'We need a bigger server.' Monitoring shows the CPU at 20%, but pages take 6 seconds because each loads 8 MB of unoptimised images. Which reframed problem statement best opens good solutions? Analysis

A. How might we buy a bigger server cheaply?
B. How might we raise CPU usage to 100%?
C. How might we cut page load time for our users?
D. How might we convince users that 6 seconds is fast?

1334. In de Bono's Six Thinking Hats, which hat focuses on facts, figures and what information is still missing? Understanding

A. Red hat
B. Yellow hat
C. White hat
D. Blue hat

1335. During a Six Thinking Hats session, a member says, 'I can't explain why, but I have a bad feeling about this logo.' Which hat is in use? Understanding

A. Red hat
B. Black hat
C. White hat
D. Green hat

1336. Phone makers removed the physical keyboard to create a full touchscreen. Which SCAMPER prompt does this match? Understanding

A. Combine
B. Eliminate
C. Substitute
D. Put to another use

1337. Merging a phone and a camera into one device is an example of which SCAMPER prompt? Understanding

A. Adapt
B. Reverse
C. Eliminate
D. Combine

1338. Why do design teams phrase challenges as 'How might we...?' questions? Understanding

A. To make the problem sound more formal for managers
B. To turn a problem into an open opportunity that invites many possible solutions
C. To force a single correct answer quickly
D. To avoid talking to users

1339. In de Bono's 'random entry' technique, what does the team do? Understanding

A. Pick an unrelated random word or object and force links between it and the problem
B. Choose the winning idea at random to save time
C. Randomly assign people to roles
D. Randomly delete half of the ideas

1340. In brainwriting (such as the 6-3-5 method), people write ideas silently and pass their sheets on for others to build on. What problem of spoken brainstorming does this mainly reduce? Understanding

A. Having too many ideas to evaluate
B. The need for a facilitator
C. A few loud or senior people dominating while others hold back
D. Ideas being too unusual

1341. A transport app copies the 'nearest available driver' model from ride-hailing to let students book tutors nearby. Which SCAMPER prompt is this? Understanding

A. Eliminate
B. Reverse
C. Adapt
D. Modify

1342. In the design-thinking model, which stage turns interview and observation findings into a clear, user-centred problem statement? Understanding

A. Empathise
B. Define
C. Ideate
D. Test

1343. A student stuck on a bug takes a walk and the solution suddenly occurs to them. Which stage of the creative process does this show? Understanding

A. Incubation, followed by illumination
B. Preparation
C. Verification
D. Convergence

1344. A designer lists a product's attributes (power source, material, size) with several options for each, then combines one option from every attribute to generate new designs. What is this technique? Understanding

A. Mind mapping
B. Five Whys
C. Reverse brainstorming
D. Morphological analysis

1345. After solving several puzzles with a long method, students keep using it on a new puzzle that has a much simpler solution. What is this effect called? Understanding

A. Einstellung effect (mental set)
B. Functional fixedness
C. Confirmation bias
D. Groupthink

1346. In supermarket self-checkout, customers scan their own items instead of the cashier. Which SCAMPER prompt best describes this? Understanding

A. Reverse (or rearrange) the roles
B. Combine
C. Put to another use
D. Magnify

1347. In de Bono's lateral thinking, what is a provocation ('Po')? Understanding

A. A criticism that ends a weak idea quickly
B. A deliberately unreasonable statement used as a stepping stone to new ideas
C. A survey of customer opinions
D. A final decision rule for choosing ideas

1348. According to Clayton Christensen, what is typical of a disruptive innovation? Understanding

A. It starts as a simpler, cheaper product for overlooked customers and later improves until it challenges established firms
B. It is always the most technically advanced product on the market
C. It is a small improvement to an existing product for current customers
D. It is any invention protected by a patent

1349. Which example is an incremental rather than a radical innovation? Understanding

A. Moving from film cameras to digital photography
B. Replacing horse-drawn carriages with motor cars
C. Introducing the first web browser
D. Releasing a phone with a slightly better battery than last year's model

1350. A project team values harmony so much that nobody questions the leader's plan, and obvious risks go unspoken. What is this, and what helps? Understanding

A. Brainstorming; add more criticism during idea generation
B. Lateral thinking; use a random word
C. Groupthink; assign a devil's advocate or gather views anonymously before discussion
D. Functional fixedness; use a mind map

1351. In the Lean Startup approach, what is a 'pivot'? Understanding

A. Abandoning the company entirely
B. Adding every feature users request
C. Repeating the same experiment until it succeeds
D. A structured change of strategy, such as a new target user or core feature, based on what experiments revealed

1352. Repeatedly asking 'Why?' about a problem (such as 'the build failed') until a root cause appears is called what? Understanding

A. The Five Whys
B. SCAMPER
C. Six Thinking Hats
D. Random entry

1353. TRIZ, the inventive problem-solving method developed by Genrich Altshuller, is built mainly on which idea? Understanding

A. Ideas should be generated without any structure or rules
B. Inventive problems contain contradictions, and patterns from thousands of patents (such as the 40 inventive principles) show how to resolve them
C. Problems are solved best by surveying customers
D. Creativity is an inborn talent that cannot be learned

1354. What is the key difference between reasoning by analogy and first-principles thinking? Understanding

A. Analogy is always wrong; first principles is always right
B. They are the same process with different names
C. Analogy reuses solutions from similar cases; first principles rebuilds from basic facts, so it can find options no existing case shows
D. First principles copies competitors; analogy starts from physics

1355. Which statement about the five design-thinking stages is accurate? Understanding

A. They must be done strictly once, in order
B. Prototyping should happen only after the product is finished
C. Testing is optional if the team agrees on the idea
D. They are iterative: testing often sends a team back to redefine the problem or ideate again

1356. Research often finds that people brainstorming alone and then pooling ideas produce more ideas than the same number brainstorming together aloud. Which cause is most cited? Understanding

A. Groups always contain less creative people
B. Production blocking: in a spoken group only one person can talk at a time, so others forget or suppress ideas while waiting
C. Individuals are allowed to criticise ideas
D. Groups spend longer on each idea because they are slower to type

1357. The UK Design Council's Double Diamond has four phases. Which order, and which pattern of thinking, is correct? Understanding

A. Define (diverge), Discover (converge), Deliver (diverge), Develop (converge)
B. Discover, Develop, Define, Deliver, all convergent
C. Develop, Deliver, Discover, Define, all divergent
D. Discover (diverge), Define (converge), Develop (diverge), Deliver (converge)

1358. Using the pattern that the sum of the first n odd numbers equals n squared, what is 1 + 3 + 5 + ... + 99? Application

A. 5000
B. 2450
C. 4950
D. 2500

1359. A straight 100-metre fence needs a post every 10 metres, including one at each end. How many posts are needed? Application

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

1360. A patch of algae doubles in area every day and covers a pond completely on day 30. On which day was the pond half covered? Application

A. Day 15
B. Day 20
C. Day 29
D. Day 28

1361. You must guess a secret number from 1 to 1,000 and are told only 'higher' or 'lower' after each guess. Using the halving strategy, what is the most guesses you could need? Application

A. 500
B. 100
C. 10
D. 1,000

1362. Students complain, 'The library is too noisy to study.' Which 'How might we' statement reframes this best for idea generation? Application

A. How might we give students quiet places to focus when they need them?
B. How might we build a new library?
C. How might we punish noisy students?
D. How might we close the library during exams?

1363. A team wants new ideas for a canteen app. Applying SCAMPER's 'Put to another use', which idea fits? Application

A. Remove the payment screen
B. Combine the app with the library catalogue
C. Make the app's buttons bigger
D. Use the app's order history to plan how much food to cook each day

1364. A drawer holds 5 pairs of black gloves and 5 pairs of brown gloves, with left and right gloves distinct. In the dark, what is the minimum number you must take to be sure of a matching pair (same colour, one left and one right)? Application

A. 3
B. 6
C. 21
D. 11

1365. How many times do the hour and minute hands of an analogue clock overlap in 12 hours? Application

A. 12
B. 11
C. 24
D. 22

1366. Instead of counting letters by hand, this function compares sorted letters. What does the program print? Application

```
def is_anagram(a, b):
    return sorted(a) == sorted(b)

print(is_anagram("listen", "silent"),
      is_anagram("apple", "paple"),
      is_anagram("rat", "car"))
```

A. True True False
B. True False False
C. False True False
D. True True True

1367. In a 100 m race, Ali beats Bilal by 10 m, and Bilal beats Chand by 10 m. All three run at constant speeds. By how many metres would Ali beat Chand? Application

A. 20 m
B. 10 m
C. 19 m
D. 18 m

1368. A trader sells to three customers in turn. Each buys half of the eggs he has left plus half an egg, and no egg is ever broken. After the third customer he has none left. How many eggs did he start with? Application

A. 7
B. 8
C. 6
D. 15

1369. How many three-digit numbers (100 to 999) contain at least one digit 7? Use counting by complement. Application

A. 225
B. 270
C. 300
D. 252

1370. A fair die is rolled four times. Using the complement, what is the probability of at least one six, to the nearest percent? Application

A. 67%
B. 52%
C. 48%
D. 17%

1371. This program rotates a list right by k using three reversals instead of shifting one step at a time. What does it print? Application

```
def rotate(a, k):
    k %= len(a)
    a.reverse()
    a[:k] = reversed(a[:k])
    a[k:] = reversed(a[k:])
    return a

print(rotate([1, 2, 3, 4, 5, 6, 7], 3))
```

A. [4, 5, 6, 7, 1, 2, 3]
B. [3, 2, 1, 7, 6, 5, 4]
C. [7, 6, 5, 4, 3, 2, 1]
D. [5, 6, 7, 1, 2, 3, 4]

1372. Every number in the list appears twice except one. Instead of counting with a dictionary, the program XORs all values. What does it print? Application

```
nums = [4, 1, 2, 1, 2]
result = 0
for n in nums:
    result ^= n
print(result)
```

A. 4
B. 0
C. 2
D. 10

1373. Instead of re-adding each window, this sliding-window code updates a running sum. What does it print? Application

```
a = [2, 1, 5, 1, 3, 2]
k = 3
window = sum(a[:k])
best = window
for i in range(k, len(a)):
    window += a[i] - a[i - k]
    best = max(best, window)
print(best)
```

A. 8
B. 9
C. 14
D. 5

1374. This naive recursive Fibonacci counts its own calls. What does it print, and what does that suggest? Application

```
calls = 0

def fib(n):
    global calls
    calls += 1
    if n < 2:
        return n
    return fib(n - 1) + fib(n - 2)

fib(10)
print(calls)
```

A. 11; one call per value of n
B. 89; one call per unit of the result
C. 177; it recomputes the same subproblems, so memoisation would cut calls to about n
D. 55; the value of fib(10)

1375. A quiz show hides a prize behind one of three doors. You pick door 1. The host, who knows where the prize is and always opens an empty door you did not pick, opens door 3. Should you switch to door 2? Application

A. Yes: switching wins with probability 2/3
B. No: staying wins with probability 2/3
C. It makes no difference: both are 1/2
D. Yes: switching wins with probability 1/2

1376. One of 1,000 bottles is poisoned. A test strip shows a result after 24 hours, and you have only 24 hours, so all tests must run at once. What is the minimum number of strips needed to find the bottle? Application

A. 999
B. 500
C. 10
D. 32

1377. 100 lockers are closed. Student 1 toggles every locker, student 2 every 2nd, student 3 every 3rd, and so on up to student 100. How many lockers are open at the end? Application

A. 50
B. 0
C. 10
D. 25

1378. You have two identical eggs and a 100-floor building. You must find the highest floor from which an egg survives a drop. What is the minimum number of drops that always works in the worst case? Application

A. 10
B. 50
C. 14
D. 7

1379. A product team's brainstorms keep producing ideas that copy what competitors already do. Which change is most likely to break the pattern? Analysis

A. Bring in stimuli from other industries, such as asking how a hotel or a game would solve the problem
B. Hold the same brainstorm for twice as long
C. Let the most senior person choose the topic each time
D. Study competitors' products in more detail first

1380. In an idea meeting, the department head shares her favourite solution first. Everyone then agrees with it. What is the best fix for the next meeting? Analysis

A. Let the head speak first but more briefly
B. Collect ideas from everyone individually or anonymously before the head gives her view
C. Hold the meeting without any discussion
D. Vote immediately on the head's idea

1381. A rule says students must visit the office in person to submit a course-change form. Which question best challenges the hidden assumption? Analysis

A. How can we make the office queue move faster?
B. How can we print more copies of the form?
C. Which students submit the most forms?
D. What if the form did not need the student to be physically present?

1382. A canteen has long lunch queues because everyone orders and pays at one counter between 1:00 and 1:20 pm. Which idea tackles the cause rather than the symptom? Analysis

A. Add more chairs in the waiting area
B. Let students pre-order and pay in an app, then collect at a chosen time slot
C. Play music to make the wait more pleasant
D. Put up a sign asking students to be patient

1383. A team rates four ideas for a student portal on an impact/effort matrix. Which idea should it do first? Analysis

A. Idea X: high impact, high effort
B. Idea Y: low impact, low effort
C. Idea W: high impact, low effort
D. Idea Z: low impact, high effort

1384. A team built three polished prototypes of a hostel complaint app but never spoke to hostel residents. Testing shows residents mainly want to know when the problem will be fixed. Which design-thinking stage was skipped? Analysis

A. Prototype
B. Empathise
C. Ideate
D. Test

1385. A hospital wants faster, error-free handovers when patients move from surgery to intensive care. Which source of analogy is most useful? Analysis

A. A Formula 1 pit crew, where each person has one rehearsed task during a fast, high-stakes handover
B. A library's book-return system
C. A social-media feed's ranking algorithm
D. A supermarket's loyalty-card scheme

1386. A student society has no budget and two weeks to run sign-ups for a 300-person event. Which option shows the best creative use of constraints? Analysis

A. Hire developers to build a custom sign-up app
B. Use a free online form linked to a shared spreadsheet and send the link through existing class groups
C. Postpone the event until a budget is approved
D. Print paper sign-up sheets for every classroom

1387. You must check whether any two numbers in an unsorted list differ by exactly k. Which approach is the best creative alternative to comparing every pair? Analysis

A. Sort the list and compare only the first and last elements
B. Compare every pair but stop after the first half of the list
C. Compute the list's mean and check whether it equals k
D. Put all numbers in a hash set, then for each x check whether x + k is in the set: O(n) expected

1388. The puzzle: make four equilateral triangles, all the same size as the sides of one matchstick, using only six matchsticks. Most people fail. Which assumption blocks them? Analysis

A. That the matchsticks cannot be broken in half
B. That the triangles must touch each other
C. That all six matchsticks must be used
D. That the matchsticks must lie flat on the table; a 3D tetrahedron solves it

1389. You must detect whether a singly linked list contains a cycle using O(1) extra memory. Which approach works? Analysis

A. Store every visited node in a hash set
B. Count the nodes; if the count exceeds 1,000 there is a cycle
C. Reverse the list and check whether the head changes
D. Move one pointer one step and another two steps at a time; if they ever meet, there is a cycle

1390. In a game, two players take turns choosing numbers from 1 to 9 without reuse. The first to hold three numbers that sum to 15 wins. Which reframing makes the game easiest to play well? Analysis

A. Always choose the largest available number
B. Write all 84 triples and recheck them every turn
C. Place 1 to 9 in a 3 x 3 magic square; every line sums to 15, so the game becomes noughts and crosses
D. Treat it as a pure luck game with no strategy

1391. You have a 4-minute and a 7-minute sand timer. Which plan measures exactly 9 minutes? Analysis

A. Start both. At 4 min flip the 4. At 7 min flip the 7. At 8 min (4 runs out) flip the 7 again; it runs 1 more minute, ending at 9
B. Start both; when the 7 ends, start the 4 again and stop at its end
C. Run the 4 twice and then add 1 minute by estimate
D. Start the 7, then run the 4 once from when the 7 ends

1392. A study app's MVP gets many sign-ups but week-2 retention is 3%. Interviews show the few loyal users open it only for one feature: timed past-topic quizzes. What is the most evidence-based next step? Analysis

A. Add more features so every user finds something they like
B. Pivot to focus the product on the timed-quiz feature and measure whether retention improves
C. Persevere unchanged, because sign-ups are high
D. Spend more on advertising to raise sign-ups further

1393. There are 25 horses and a track that races 5 at a time. There is no stopwatch; each race only gives finishing order. What is the minimum number of races needed to find the 3 fastest horses? Analysis

A. 6
B. 7
C. 5
D. 11

Real-World Problem Solving MCQs for the NSCT

1394. In project scheduling, what is the critical path? Understanding

A. The chain of tasks with the highest cost
B. The shortest route from the first task to the last
C. The longest chain of dependent tasks, which sets the minimum project duration
D. The list of tasks assigned to the project manager

1395. What does 'slack' (or 'float') mean for a project task? Understanding

A. How long the task can be delayed without delaying the whole project
B. The extra budget reserved for the task
C. The time the task takes when done at full speed
D. The number of people assigned to it

1396. A university must schedule exams so that no student has two exams in the same slot, using as few slots as possible. Which model fits? Understanding

A. Shortest path between two courses
B. Graph colouring: courses are vertices, shared students are edges, colours are slots
C. Sorting courses by enrolment
D. A binary search over room numbers

1397. You want to fit the largest number of non-overlapping talks into one hall. Which greedy rule is proven optimal? Understanding

A. Pick the shortest talks first
B. Pick the talks that start earliest first
C. Repeatedly pick the talk that finishes earliest among those that do not clash with talks already chosen
D. Pick the talks with the fewest clashes first

1398. Given a day's meeting times, what determines the minimum number of meeting rooms needed? Understanding

A. The total number of meetings
B. The length of the longest meeting
C. The number of distinct start times
D. The maximum number of meetings in progress at the same moment

1399. In a production line, what is the bottleneck? Understanding

A. The first stage of the line
B. The stage with the lowest capacity, which limits the output of the whole line
C. The most expensive machine
D. The stage with the most workers

1400. What is the break-even point of a small business? Understanding

A. The point where fixed costs are paid off for ever
B. The month of highest sales
C. The price at which competitors stop selling
D. The sales level at which total revenue equals total cost, so profit is zero

1401. A brief says: 'Minimise travel cost, but the team must arrive by Friday 9 a.m.' What is the Friday deadline? Understanding

A. The objective
B. A constraint
C. An assumption
D. An output

1402. What does a Gantt chart show? Understanding

A. Tasks as bars along a time axis, showing when each starts, how long it lasts and how tasks overlap
B. The organisation's reporting structure
C. The flow of data between system components
D. The cost breakdown of each task as a pie

1403. 'Testing cannot start until coding has finished.' Which task dependency type is this? Understanding

A. Finish-to-start
B. Start-to-start
C. Finish-to-finish
D. Start-to-finish

1404. For a budget-limited choice of whole projects (each taken fully or not at all), when does picking by best benefit-to-cost ratio guarantee the optimal total benefit? Understanding

A. Always, for any set of projects
B. Only when every project has the same benefit
C. Never, in any version of the problem
D. Only when projects can be taken in fractions (the fractional knapsack); for whole projects it can fail

1405. Using the Eisenhower matrix, how should a task that is both urgent and important be handled? Understanding

A. Do it now
B. Schedule it for later
C. Delegate it
D. Drop it

1406. In project management, what does 'crashing' a schedule mean? Understanding

A. Cancelling the project
B. Removing tasks from the critical path
C. Adding resources to critical-path tasks to shorten the project, at extra cost
D. Letting tasks overlap without any extra resources

1407. According to Brooks's law (The Mythical Man-Month), what usually happens when you add programmers to a software project that is already late? Understanding

A. It finishes early in proportion to the number added
B. It becomes later, because new people need ramp-up time and add communication overhead
C. Nothing changes
D. Quality always improves

1408. Little's law links three quantities in a stable queueing system. Which formula states it? Understanding

A. $L = \lambda \times W$ (average number in system = arrival rate × average time in system)
B. $L = \lambda / W$
C. $W = L \times \lambda$
D. $L = \lambda + W$

1409. A project's critical path is 20 days. You finish a task that is not on the critical path two days early. What happens to the project duration? Understanding

A. It drops to 18 days
B. It drops to 19 days
C. It stays 20 days, because the critical path is unchanged
D. It rises, because the schedule is disrupted

1410. Ali can set up a computer lab in 6 hours; Sana can do it in 3 hours. Working together at the same rates, how long will they take? Application

A. 4.5 hours
B. 3 hours
C. 2 hours
D. 9 hours

1411. Bus X leaves the campus every 12 minutes and bus Y every 18 minutes. Both leave together at 8:00. When do they next leave together? Application

A. 8:30
B. 9:00
C. 8:36
D. 8:06

1412. Tasks: A (3 days); B (2 days, after A); C (4 days, after A); D (1 day, after both B and C). What is the minimum project duration? Application

A. 10 days
B. 6 days
C. 7 days
D. 8 days

1413. One hall, seven talks (start-end, 24-hour clock): 9-10, 9-12, 10-11, 11-13, 12-13, 13-15, 14-16. A talk may start exactly when another ends. What is the maximum number of talks that can be held? Application

A. 3
B. 4
C. 5
D. 7

1414. Meetings: 9:00-10:30, 9:30-11:00, 10:00-11:30, 11:00-12:00, 11:30-12:30. A room freed at a given time can be reused at that same time. What is the minimum number of rooms needed? Application

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

1415. A student's printing shop has fixed monthly costs of 60,000 rupees. Each bound report sells for 500 rupees and costs 300 rupees to make. How many reports per month are needed to break even? Application

A. 300
B. 120
C. 200
D. 750

1416. A campus café serves a steady 30 customers per hour, and each customer spends 10 minutes inside on average. By Little's law, how many customers are inside on average? Application

A. 5
B. 3
C. 300
D. 20

1417. Five one-day jobs each earn a profit only if finished by their deadline (day number). Only one job can be done per day. What is the maximum total profit? Application

```
job | deadline | profit
J1  | 2        | 100
J2  | 1        | 19
J3  | 2        | 27
J4  | 1        | 25
J5  | 3        | 15
```

A. 167
B. 142
C. 127
D. 125

1418. An inlet pipe fills a water tank in 4 hours; a leak empties a full tank in 6 hours. With both acting on an empty tank, how long until it is full? Application

A. 10 hours
B. 2.4 hours
C. 12 hours
D. 5 hours

1419. A lab has a budget of 10 units. Each upgrade must be bought whole. Which choice gives the most total benefit within budget? Application

```
upgrade | cost | benefit
P1      | 5    | 10
P2      | 4    | 7
P3      | 6    | 12
P4      | 3    | 5
```

A. P1 + P2, benefit 17
B. P1 + P3, benefit 22
C. P3 + P4, benefit 17
D. P2 + P3, benefit 19

1420. An online viva is set for 14:00 Pakistan Standard Time (UTC+5). What time is that for an examiner on UTC+0? Application

A. 09:00
B. 19:00
C. 14:00
D. 05:00

1421. Five equipment loads weigh 6, 5, 4, 3 and 2 tonnes. Each van carries at most 10 tonnes, and loads cannot be split. What is the minimum number of vans? Application

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

1422. Travel times (minutes) between points are listed below; roads work both ways. What is the shortest time from Home to Uni? Application

```
Home-A 4
Home-B 2
B-A    1
A-C    5
B-C    8
C-Uni  3
A-Uni  10
```

A. 12 minutes
B. 13 minutes
C. 14 minutes
D. 11 minutes

1423. A three-stage form-processing line handles 40, 25 and 30 forms per hour at stages 1, 2 and 3. How many forms can it complete in an 8-hour shift? Application

A. 320
B. 760
C. 200
D. 240

1424. Tasks: A (2 days); B (4, after A); C (3, after A); D (2, after C); E (3, after B and D). Which task has slack, and how much? Analysis

A. C, with 1 day of slack
B. B, with 2 days of slack
C. No task has slack
D. B, with 1 day of slack

1425. Path A → B → D takes 10 days and path A → C → D takes 9 (A = 3, B = 5, C = 4, D = 2). Crash cost per day saved: A 500, B 300, C 100, D 400 (thousand rupees). What is the cheapest way to finish one day earlier? Analysis

A. Shorten C by one day, costing 100
B. Shorten B by one day, costing 300
C. Shorten D by one day, costing 400
D. Shorten A by one day, costing 500

1426. A university café has long queues only between 1:00 and 1:20 p.m., when every department breaks for lunch at once. Which change targets the root cause at the lowest cost? Analysis

A. Build a second café
B. Hire three extra cashiers for the whole day
C. Raise prices so fewer students buy lunch
D. Stagger department lunch breaks by 15-20 minutes

1427. A token machine dispenses 1, 3 and 4-token coins. Its greedy rule always gives the largest coin first. For 6 tokens, how does greedy compare with the best possible answer? Analysis

A. Greedy gives the optimum of 2 coins
B. Greedy gives 3 coins (4 + 1 + 1); the optimum is 2 coins (3 + 3)
C. Greedy and the optimum both give 3 coins
D. Greedy gives 2 coins but the optimum is 1 coin

1428. A final-year project is two weeks behind. The supervisor proposes adding five new students who have never seen the code. Which is the strongest concern? Analysis

    A. Five students are too few to make any difference
    B. New students always write better code than the original team
    C. Adding people to a project has no costs at all
    D. Training and coordinating newcomers will absorb the existing team's time, so the project may slip further (Brooks's law)

1429. A lab technician must run four print jobs taking 8, 2, 5 and 1 minutes, one at a time. Which order minimises the average completion time, and what is that average? Analysis

    A. Given order (8, 2, 5, 1): average 12.25 minutes
    B. Longest first (8, 5, 2, 1): average 13 minutes
    C. Shortest first (1, 2, 5, 8): average 7 minutes
    D. Any order: the average is always 4 minutes

1430. A farmer must ferry a wolf, a goat and a cabbage across a river. The boat holds the farmer plus one item. Left alone, the wolf eats the goat and the goat eats the cabbage. What is the minimum number of crossings? Analysis

    A. 5
    B. 3
    C. 7
    D. 9

1431. Five courses A-E share students as listed. Courses that share students cannot have exams in the same slot. What is the minimum number of exam slots? Analysis

```
clashes: A-B, A-C, B-C, C-D, D-E, B-D
```

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

1432. A client asks for 'a booking system ready by Monday' and also 'full payment integration'. Your team cannot deliver both by Monday. What is the best first step? Analysis

    A. Explain the trade-off to the client and agree which matters more: the Monday date or payment integration
    B. Quietly skip payment integration
    C. Promise both and work through the weekend hoping for the best
    D. Start coding at random and see what gets finished

1433. A society compares two options for its web app over 3 years: its own server (300 thousand rupees up front plus 5 thousand per month) or cloud hosting (15 thousand per month). Which statement is correct? Analysis

    A. Cloud is cheaper over 36 months, 540 vs 480 thousand
    B. Costs are equal at month 30; over 36 months the own server is cheaper, 480 vs 540 thousand
    C. The own server only becomes cheaper after month 60
    D. Both options cost the same over 36 months

1434. In an optimisation problem, what is a feasible solution? Understanding

    A. The solution with the best possible value of the objective
    B. Any solution that the client finds attractive
    C. One that satisfies every stated constraint, whether or not it is the best
    D. A solution that meets the objective but may break one constraint

1435. For choosing the most non-overlapping talks for one hall, why is 'pick the shortest talk first' not a safe greedy rule? Understanding

A. Short talks always start earliest, so the rule is the same as earliest start
B. It is optimal, but it is slower to compute than earliest finish
C. It only fails when two talks have exactly the same length
D. A short talk can overlap two longer talks that could both have been chosen, so it may block a larger set

1436. In problem solving, what does the Pareto principle (80/20 rule) suggest? Understanding

A. 80% of a project's budget should go on the final 20% of tasks
B. A solution is acceptable once it is 80% correct
C. Every problem should be split into exactly five equal parts
D. A small number of causes often produce most of the problems, so fix those first

1437. When packing loads into identical vans, what does (total load) ÷ (van capacity), rounded up, tell you? Understanding

A. The exact minimum number of vans in every case
B. An upper bound: you never need more vans than this
C. A lower bound: no packing can use fewer vans than this
D. The number of vans the first-fit method will use

1438. A production line's slowest stage is assembly. What happens to the line's output if the faster painting stage is made even faster? Understanding

A. Nothing, because output is still limited by the assembly stage
B. Output rises by the same amount as the painting improvement
C. Output rises by half the painting improvement
D. Output falls, because work piles up faster

1439. In a project schedule, what is a milestone? Understanding

A. The longest task in the project
B. A zero-duration point that marks the completion of a key deliverable or phase
C. A task that has the most slack
D. A weekly progress meeting

1440. In PERT, which formula gives the expected duration of a task from its optimistic (O), most likely (M) and pessimistic (P) estimates? Understanding

A. (O + M + P) / 3
B. (O + 2M + P) / 4
C. (O + 4M + P) / 6
D. (P - O) / 6

1441. In the critical path method, how is a task's total float (slack) calculated? Understanding

A. Earliest finish minus earliest start
B. Project duration minus the task's duration
C. Latest start minus earliest start (equivalently, latest finish minus earliest finish)
D. Latest finish of the task minus the earliest start of the project

1442. How does a task's free float differ from its total float? Understanding

A. Free float applies only to critical tasks; total float applies only to non-critical tasks
B. Free float is the delay possible without delaying any successor's earliest start; total float is the delay possible without delaying the project
C. Free float is always larger than total float
D. They are two names for the same quantity

1443. The project 'triple constraint' (iron triangle) links which three factors? Understanding

A. People, process and tools
B. Risk, quality and testing
C. Scope, time and cost
D. Design, code and deployment

1444. What is resource levelling in scheduling? Understanding

A. Adding extra staff to critical tasks to finish sooner
B. Running sequential tasks in parallel to save time
C. Giving every team member the same number of tasks
D. Shifting task start times so no resource is over-allocated, which may lengthen the project

1445. What does fast-tracking a project schedule mean? Understanding

A. Doing tasks in parallel that were planned in sequence, which saves time but adds risk
B. Adding more resources to critical tasks at extra cost
C. Removing tasks from the project scope
D. Moving the deadline earlier without changing the plan

1446. In a SMART goal, what does the 'M' stand for? Understanding

A. Measurable
B. Manageable
C. Minimal
D. Motivating

1447. A courier must visit 12 shops once each and return to the depot by the shortest total route. Which classic problem is this, and what does it imply? Understanding

A. The travelling salesman problem; it is NP-hard, so large cases are usually solved with heuristics
B. The shortest path problem; Dijkstra's algorithm solves it exactly in polynomial time
C. The minimum spanning tree problem; Kruskal's algorithm solves it exactly
D. The activity selection problem; earliest finish time solves it

1448. A university wants to connect all its buildings with fibre cable using the least total cable length. Which model fits? Understanding

A. Shortest path from one building
B. Travelling salesman tour
C. Graph colouring
D. Minimum spanning tree

1449. A navigation app uses Dijkstra's algorithm for shortest routes. Which condition must the road weights meet for it to give correct answers? Understanding

A. All weights must be whole numbers
B. All weights must be equal
C. No edge weight may be negative
D. The graph must have no cycles

1450. Four lab assistants must each be given exactly one of four duties, and each assistant-duty pair has a cost. Minimising the total cost is which classic problem? Understanding

A. The 0/1 knapsack problem
B. The assignment problem, solved exactly by the Hungarian algorithm
C. The bin packing problem
D. The interval partitioning problem

1451. A student turns down a 30,000-rupee paid internship to take an unpaid course. In decision-making terms, what is the 30,000 rupees? Understanding

A. The opportunity cost of taking the course
B. A sunk cost
C. A fixed cost of the course
D. The break-even point

1452. A department has spent 2 million rupees on software that still does not work. How should that 2 million affect the decision to continue? Understanding

A. It should decide the matter: continue so the money is not wasted
B. It should be added to the cost of every alternative
C. It should be divided across the remaining months of the project
D. It should not; it is a sunk cost, so compare only future costs and benefits

1453. One machine processes jobs one at a time, and each job has a deadline. Which ordering rule minimises the maximum lateness? Understanding

A. Shortest processing time first
B. Longest processing time first
C. The order in which the jobs arrived
D. Earliest deadline first

1454. What does a weighted decision matrix do? Understanding

A. Scores each option against criteria, multiplies by each criterion's importance and adds them up to compare options
B. Lists the steps of a process in time order
C. Shows which tasks depend on which others
D. Records who is responsible for each decision

1455. In a single-server queue, customers arrive at rate λ and are served at rate μ. What must hold for the queue to stay stable over time? Understanding

A. $\lambda > \mu$, so the server is always busy
B. $\lambda = \mu$ exactly, so arrivals and services balance
C. μ must be at least double λ
D. $\lambda < \mu$, so utilisation $\rho = \lambda/\mu$ is below 1

1456. Three volunteers can each pack all the exam papers alone in 4, 6 and 12 hours respectively. Working together at those rates, how long will they take? Application

A. 2 hours
B. 7.3 hours
C. 3 hours
D. 4 hours

1457. A canteen stall has fixed monthly costs of 48,000 rupees. Each meal sells for 200 rupees and costs 120 rupees to make. How many meals a month does it need to break even? Application

A. 240
B. 600
C. 400
D. 150

1458. A line has three stages: stage 1 has 2 machines at 15 forms per hour each, stage 2 has 1 machine at 20 per hour, and stage 3 has 3 machines at 10 per hour each. How many forms can it finish in a 6-hour shift? Application

A. 60
B. 120
C. 180
D. 90

1459. A task's estimates are: optimistic 4 days, most likely 6 days, pessimistic 14 days. What is its PERT expected duration? Application

A. 8 days
B. 6 days
C. 9 days
D. 7 days

1460. Tasks: A (4 days); B (3, after A); C (6, after A); D (2, after B); E (5, after C and D). What is the minimum project duration? Application

A. 14 days
B. 20 days
C. 11 days
D. 15 days

1461. Meetings: 8:00-9:00, 8:30-10:00, 9:00-9:30, 9:15-11:00, 10:00-10:30, 10:15-11:15. A room freed at a given time can be reused at that same time. What is the minimum number of rooms? Application

A. 2
B. 4
C. 6
D. 3

1462. One hall, eight talks (start-end, 24-hour clock): 8-10, 9-10, 9-11, 10-12, 11-13, 12-14, 13-15, 14-16. A talk may start when another ends. What is the maximum number of talks? Application

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

1463. On average 12 students are inside a fee office, and students arrive at 4 per minute. By Little's law, how long does a student spend there on average? Application

A. 48 minutes
B. 16 minutes
C. 0.33 minutes
D. 3 minutes

1464. Visitors arrive at a library desk at 2 per minute and each spends 15 minutes in the queue and at the desk. By Little's law, how many visitors are there on average? Application

A. 30
B. 7.5
C. 17
D. 0.13

1465. A lab has 15 budget units. Upgrades must be bought whole: A costs 9 (benefit 16), B costs 6 (11), C costs 5 (9), D costs 4 (8), E costs 3 (5). What is the maximum total benefit? Application

A. 27, with A + B
B. 25, with A + C
C. 28, with B + C + D
D. 29, with A + B + E

1466. Four print jobs take 6, 3, 9 and 2 minutes and run one at a time. Run shortest first, what is the average waiting time before a job starts? Application

A. 9.5 minutes
B. 5 minutes
C. 4.5 minutes
D. 11 minutes

1467. An assignment portal closes at 23:59 UTC on 10 March. What is that in Pakistan Standard Time (UTC+5)? Application

A. 18:59 on 10 March
B. 23:59 on 10 March
C. 04:59 on 11 March
D. 04:59 on 10 March

1468. Printer X prints 40 pages per minute and printer Y 60 pages per minute. A 1,500-page job is shared so both run together until it is done. How long does it take? Application

A. 15 minutes
B. 25 minutes
C. 30 minutes
D. 37.5 minutes

1469. A society scores three vendors (out of 10) on cost, speed and support, with weights 0.5, 0.3 and 0.2. P scores 9, 6, 5; Q scores 5, 9, 9; R scores 8, 7, 7. Which vendor wins, and with what weighted score? Application

A. R, with 7.5
B. Q, with 23
C. P, with 7.3
D. P and R tie at 7.3

1470. Loads of 7, 5, 4, 4, 3, 2 and 1 tonnes must go in vans that carry at most 10 tonnes each. Loads cannot be split. What is the minimum number of vans? Application

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

1471. Tasks: A (3 days); B (4, after A); C (2, after A); D (5, after C); E (1, after B and D). Counting days from the project start (day 0), what is the latest day B can start without delaying the project? Application

A. Day 3
B. Day 7
C. Day 10
D. Day 6

1472. Cable costs between buildings: A-B 4, A-C 3, B-C 2, B-D 5, C-D 6, C-E 7, D-E 3. What is the minimum total cost to connect all five buildings? Application

A. 13
B. 12
C. 15
D. 30

1473. Jobs (duration, deadline in days): J1 (3, 4), J2 (2, 6), J3 (4, 9), J4 (1, 3). They run one after another from day 0. Scheduled by earliest deadline first, what is the maximum lateness? Application

A. 0 days
B. 2 days
C. 1 day
D. 3 days

1474. A computer lab has 40 PCs. Each of 250 students needs one PC for a 2-hour practical, and sessions cannot overlap. What is the minimum number of sessions? Application

A. 6
B. 7
C. 6.25
D. 8

1475. A help desk is modelled as an M/M/1 queue: 8 requests arrive per hour and the single agent can serve 10 per hour. Using $L = \rho/(1 - \rho)$, what is the average number of requests in the system? Application

A. 4
B. 0.8
C. 5
D. 2

1476. Three workers must each take one different job. Costs: W1 (J1 9, J2 2, J3 7), W2 (J1 6, J2 4, J3 3), W3 (J1 5, J2 8, J3 1). What is the minimum total cost? Application

A. 10
B. 14
C. 9
D. 6

1477. 300 handouts must be ready by 9 a.m. at minimum cost. Shop P charges 9,000 rupees but delivers at 11 a.m.; Q charges 12,000 and delivers at 8 a.m.; R charges 15,000 and delivers at 7 a.m.; S prints 200 by 8 a.m. for 7,000. Which shop should be chosen? Analysis

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

1478. Talks: 9-17, 10-11, 11-12, 12-13. A student picks the talk that starts earliest, then repeats with talks that do not overlap. How does this compare with the optimum? Analysis

A. Greedy picks 3 talks, which is optimal
B. Greedy picks 1 talk; the optimum is 3
C. Greedy picks 2 talks; the optimum is 3
D. Greedy picks 1 talk, which is optimal

1479. A rider starts and ends at depot S and visits A, B and C once. Distances: S-A 1, S-B 2, S-C 3, A-B 4, A-C 2, B-C 9. Using nearest neighbour (always go to the closest unvisited point), how does the route compare with the best? Analysis

A. Nearest neighbour gives 11, which is optimal
B. Nearest neighbour gives 14; the best tour is 11
C. Nearest neighbour gives 14, which is optimal
D. Nearest neighbour gives 12; the best tour is 11

1480. Lab printers jam every morning but rarely in the afternoon. The paper is stored overnight beside a leaking air conditioner. Which action addresses the root cause? Analysis

A. Buy two extra printers for the morning
B. Assign a technician to clear jams every morning
C. Ask students to print only in the afternoon
D. Store the paper in a dry cupboard away from the air conditioner

1481. Path 1 takes 15 days and is critical; path 2 takes 14 days. A manager pays to shorten a task that is only on path 1 by 3 days. What is the new project duration? Analysis

A. 12 days
B. 15 days, because nothing changes
C. 14 days, so only 1 of the 3 days paid for is gained
D. 13 days

1482. After a university launched an online complaint form, recorded hostel complaints rose from 20 to 90 a month. What is the most reasonable conclusion? Analysis

A. Hostel conditions became much worse after the form was launched
B. The form should be removed to reduce complaints
C. Complaints are random, so the data cannot be used
D. The rise may reflect easier reporting, so compare like-for-like data before deciding hostel conditions got worse

1483. Meetings: X 9:00-12:00, Y 9:30-10:30, Z 11:00-12:30. A student says 3 rooms are needed because X overlaps both Y and Z. What is the correct minimum, and why? Analysis

A. 3, because X overlaps two other meetings
B. 2, because Y and Z never overlap each other, so at most two meetings run at once
C. 1, because Y and Z fit inside X's time
D. 2, because X ends before Z

1484. A line's stages process 50, 30 and 35 units per hour. Stage 2 is upgraded from 30 to 45 per hour. What is the line's new output? Analysis

A. 45 per hour
B. 50 per hour
C. 30 per hour, because nothing changes
D. 35 per hour, because stage 3 becomes the bottleneck

1485. A tuition centre has fixed costs of 90,000 rupees a month. Each student pays 400 and costs 250 to teach. If the fee rises to 450, which statement is correct? Analysis

A. Break-even falls from 600 to 533 students, in proportion to the fee
B. Break-even falls from 225 to 200 students
C. Break-even falls from 600 to 450 students, a 25% drop from a 12.5% fee rise
D. Break-even does not change, because fixed costs are the same

1486. A society needs a venue that holds at least 120 people, costs at most 50,000 rupees and is free on Friday. Venue 1: 150 seats, 45,000, free Friday. Venue 2: 200 seats, 40,000, booked Friday. Venue 3: 100 seats, 30,000, free Friday. Venue 4: 130 seats, 55,000, free Friday. Which venue should be booked? Analysis

A. Venue 1
B. Venue 2
C. Venue 3
D. Venue 4

1487. 2 million rupees has been spent on an old system. Finishing it costs another 1.5 million and gives benefits worth 1 million. A new system costs 1 million and gives benefits worth 1.8 million. Which choice is better? Analysis

A. Build the new system: net gain 0.8 million versus a net loss of 0.5 million
B. Finish the old system, so the 2 million already spent is not wasted
C. Finish the old system, because its total cost of 3.5 million shows its value
D. Do nothing, because both options lose money once the 2 million is counted

1488. A Lahore team (UTC+5) and a London team (UTC+0, winter) both work 9:00-17:00 local time. How long is their shared working window, and when is it in Lahore? Analysis

A. 8 hours, 9:00-17:00 in Lahore
B. 3 hours, 14:00-17:00 in Lahore
C. 3 hours, 9:00-12:00 in Lahore
D. 5 hours, 12:00-17:00 in Lahore

1489. Vendor A quotes 100,000 rupees with a 20% chance of a late-delivery penalty of 50,000. Vendor B quotes 105,000 with a 5% chance of the same penalty. On expected cost, which is better? Analysis

A. B: 107,500 versus A's 110,000
B. A: its quote is 5,000 lower
C. A: 100,000 versus B's 107,500
D. They are equal at 110,000 each

1490. A month's printer faults: paper jam 45, toner 25, network 15, driver 10, other 5. What is the fewest categories the team must fix to remove at least 70% of faults? Analysis

A. 1 (paper jam)
B. 3 (paper jam, toner and network)
C. 2 (paper jam and toner)
D. 4 (all except other)

1491. Students complain that the university results website is 'slow'. What should the team do first? Analysis

A. Buy a more powerful server straight away
B. Rewrite the website in a faster framework
C. Measure page load times by page and time of day to find where and when it is slow
D. Ask students to use the site late at night

Communication & Documentation of Solutions MCQ...

1492. Which of these is a functional requirement for a student portal? Understanding

A. The system shall load every page within 2 seconds
B. The system shall be available 99.5% of the time
C. The system shall use colours that meet accessibility contrast guidelines
D. The system shall let a student reset their password through a link sent to their university email

1493. Which of these is a non-functional requirement? Understanding

A. A teacher shall be able to upload marks as a CSV file
B. A student shall be able to view their timetable
C. 95% of search requests shall return results within 1 second
D. An admin shall be able to deactivate an account

1494. Which template is the standard format for an agile user story? Understanding

A. The system shall <function> within <time>
B. Given <context>, when <action>, then <result>
C. If <condition> then <action> else <action>
D. As a <type of user>, I want <goal>, so that <benefit>

1495. What is the purpose of the Given/When/Then format? Understanding

A. To describe the system's class hierarchy
B. To write acceptance criteria as testable scenarios: starting context, action and expected outcome
C. To record who approved a requirement
D. To list the steps of an algorithm in pseudocode

1496. According to common style guides, what should a good code comment mainly explain? Understanding

A. Exactly what each line does, repeating the code in words
B. Why the code does something non-obvious: intent, reasons and constraints
C. The author's name and the date for every line
D. Nothing: good code never needs comments

1497. In Python, what is a docstring? Understanding

A. Any comment that starts with #
B. A string literal written as the first statement of a module, class or function, readable through help() or __doc__
C. A string stored in a separate documentation file
D. A variable that holds the function's name

1498. Why is 'The app shall be fast and user-friendly' a weak requirement? Understanding

A. It is too short to be a requirement
B. Requirements must never mention users
C. It describes a functional requirement in the wrong format
D. It cannot be verified, because 'fast' and 'user-friendly' have no measurable criteria

1499. What is the main purpose of a Software Requirements Specification (SRS)? Understanding

A. To store the final source code
B. To list the project team's salaries
C. To replace all communication with the client
D. To record an agreed, complete description of what the system must do, used as the basis for design, testing and acceptance

1500. A function's documentation says: 'Requires: list is non-empty. Ensures: returns the largest element.' What are these two statements called? Understanding

A. A precondition and a postcondition
B. An input and a variable
C. A test case and a bug report
D. An invariant and a comment

1501. Which content is most essential in a project's README file? Understanding

A. The full history of every commit
B. Every team member's CNIC number
C. What the project does, how to install and run it, and a basic usage example
D. The complete source code pasted inline

1502. Under Semantic Versioning (MAJOR.MINOR.PATCH), which part do you increase when you make a change that breaks backward compatibility? Understanding

A. MINOR
B. MAJOR
C. PATCH
D. None; the version stays the same

1503. Which detail is most essential in a useful bug report? Understanding

A. Exact steps to reproduce, with the expected and the actual result
B. The reporter's opinion of the developer
C. A request to fix it urgently, without details
D. A screenshot of the login page only

1504. Which UML diagram shows the order of messages exchanged between objects over time? Understanding

A. Sequence diagram
B. Class diagram
C. Use case diagram
D. Deployment diagram

1505. What is a requirements traceability matrix used for? Understanding

A. Calculating the project budget
B. Listing team members' working hours
C. Linking each requirement to the design elements and test cases that cover it
D. Tracking server uptime

1506. What does an Architecture Decision Record (ADR) capture? Understanding

A. Every line of code changed in a release
B. The context, the decision taken, and its consequences for one significant design choice
C. The minutes of every daily stand-up
D. The user interface colour palette

1507. In requirement documents that follow RFC 2119-style keywords, how do 'shall'/'must' and 'should' differ? Understanding

A. They mean exactly the same thing
B. 'Shall' or 'must' marks a mandatory requirement; 'should' marks a recommended one that may be skipped for good reason
C. 'Should' is mandatory and 'shall' is optional
D. 'Shall' is only used for non-functional requirements

1508. A function contains: total = price * 1.17. A reviewer says it is hard to understand. Which change improves it most? Application

A. Define SALES_TAX_RATE = 0.17 once, with a comment naming its source, and write total = price * (1 + SALES_TAX_RATE)
B. Add the comment '# multiply price by 1.17'
C. Rename price to p
D. Replace 1.17 with 117 / 100

1509. Which rewrite turns 'The system should be user-friendly' into a testable requirement? Application

A. The system shall be very user-friendly
B. The system shall be as user-friendly as possible
C. Users shall like the system
D. A first-time user shall complete course registration in under 3 minutes without help, in 9 of 10 test sessions

1510. Which docstring describes this function accurately? Application

```
def avg_positive(nums):
    pos = [n for n in nums if n > 0]
    if not pos:
        return 0
    return sum(pos) / len(pos)
```

A. Return the mean of all values in nums.
B. Return the mean of the values in nums that are greater than 0; return 0 if there are none.
C. Return the mean of the non-negative values in nums, including zeros.
D. Return the mean of the positive values; raise ValueError if there are none.

1511. You fixed a bug where the last page of search results showed one item fewer than it should. Which commit message is most useful? Application

A. Fix off-by-one error that dropped the last item on the final results page
B. fixed stuff
C. Update search.py
D. Changes made after discussion with Ali on Tuesday

1512. Requirement: an account locks after 3 consecutive failed logins. Which acceptance criterion states it correctly? Application

A. Given a locked account, when the user waits, then it is fast
B. Given the login page, when a user logs in, then they are happy
C. Given an account with 2 consecutive failed logins, when a 3rd wrong password is entered, then the account is locked and a lock message is shown
D. When 3 logins happen, then something is locked

1513. A reviewer reads this function. What should be done? Application

```
def fetch_with_retry(fetch):
    # Try three times before giving up
    for attempt in range(5):
        if fetch():
            return True
    return False
```

A. Nothing, because comments are ignored by Python
B. The comment says three attempts but the loop makes five; confirm the intended number and make code and comment agree
C. Delete the loop, because the comment explains the behaviour
D. Change range(5) to range(3) without checking with anyone

1514. A library is at version 2.4.7. You add a new, backward-compatible function and change nothing else. What is the next version under Semantic Versioning? Application

A. 2.4.8
B. 2.5.0
C. 3.0.0
D. 2.5.7

1515. A library at version 3.2.1 releases one backward-compatible new feature and two bug fixes together. What is the correct new version? Application

A. 3.3.0
B. 3.2.3
C. 4.0.0
D. 3.3.2

1516. Your release will slip by one week because a payment provider changed its API. Which message to a non-technical manager is best? Application

A. The REST endpoints changed their JSON schema, so our serializers throw 422s.
B. There is a delay. Details later.
C. It is not our fault; the provider broke everything.
D. Release moves from 10 to 17 March because our payment provider changed its system. Option: launch on the 10th without online payments and add them on the 17th.

1517. Which pseudocode correctly documents this function? Application

```
def find(items, target):
    for i in range(len(items)):
        if items[i] == target:
            return i
    return -1
```

A. FOR i FROM 0 TO length - 1: IF items[i] = target THEN RETURN i. After the loop, RETURN -1.
B. FOR i FROM 0 TO length - 1: IF items[i] = target THEN RETURN i ELSE RETURN -1.
C. Count how many items equal target and RETURN the count.
D. FOR i FROM length - 1 DOWN TO 0: IF items[i] = target THEN RETURN i. RETURN -1.

1518. An API document describes parse_date(text): its parameter and its return value. What important item is missing? Application

A. The developer's favourite editor
B. How many lines the function has
C. The date the function was written
D. What happens on invalid input, for example 'Raises ValueError if text is not in YYYY-MM-DD format'

1519. Which precondition must the documentation of this function state? Application

```
def binary_search(a, x):
    lo, hi = 0, len(a) - 1
    while lo <= hi:
        mid = (lo + hi) // 2
        if a[mid] == x:
            return mid
        if a[mid] < x:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1
```

A. a must be sorted in ascending order
B. a must contain no negative numbers
C. x must be present in a
D. a must have an even length

1520. Which bug report is most useful to a developer? Application

A. The results page is broken.
B. Fix the website ASAP!!!
C. On Chrome 131, open /results, filter by 'CS', click page 3: expected 20 rows, got 19. Happens every time.
D. Something went wrong yesterday, not sure what.

1521. What is wrong with this function's docstring? Application

```
def count_zero_pairs(a):
    """Count pairs i < j with a[i] + a[j] == 0. Runs in O(n) time."""
    c = 0
    for i in range(len(a)):
        for j in range(i + 1, len(a)):
            if a[i] + a[j] == 0:
                c += 1
    return c
```

A. Nothing; the function is O(n)
B. It claims O(n) time, but the nested loops make n(n - 1)/2 comparisons, which is $O(n^2)$
C. It should claim O(log n)
D. Docstrings must never mention complexity

1522. A requirement reads: 'The app shall show the last 10 transactions sorted by date.' Which clarifying question matters most before building it? Analysis

A. Which font should the dates use?
B. Should the team use Git or another version-control tool?
C. How many developers will work on it?
D. Should the newest transaction be first or last, and does 'last 10' mean the 10 most recent?

1523. An SRS contains R4: 'All exports shall be in PDF format' and R9: 'Finance users shall export transactions to Excel.' What should the analyst do? Analysis

A. Implement R4 and ignore R9, because it comes first
B. Implement whichever is easier
C. Implement both by generating a PDF and renaming it .xlsx
D. Record the conflict, raise it with the stakeholders, and update the SRS with the agreed resolution

1524. Which call shows that this code does not match its docstring? Analysis

```
def is_prime(n):
    """Return True if n is a prime number."""
    for d in range(2, n):
        if n % d == 0:
            return False
    return True
```

A. is_prime(1)
B. is_prime(2)
C. is_prime(9)
D. is_prime(13)

1525. A file has a comment on every line, such as 'i = i + 1 # add one to i' and 'return total # return the total'. What is the best review advice? Analysis

    A. Remove comments that repeat the code, keep or add comments that explain intent or non-obvious choices, and use clear names
    B. Add even more comments so nothing is missed
    C. Delete every comment in the project
    D. Move all comments into one large block at the top of the file

1526. After a = [3, 1, 2] and b = sorted_copy(a), which statement is true? Analysis

```
def sorted_copy(items):
    """Return a new list in ascending order; items is not changed."""
    items.sort()
    return items
```

    A. a is still [3, 1, 2] and b is a new sorted list, as documented
    B. b is None, because list.sort() returns None
    C. a is now [1, 2, 3] and b is the same list object, so the docstring is wrong on both counts
    D. The call raises an error, because a list cannot be sorted in place

1527. In a technical interview you are asked to explain your solution to a problem. Which structure communicates it best? Analysis

    A. Start typing code immediately and explain nothing
    B. List every algorithm you know before choosing one
    C. Restate the problem and assumptions, describe the approach, walk through a small example, then give time/space complexity and edge cases
    D. Give only the final complexity, with no approach

1528. A traceability matrix shows that requirement R7, 'Passwords shall be stored hashed', has no linked test case. What is the main risk? Analysis

    A. The project will run out of budget
    B. R7 must be deleted from the SRS
    C. R7 may be unimplemented or broken without anyone noticing, because nothing verifies it
    D. Other requirements will stop working

1529. The team's internal changelog says: 'Refactored AuthService; migrated session store to Redis; fixed NPE in TokenRefresher.' Which version suits release notes for students? Analysis

    A. Refactored AuthService; migrated session store to Redis; fixed NPE in TokenRefresher.
    B. Various improvements.
    C. You now stay signed in more reliably, and a bug that sometimes logged you out early is fixed.
    D. We changed the code.

1530. You need to document how a login request passes from the browser to the web server, then to the database and back, in time order. Which diagram fits best? Analysis

    A. A UML class diagram
    B. A pie chart
    C. A UML sequence diagram
    D. An entity-relationship diagram

1531. Your algorithm for a fee-reconciliation script loads the whole CSV into memory and assumes each student ID appears once. What should its documentation include? Analysis

    A. Only the function name
    B. A promise that it works for any input
    C. The number of hours spent writing it
    D. The assumptions (unique IDs, file fits in memory) and what happens if they are broken

1532. What does a UML use case diagram mainly show? Understanding

A. The classes in the code and their attributes
B. The order of messages between objects over time
C. The physical servers the software is deployed on
D. The actors who use a system and the goals (use cases) they achieve with it

1533. Which UML diagram documents the classes of a system with their attributes, operations and relationships? Understanding

A. Activity diagram
B. Use case diagram
C. Class diagram
D. Sequence diagram

1534. In a standard flowchart, which symbol represents a yes/no decision such as 'marks >= 50?' Understanding

A. A parallelogram
B. A rectangle
C. A diamond
D. An oval

1535. In a standard flowchart, which symbol is used for 'Read student marks' or 'Print result'? Understanding

A. A diamond
B. An oval
C. A parallelogram
D. A rectangle

1536. Under Semantic Versioning, which list is in correct order of precedence, lowest to highest? Understanding

A. 1.0.0 < 1.0.0-alpha < 1.0.0-beta < 1.0.1
B. 1.0.0-alpha < 1.0.0-beta < 1.0.0 < 1.0.1
C. 1.0.0-beta < 1.0.0-alpha < 1.0.0 < 1.0.1
D. 1.0.0-alpha < 1.0.0 < 1.0.0-beta < 1.0.1

1537. According to the Semantic Versioning specification, what does a version starting with 0 (such as 0.4.2) signal? Understanding

A. Initial development: the public API may change at any time and should not be considered stable
B. The software has no bugs left to fix
C. The project has been discontinued
D. Only patch releases are allowed from now on

1538. In the Conventional Commits format used with Semantic Versioning, which commit type corresponds to a MINOR version bump? Understanding

A. fix:
B. feat:
C. docs:
D. chore:

1539. Widely used Git guidance recommends writing a commit's subject line in the imperative mood. Which subject follows that convention? Understanding

A. Added input validation to the signup form
B. Adding input validation to the signup form
C. Add input validation to the signup form
D. I have added input validation to the signup form

1540. What is the main purpose of a project's CHANGELOG file? Understanding

A. To store the full output of git log
B. To list every developer who worked on the project
C. To record the project's installation steps
D. To give users a human-readable list of notable changes in each release, grouped by type such as Added, Changed and Fixed

1541. In program documentation, what is a loop invariant? Understanding

A. A variable whose value never changes inside the loop
B. The number of times the loop will run
C. A condition that is true before the loop starts and remains true after every iteration
D. A comment placed at the end of a loop

1542. How does a Python docstring differ from an ordinary # comment? Understanding

A. A docstring is ignored by Python, while a # comment is stored with the function
B. A docstring must contain code, while a comment must contain text
C. A docstring is stored with the object and can be read at runtime with help() or __doc__; a # comment is discarded
D. There is no difference; they are interchangeable

1543. In Javadoc, which tag documents an exception that a method can raise? Understanding

A. @param
B. @throws
C. @return
D. @see

1544. A Python function is declared def grade(marks: int) -> str. What do these type hints do when the program runs? Understanding

A. They make Python raise a TypeError if marks is not an int
B. They automatically convert marks to an int
C. They document the intended types for readers and tools, but the interpreter does not enforce them
D. They make the function run faster

1545. In MoSCoW prioritisation of requirements, what do the four letters M, S, C and W stand for? Understanding

A. Major, Small, Critical, Weak
B. Mandatory, Secure, Correct, Working
C. Must, Shall, Can, Will
D. Must have, Should have, Could have, Won't have (this time)

1546. In the INVEST checklist for good user stories, what does the 'T' stand for? Understanding

A. Testable
B. Technical
C. Timed
D. Traceable

1547. In Scrum, how does the Definition of Done differ from a story's acceptance criteria? Understanding

A. The Definition of Done is a shared quality standard for every item, while acceptance criteria are specific to one story
B. They are the same list under two names
C. Acceptance criteria apply to the whole sprint, while the Definition of Done applies to one story
D. The Definition of Done is written by the customer, while acceptance criteria are written by testers only

1548. In the Diátaxis documentation framework, what is a 'how-to guide'? Understanding

A. A learning lesson that takes a beginner through a first project
B. A complete technical description of every function and parameter
C. Steps to accomplish a specific task for a user who already knows the basics
D. A discussion of background and design reasons

1549. What does 'self-documenting code' mainly rely on? Understanding

A. Clear names and simple structure so the code shows what it does without extra comments
B. Tools that write comments automatically
C. A comment on every line
D. Very short variable names to keep the code compact

1550. In a requirements document, what is a data dictionary? Understanding

A. A list defining each data item: its name, meaning, type, format and allowed values
B. A glossary of the programming language's keywords
C. A list of all the database queries the system runs
D. A translation of the requirements into another language

1551. Which diagram is used to document a database design, showing entities, their attributes and the relationships between them? Understanding

A. Gantt chart
B. Flowchart
C. Sequence diagram
D. Entity-relationship (ER) diagram

1552. What does a UML state machine diagram document? Understanding

A. The classes and their relationships
B. The states an object can be in and the events that move it from one state to another
C. The parallel steps in a business process across departments
D. The hardware nodes and network links

1553. Sommerville distinguishes user requirements from system requirements. Which statement is correct? Understanding

A. User requirements are high-level natural-language statements for customers; system requirements are detailed descriptions for developers
B. User requirements are written by developers; system requirements are written by end users
C. User requirements are non-functional; system requirements are functional
D. User requirements describe hardware; system requirements describe software

1554. When asking for help on a forum or filing a bug, what is a 'minimal reproducible example'? Understanding

A. The whole project zipped up so others can run it
B. The smallest complete piece of code and input that still shows the problem
C. A screenshot of the error message
D. A description of the problem without any code

1555. In requirements engineering, what is requirements validation? Understanding

A. Checking that the code passes its unit tests
B. Collecting requirements from stakeholders through interviews
C. Converting requirements into a design
D. Checking that the documented requirements describe the system the customer actually needs

1556. A library is at version 4.1.9. You release one backward-compatible bug fix. What is the new version under Semantic Versioning? Application

A. 4.1.10
B. 4.2.0
C. 4.2.9
D. 5.0.0

1557. A package has been at 0.9.3 during development. The team now declares its public API stable for production use. What should the next version be? Application

A. 0.10.0
B. 0.9.4
C. 1.0.0
D. 1.9.3

1558. Which of these is the highest version under Semantic Versioning? Application

A. 1.9.9
B. 1.10.0-rc.1
C. 1.2.10
D. 1.10.0

1559. Which is the best testable availability requirement for an exam portal? Application

A. The portal shall be highly available
B. The portal shall almost never go down during exams
C. The portal shall be available 99.9% of each calendar month, excluding maintenance announced 48 hours in advance
D. The portal shall be as available as possible

1560. A librarian wants to see which books are overdue so she can send reminders. Which is the best user story? Application

A. As a librarian, I want to see a list of overdue books, so that I can send reminders to borrowers
B. The system shall have an overdue books table in the database
C. As a system, I want to store due dates, so that the database is complete
D. Overdue books should be shown somehow

1561. Rule: 'Orders of 5,000 rupees or more get free delivery.' Which Given/When/Then scenario tests the boundary of this rule? Application

A. Given a cart total of exactly 5,000 rupees, When the customer checks out, Then delivery is free
B. Given a cart total of 10,000 rupees, When the customer checks out, Then delivery is free
C. Given a cart, When the customer checks out, Then delivery works correctly
D. Given a cart total of 2,000 rupees, When the customer checks out, Then the page loads

1562. Which docstring describes this function accurately and completely? Application

```
def clamp(x, lo, hi):
    return max(lo, min(x, hi))
```

A. Return the larger of x and hi.
B. Return x limited to the range [lo, hi]. Assumes lo <= hi.
C. Return the average of x, lo and hi.
D. Return x if it is between lo and hi, otherwise raise ValueError.

1563. Which call returns a result that contradicts the docstring? Application

```
def last_word(s):
    """Return the last word of s."""
    return s.split(" ")[-1]
```

A. last_word("hello world ")
B. last_word("hello world")
C. last_word("one")
D. last_word("a b c")

1564. Which comment adds the most value above this line: time.sleep(2 ** attempt)? Application

A. # Sleep for 2 to the power of attempt seconds
B. # Call the sleep function
C. # Exponential backoff: the API rate-limits clients that retry immediately
D. # Wait

1565. You changed the login page so the password field hides its text. Which commit subject line is best? Application

A. changes
B. Mask password input on the login page
C. Fixed stuff in login
D. Updated login.html, login.css and auth.js files

1566. Just before an investor demo, the company name on the home page is misspelt. Nothing else is affected. How should the bug be classified? Application

A. Low severity, high priority
B. High severity, high priority
C. Low severity, low priority
D. High severity, low priority

1567. Which time complexity should the docstring of this function state? Application

```
def halvings(n):
    count = 0
    while n > 1:
        n //= 2
        count += 1
    return count
```

A. O(n)
B. O(n log n)
C. O(log n)
D. O(1)

1568. The doctest example's expected output is missing (shown as ???). What should be written there so the doctest passes? Application

```
def initials(name):
    """Return the capitalised first letter of each word.

    >>> initials("ada  lovelace")
    ???
    """
    return "".join(w[0].upper() for w in name.split())
```

A. 'A L'
B. 'AL'
C. 'Al'
D. 'AlL'

1569. What should replace ??? so this doctest passes on standard CPython? Application

```
def to_2dp(x):
    """Round x to 2 decimal places.

    >>> to_2dp(2.675)
    ???
    """
    return round(x, 2)
```

A. 2.68
B. 2.67
C. 2.7
D. 2.675

1570. The results portal was down from 10:00 to 10:40 today. Which update to the head of department is best? Application

A. There was an issue with the system, but it is fine now.
B. Results were unavailable 10:00-10:40 after a database server ran out of disk space. It is fixed; we have added a disk-space alert so it does not recur.
C. The PostgreSQL WAL volume hit 100% on db-02, causing checkpoint failures and connection resets.
D. Some students may have had problems, we are not sure why.

1571. Developers merged 'Add server-side pagination to /api/results'. Which release note suits students who use the portal? Application

A. Implemented server-side pagination for the /api/results endpoint
B. Results pages now load faster, even if you have many entries
C. Refactored the results controller
D. Pagination added (see PR #482)

1572. Which rewrite makes 'The system shall respond quickly to large uploads' testable? Application

A. The system shall respond very quickly to all uploads
B. The system shall use an efficient upload algorithm
C. Large uploads shall not make users wait too long
D. For files up to 50 MB, the system shall confirm receipt within 5 seconds of the upload finishing, on a 10 Mbps connection

1573. A traceability matrix is shown below. Which statement is correct? Application

```
Requirement | Test cases
R1 Login       | T1, T2
R2 Reset pwd   | T3
R3 Export CSV  | (none)
R4 Audit log   | T5
Unlinked tests : T4
```

A. Every requirement is covered, so testing is complete
B. T4 should be deleted, but R3 needs no action
C. R3 is covered by T4
D. R3 has no test, and T4 traces to no requirement; both gaps need fixing

1574. You need to document how an online order moves through Placed, Paid, Shipped, Delivered and Cancelled. Which UML diagram fits best? Application

A. Class diagram
B. Use case diagram
C. Deployment diagram
D. State machine diagram

1575. A library at 2.3.4 removes a public function that some users call. Under Semantic Versioning, what is the next version? Application

A. 2.4.0
B. 2.3.5
C. 3.3.4
D. 3.0.0

1576. Which docstring states this function's contract correctly? Application

```
def withdraw(balance, amount):
    if amount <= 0 or amount > balance:
        raise ValueError("invalid amount")
    return balance - amount
```

A. Requires amount >= 0; returns balance - amount, which may be negative
B. Requires amount < balance; returns balance - amount; otherwise returns balance unchanged
C. Requires 0 < amount <= balance; returns balance - amount; otherwise raises ValueError
D. Returns balance - amount; never raises an error

1577. The comment states the business rule. Which input shows that the code and the comment disagree? Analysis

```
def final_price(total):
    # 10% off orders over 1000 rupees
    if total >= 1000:
        total = total * 0.9
    return total
```

A. final_price(1000)
B. final_price(1500)
C. final_price(999)
D. final_price(500)

1578. Which call shows that this function does not do what its docstring says? Analysis

```
def top_scores(scores):
    """Return the scores sorted from highest to lowest."""
    return sorted(scores)
```

A. top_scores([5])
B. top_scores([1, 2])
C. top_scores([])
D. top_scores([3, 3])

1579. Using the standard definition of the median, which input shows that this function is wrong? Analysis

```
def median(nums):
    """Return the median of nums (nums is non-empty)."""
    a = sorted(nums)
    return a[len(a) // 2]
```

A. median([3, 1, 2])
B. median([7])
C. median([5, 5, 5])
D. median([1, 2, 3, 4])

1580. A data breach exposed students' email addresses. Which message to students communicates it best? Analysis

A. Explain what was exposed (email addresses only, no passwords), when, what has been fixed, and what students should do, such as watching for phishing emails
B. Send the full technical incident report, including server logs
C. Say only that 'a minor technical issue occurred'
D. Say nothing until the investigation is completely finished

1581. Which of these requirements is still NOT verifiable? Analysis

A. The system shall lock an account after 5 consecutive failed logins
B. The system shall export reports as PDF and CSV
C. The system shall handle errors gracefully
D. The system shall send a confirmation email within 2 minutes of registration

1582. add_item(1) is called, then add_item(2). Why does the second call show that the docstring is wrong? Analysis

```
def add_item(x, items=[]):
    """Return a new list that contains x appended to items."""
    items.append(x)
    return items
```

A. It returns [2], so the docstring is correct
B. It raises an error because items is empty
C. It returns [1, 2] because Python joins the arguments of all calls
D. It returns [1, 2], because the default list is created once and shared between calls, so no new list is made

1583. A library at 2.3.4 releases 2.3.5. Its only change: parse() now raises an error on empty input instead of returning None. Many users' programs crash. What went wrong? Analysis

    A. Nothing; bug fixes always bump PATCH
    B. The change breaks existing callers, so it should have been released as 3.0.0 with a migration note
    C. It should have been 2.4.0, because it adds behaviour
    D. Users should have pinned the exact version, so the release was correct

1584. A developer's single commit, 'misc fixes', changes the payment logic, renames CSS classes and updates a dependency. What is the best review advice? Analysis

    A. Keep it, but make the message longer
    B. Split it into separate commits, one per change, each with a clear message
    C. Squash it with earlier commits to keep the history short
    D. Merge it now and document the changes later

1585. For three weeks a developer's status report has said 'Feature is 90% done'. What is the main communication problem? Analysis

    A. 90% is too high a number to report
    B. Status reports should be sent daily
    C. The percentage is not tied to anything checkable; reporting the tasks finished, the tasks left and an expected date would be clearer
    D. The report should use more technical detail

1586. Which input exposes the gap between this function and its docstring? Analysis

```
def is_palindrome(s):
    """Return True if s reads the same backwards, ignoring case."""
    return s == s[::-1]
```

    A. is_palindrome("Aba")
    B. is_palindrome("aba")
    C. is_palindrome("")
    D. is_palindrome("abc")

1587. A function was refactored to return None on failure instead of raising an exception, but its docstring still says 'Raises KeyError if the ID is not found'. What is the main risk? Analysis

    A. Callers will wrap it in try/except and miss failures, because the outdated doc tells them to expect an exception
    B. Python will refuse to run the function until the docstring is fixed
    C. The function will become slower
    D. There is no risk, because docstrings do not affect behaviour

1588. A team must explain to new developers how the Student, Course and Enrolment tables relate. Which artefact communicates this best? Analysis

    A. A Gantt chart
    B. An ER diagram
    C. A sequence diagram
    D. A list of commit messages

1589. A team writes the acceptance criterion: 'Given a user, When they log in, Then it works.' What is the main weakness? Analysis

A. It uses the Given/When/Then format, which is for code, not requirements
B. It is too long
C. It should start with 'When' instead of 'Given'
D. Nothing in it can be checked: the starting state, the action and the expected outcome are all vague

1590. A developer replaced a list with a hash set for checking student IDs. Which explanation best suits a non-technical manager? Analysis

A. We changed from O(n) list membership to an O(1) average-case hash set lookup
B. We used a better data structure
C. We refactored the ID-checking module
D. Checking whether a student is registered now takes about the same time however many students there are, so the portal stays fast during admissions

1591. Which call shows that this function does not follow its docstring? Analysis

```
def page_items(items, page, size):
    """Return the items on the given page. Pages are numbered from 1."""
    return items[page * size:(page + 1) * size]
```

A. page_items(list(range(10)), 0, 5)
B. page_items([], 1, 5)
C. page_items(list(range(3)), 2, 5)
D. page_items(list(range(10)), 1, 5)

Number Series Questions

1592. A series has constant second differences equal to 6. What is the coefficient of n^2 in its nth-term formula $an^2 + bn + c$? Understanding

A. 3
B. 6
C. 12
D. 2

1593. What do the terms of 11, 13, 17, 19, 23, 29 have in common, and what comes next? Understanding

A. They are odd numbers; next is 31
B. They increase by 2 then 4; next is 33
C. They are consecutive primes; next is 37
D. They are consecutive primes; next is 31

1594. In the interleaved series 2, 30, 4, 27, 6, 24, which terms form the second sub-series? Understanding

A. 2, 4, 6 (the 1st, 3rd and 5th terms)
B. 30, 27, 24 (the 2nd, 4th and 6th terms)
C. 2, 30, 4 (the first three terms)
D. 27, 6, 24 (the last three terms)

1595. A series is generated by $a_n = 2n^3 - n$. What constant value do its third differences take? Understanding

A. 12
B. 6
C. 2
D. 24

1596. Letters are numbered A = 1 to Z = 26 and counting wraps from Z back to A. Which letter is 5 places after V? Understanding

A. Z
B. B
C. U
D. A

1597. A series has first differences 3, 6, 12, 24. What does this tell you about the next term? Understanding

A. The differences rise by 3, so the next difference is 27
B. The differences double, so the next difference is 48
C. The second differences are constant, so the series is quadratic
D. The series itself is geometric with ratio 2

1598. What is the next term: 8, 10, 15, 25, 42, ? Application

A. 59
B. 66
C. 60
D. 68

1599. What is the next term: 3, 10, 29, 66, 127, ? Application

A. 216
B. 218
C. 188
D. 225

1600. What is the next term: 4, 6, 12, 14, 28, 30, ? Application

A. 60
B. 32
C. 56
D. 62

1601. What is the next term: 1, 50, 2, 45, 4, 40, 8, ? Application

A. 16
B. 35
C. 36
D. 30

1602. Using A = 1 to Z = 26 and wrapping from Z to A, what is the next letter: C, F, L, U, ? Application

A. F
B. H
C. G
D. Z

1603. What is the next term: 2, 6, 15, 31, 56, ? Application

A. 81
B. 87
C. 91
D. 92

1604. What is the next term: 1, 3, 4, 7, 11, 18, ? Application

A. 29
B. 25
C. 26
D. 36

1605. What is the 50th term of 5, 14, 27, 44, 65, ...? Application

A. 5150
B. 5050
C. 5003
D. 10150

1606. One term in this series is wrong: 3, 5, 9, 17, 33, 66, 129. Which is it? Analysis

A. 33
B. 129
C. 66
D. 17

1607. Find the missing term: 4, 9, ?, 69, 204 Analysis

A. 19
B. 39
C. 24
D. 23

1608. Which formula generates 3, 8, 15, 24, 35 for n = 1, 2, 3, 4, 5? Analysis

A. $5n - 2$
B. $n^2 + 2$
C. $n(n + 2)$
D. $2n^2 + 1$

1609. A student says the next term of 1, 4, 10, 20, 35 is 50 because 'the differences go up by 3'. What is the correct next term? Analysis

A. 50
B. 55
C. 52
D. 56

1610. Which of these series has second differences that are constant and non-zero? Analysis

A. 2, 4, 8, 16, 32
B. 5, 8, 13, 20, 29
C. 1, 8, 27, 64, 125
D. 3, 6, 9, 12, 15

1611. One term breaks the pattern: 2, 15, 4, 12, 8, 9, 16, 7, 32. Which is it? Analysis

A. 9
B. 16
C. 7
D. 32

Logical Reasoning Puzzles

1612. Eight people sit evenly around a round table. Counting one way round, how many people sit between X and the person directly opposite X? Understanding

A. 4
B. 2
C. 3
D. 6

1613. People sit around a round table facing the centre. Viewed from above, which neighbour is a person's immediate left? Understanding

A. The next seat anticlockwise
B. The person directly opposite
C. It depends on who sits first
D. The next seat clockwise

1614. Given only 'Some engineers are poets', which statement must also be true? Understanding

A. Some poets are engineers
B. All engineers are poets
C. Some poets are not engineers
D. All poets are engineers

1615. Assume every group mentioned has at least one member. From 'All managers are graduates', which statement does NOT follow? Understanding

A. Some graduates are managers
B. Some managers are graduates
C. No manager is a non-graduate
D. All graduates are managers

1616. On an island, knights always tell the truth and knaves always lie. Who could say 'I am a knave'? Understanding

A. Only a knight
B. Only a knave
C. Nobody
D. Either a knight or a knave

1617. In how many distinct ways can 5 people sit around a round table, if rotations of the same order count as one arrangement? Understanding

A. 24
B. 120
C. 60
D. 12

1618. Sara is 9th from the top and 14th from the bottom in her class ranking. How many students are in the class? Application

A. 23
B. 22
C. 21
D. 24

1619. K, L, M, N and O sit in a row of five seats, all facing north. M is in the middle seat. K sits immediately to M's left. O is at one end, and L sits next to O. Who is at the left end? Application

A. O
B. N
C. L
D. K

1620. G, H, I, J, K and L sit around a round table facing the centre. H is immediately to G's left. J is opposite G. K is immediately to J's left. I is not immediately to H's left. Who sits opposite I? Application

A. H
B. K
C. G
D. L

1621. R is taller than S. T is taller than R. U is shorter than S. V is shorter than T but taller than R. Who is the third tallest? Application

A. R
B. V
C. S
D. T

1622. Assume every group mentioned has at least one member. 'No pens are pencils. All pencils are erasers.' Which conclusion definitely follows? Application

A. No pens are erasers
B. Some pens are erasers
C. All erasers are pencils
D. Some erasers are not pens

1623. Knights always tell the truth and knaves always lie. A says: 'Exactly one of us three is a knight.' B says: 'Exactly two of us are knights.' C says: 'None of us is a knight.' What are they? Application

A. B knight; A and C knaves
B. A and B knights; C knave
C. A knight; B and C knaves
D. All three knaves

1624. One of Ali, Bilal and Chand broke a window. Ali: 'I did not do it.' Bilal: 'Ali did it.' Chand: 'Bilal did it.' Exactly one of them is lying. Who broke the window? Application

A. Ali
B. Bilal
C. Chand
D. It cannot be decided

1625. In a row of 40 students, A is 12th from the left and B is 20th from the right. How many students sit between A and B? Application

A. 8
B. 9
C. 7
D. 10

1626. A, B, C, D and E sit in a row of five seats facing north. C is in the middle, A sits next to C, and E is at one end. Which one extra clue fixes the whole arrangement? Analysis

A. D sits at the right end
B. A sits to the left of C
C. B sits immediately to the right of E
D. B is not at an end

1627. 'Some phones are tablets. No tablets are laptops.' Which conclusion definitely follows? Analysis

A. No phones are laptops
B. Some phones are not laptops
C. Some phones are laptops
D. Some laptops are not phones

1628. One of W, X, Y and Z took the last samosa. W: 'I did not.' X: 'Y did.' Y: 'Z did.' Z: 'I did not.' Exactly one of them tells the truth. Who took it? Analysis

A. X
B. Y
C. Z
D. W

1629. 'All analysts are bloggers. Some bloggers are coders.' A student concludes 'Some analysts are coders'. Which assessment is correct? Analysis

A. It does not follow: the bloggers who code might include no analysts
B. It follows, because analysts are bloggers and some bloggers code
C. It follows only if every group has at least one member
D. The opposite follows: no analysts are coders

1630. P, Q, R, S, T and U sit around a round table facing the centre. Q is opposite P. R sits next to P. S is immediately to Q's left. How many different arrangements are possible? Analysis

A. 2
B. 6
C. 4
D. 8

1631. Among four friends: R is taller than S, T is taller than R, and U is shorter than T. Which statement must be true? Analysis

A. R is the second tallest
B. T is the tallest
C. U is the shortest
D. S is shorter than U

Data Interpretation Questions

1632. A department's pass rate rose from 32% to 40%. Which description is correct? Understanding

A. A rise of 8%, which is also 8 percentage points
B. A rise of 8 percentage points, which is a 25% relative increase
C. A rise of 25 percentage points
D. A rise of 8 percentage points, which is a 20% relative increase

1633. A table shows that a value grew by 100% over a year. What happened to the value? Understanding

A. It stayed the same
B. It increased by 100 units
C. It doubled
D. It tripled

1634. An index uses 2020 = 100 as its base year. In 2024 the index is 80. What does this mean? Understanding

A. The 2024 value is 20% below the 2020 value
B. The 2024 value is 80% below the 2020 value
C. The value fell by 20 units since 2020
D. The value fell by 20% every year

1635. A price rises by 25% and then falls by 20%. What is the overall change? Understanding

A. A 5% increase
B. A 5% decrease
C. A 2.5% increase
D. No change

1636. Three sections of different sizes report their own pass rates. How should you find the overall pass rate? Understanding

A. Take the simple mean of the three rates
B. Weight each rate by its section size (total passes ÷ total students)
C. Take the median of the three rates
D. Use the highest rate, since the largest section dominates

1637. In a pie chart, one category is one-eighth of the total. What is its central angle? Understanding

A. 12.5°
B. 40°
C. 60°
D. 45°

1638. App downloads (thousands) by quarter are shown. By what percentage did Beta's downloads grow from Q1 to Q4? Application

app	Q1	Q2	Q3	Q4
Alpha	240	300	270	360
Beta	160	176	220	242
Gamma	90	135	162	162

A. 82%
B. 51.25%
C. 33.88%
D. 45%

1639. Using the same table, what were the total downloads (thousands) across all three apps in Q3? Application

app	Q1	Q2	Q3	Q4
Alpha	240	300	270	360
Beta	160	176	220	242
Gamma	90	135	162	162

A. 611
B. 764
C. 652
D. 627

1640. Using the same table, which app had the largest percentage increase from Q1 to Q4? Application

app	Q1	Q2	Q3	Q4
Alpha	240	300	270	360
Beta	160	176	220	242
Gamma	90	135	162	162

A. Gamma
B. Alpha
C. Beta
D. Alpha and Beta equally

1641. Using the same table, what share of all Q4 downloads came from Alpha, to one decimal place? Application

app	Q1	Q2	Q3	Q4
Alpha	240	300	270	360
Beta	160	176	220	242
Gamma	90	135	162	162

A. 36.0%
B. 50.0%
C. 47.1%
D. 41.4%

1642. A pie chart shows the preferred languages of 720 surveyed students. How many students prefer Java? Application

```
language | angle
Python   | 140
Java     |  90
C++      |  70
JS       |  40
Other    |  20
```

A. 90
B. 200
C. 180
D. 160

1643. Section A has 45 students with an average of 64 marks. Section B has 35 students with an average of 72. What is the combined average? Application

A. 68
B. 67.5
C. 66.8
D. 69.2

1644. A startup's monthly users grew from 50,000 to 72,000 over two years. What is the compound annual growth rate (CAGR)? Application

A. 20%
B. 22%
C. 44%
D. 21%

1645. After a 15% fall, a laptop's resale value is 340 (thousand rupees). What was the value before the fall? Application

A. 400
B. 391
C. 355
D. 425

1646. A price index (base 2020 = 100) is 120 in 2023 and 135 in 2024. A report says 'prices rose 15% in 2024'. What is the actual rise? Analysis

A. 15%
B. 35%
C. 11.1%
D. 12.5%

1647. A report gives only the pass rates for two departments: CS 60% and IT 80%. What is the overall pass rate for the two departments together? Analysis

A. 70%
B. Between 60% and 80%, and exactly 70% if the departments have similar sizes
C. 80%, because the higher rate dominates
D. It cannot be determined without the number of students in each department

1648. Using the app table, which statement is supported by the data? Analysis

```
app   | Q1  | Q2  | Q3  | Q4
Alpha | 240 | 300 | 270 | 360
Beta  | 160 | 176 | 220 | 242
Gamma |  90 | 135 | 162 | 162
```

A. Beta's share of total downloads rose every quarter
B. Gamma's downloads were unchanged from Q3 to Q4, but its share of total downloads fell
C. Alpha grew in every quarter
D. Total downloads fell from Q2 to Q3

1649. Using the app table, which single quarter-on-quarter change was the largest in percentage terms? Analysis

```
app   | Q1  | Q2  | Q3  | Q4
Alpha | 240 | 300 | 270 | 360
Beta  | 160 | 176 | 220 | 242
Gamma |  90 | 135 | 162 | 162
```

A. Gamma, Q1 to Q2
B. Alpha, Q3 to Q4
C. Beta, Q2 to Q3
D. Alpha, Q1 to Q2

1650. Branch X's sales rose from 200 to 260 units; Branch Y's rose from 1,000 to 1,200 units. A manager says 'X grew faster'. Which assessment is correct? Analysis

A. False, because Y added more units
B. True on every measure
C. True in percentage terms (30% vs 20%), but Y added more units (200 vs 60)
D. False, because both grew by the same percentage

1651. Monthly site visitors (thousands) were Jan 50, Feb 70, Mar 63, Apr 84, May 110. Which month had the largest percentage increase over the previous month? Analysis

A. May
B. April
C. March
D. February

Answer key and explanations

1. **B** — Pólya's method runs: understand the problem, devise a plan, carry out the plan, then look back. You cannot plan well until you know what is given, what is unknown and what conditions apply, so understanding always comes first.
2. **A** — IPO stands for Input, Processing, Output. Processing is the set of operations that turn the given inputs into the required outputs. It describes what must happen, not the code or the notation used to write it.
3. **B** — Finiteness means the algorithm always terminates after a finite number of steps. Definiteness is about each step being unambiguous, effectiveness about each step being basic enough to perform, and input about the values it receives.
4. **C** — Definiteness requires every step to be precise and unambiguous. "A little more" and "tastes right" mean different things to different people, so two people following the same step could do different things.
5. **B** — A diamond is the decision symbol: one condition goes in, and two or more labelled branches (such as Yes and No) come out. Parallelograms are for input/output, rectangles for processing and ovals for start/end.
6. **B** — The parallelogram marks data entering or leaving the algorithm: reading input or displaying output. Calculations use a rectangle, decisions a diamond and the start/end points an oval.
7. **A** — An algorithm is the logical method, which can be written in English, pseudocode or a flowchart. A program is one concrete implementation of it in a language such as Python or C++. One algorithm can be implemented as many different programs.
8. **A** — A well-defined problem states where you start, what counts as reaching the goal and which moves or operations are allowed. Ill-defined problems leave at least one of these vague, so the first job is to pin them down.
9. **C** — Working backwards starts from the known goal state and reverses each step until it reaches the start. It suits problems where the end result is given and the operations can be inverted.
10. **A** — Looking back happens after the plan is carried out. You verify that the answer satisfies every condition, check for errors and ask whether the method generalises. Re-reading the question before starting belongs to the first step.
11. **A** — A dry run is a manual trace, usually recorded in a trace table with one column per variable. It shows how values change at each step and catches logic errors before any code is executed.
12. **D** — Top-down design, also called stepwise refinement, begins with a one-line statement of the whole task and repeatedly breaks each step into more detailed steps. It is the usual way to turn a problem definition into an algorithm.
13. **D** — Pseudocode mixes plain language with programming-style structure (IF, WHILE, FOR) to describe an algorithm. It has no fixed syntax and is never compiled; its purpose is to make the logic clear before coding.
14. **C** — A problem definition says what must be solved, not how it will be coded. Output format, input ranges and resource limits all change what counts as a correct solution. Variable names are an implementation choice made later.
15. **D** — Knuth (The Art of Computer Programming, Vol. 1) lists five properties: finiteness, definiteness, input, output and effectiveness. An algorithm may take zero inputs (for example, one that prints the first 10 primes) but must produce at least one output, otherwise it has no observable result.
16. **A** — Effectiveness, in Knuth's terms, means each individual operation is basic and can be carried out exactly in finite time. Definiteness is about steps being unambiguous; finiteness is about the whole algorithm stopping.
17. **D** — The loop runs for i = 1, 2, 3, 4, 5 and adds the squares: 1 + 4 + 9 + 16 + 25 = 55. range(1, 6) stops before 6, so 5 is included. 15 is the plain sum, 30 stops at 4 and 25 is only the last square.
18. **D** — Outputs are what the program must produce for the user: here the average and the grade. The three marks are inputs, and the number 3 is a fixed value used during processing.
19. **D** — Work backwards and undo each step in reverse order: 9 × 3 = 27, then 27 + 7 = 34, then 34 ÷ 2 = 17. Check forwards: 17 × 2 = 34, 34 − 7 = 27, 27 ÷ 3 = 9. Getting 10 comes from subtracting 7 instead of adding it back.
20. **B** — Trace x: 10 → 5 → 16 → 8 → 4 → 2 → 1. That is 6 changes, so steps ends at 6. Counting the starting value 10 as a step gives 7, a common off-by-one slip.

21. **C** — After a = a + b, a is 8. Then b = 8 − 5 = 3 and a = 8 − 3 = 5. The three lines swap the values without a temporary variable, so the output is 5 3.
22. **A** — The first number becomes the running maximum without any comparison. Each of the other 9 numbers is compared with it once, so the count is n − 1 = 9. 45 would be comparing every pair.
23. **C** — Boundary testing puts values on and just outside each limit: 17 (reject), 18 (accept), 60 (accept), 61 (reject). Errors such as writing > instead of >= show up exactly at these points.
24. **D** — The inner loop runs from i to 4, so it runs 4, 3, 2 and 1 times for i = 1, 2, 3, 4. Total = 4 + 3 + 2 + 1 = 10. 16 assumes the inner loop always runs 4 times; 6 forgets that j can equal i.
25. **C** — n % 10 takes the last digit and n // 10 removes it. The loop adds 4, 3, 2 and 1, so s = 10. 4 would be the number of digits and 4321 the reversed number.
26. **C** — i goes 1, 3, 5, 7, 9, 11, 13 and so on. It is always odd, so i ≠ 10 is always true and the loop never ends. This breaks finiteness. Using i < 10 as the condition would fix it.
27. **B** — At exactly 50 the requirement says Pass, but marks > 50 is false, so the code prints Fail. For 49, 51 and 100 both versions agree, so those tests would pass and hide the bug.
28. **A** — You first understand what is given and what is asked (U), then devise a plan (P), carry it out (C) and finally look back to check and reflect (L).
29. **D** — Halving the range each time is divide and conquer. Six guesses can tell apart at most 2^6 − 1 = 63 numbers, and seven can handle 2^7 − 1 = 127, so 100 numbers need at most 7 guesses ($\lceil \log_2 101 \rceil = 7$).
30. **B** — a = b sets a to 5, and the old value 3 is lost. b = a then copies 5 back into b. Both are 5. A correct swap needs a temporary variable or Python's a, b = b, a.
31. **D** — When b > a, the elif branch is skipped, so c is never compared. For (7, 12, 15) m becomes 12 and 15 is ignored. The fix is two separate if statements. The other calls happen to work because the maximum is either a, b, or a c that the elif still reaches.
32. **C** — The correct sum is 18. 18 ÷ 2 = 9, which matches dividing by n − 1 = 2. Adding the first element twice gives 22 ÷ 3 ≈ 7.33, integer division gives 6, and skipping the last element gives 10 ÷ 3 ≈ 3.33.
33. **B** — The task is ill-defined until you know the sort key (name, roll number, CGPA) and the direction. Without that, any output could be called correct or wrong. Language and loop choice are implementation details that do not change what a correct result is.
34. **A** — "Education" has E, u, a, i and o. The test ch in "aeiou" is case-sensitive, so the uppercase E fails. Converting with s.lower() or adding uppercase vowels fixes it.
35. **D** — range(1, n) stops at n − 1, so n itself is never added: an off-by-one error. range(1, n + 1) includes n. Starting total at 10 only works when n = 10, starting the range at 0 adds nothing, and adding 1 is simply wrong.
36. **B** — $\lfloor \sqrt{97} \rfloor = 9$, so B tries 2, 3, …, 9, which is 8 divisors. A would try 2 to 96, which is 95. The look-back step asks exactly this: is there a cheaper plan with the same answer? Any factor above √n pairs with one below it, so B is enough.
37. **C** — With n > 0 and n decreasing by 2, n eventually reaches 0 or −1 and the condition fails, so the loop always ends. The ≠ 0 version skips 0 for odd n, the ≠ 10 version runs forever for n > 10 and the last one only grows.
38. **C** — Testing can show the presence of bugs, not their absence. All three tests are even, so a function that always returns True would pass them. Good test sets cover each class of input: odd, even, zero and negative.
39. **A** — Work backwards from the goal: if coverage doubles each day, the day before full coverage it must have been half. So it was half covered on day 29. Day 15 assumes linear growth, which the doubling rule rules out.
40. **B** — The median needs sorted data and a rule for even lengths. Taking the middle of an unsorted list is wrong in general, the mean is a different statistic, and always taking index length ÷ 2 gives the upper middle value for even lengths instead of the average of the two.
41. **B** — An oval, also called a terminator, marks Start and End. A diamond is a decision, a parallelogram is input or output, and a rectangle is a processing step.
42. **D** — A rectangle is the process box, used for assignments and calculations. A diamond asks a yes/no question, an oval marks start or end, and a parallelogram reads or displays data.
43. **A** — A trace table has one column per variable, plus often one for output. Each row shows the values after a step, so you can follow the algorithm by hand without running it.

44. **C** — The Böhm–Jacopini result shows that sequence (one step after another), selection (if/else) and iteration (loops) can express any algorithm. Input, processing and output is the IPO model, which describes a problem, not control flow.
45. **B** — Only the radius comes from the user, so it is the only input. 3.14159 is a fixed constant used inside the processing step, not an input. The area is what the user needs, so it is the output.
46. **C** — An algorithm is a finite set of precise, unambiguous steps that solves a problem. It can be written in pseudocode, as a flowchart or as code; a program is just one way to express it. Vague steps break definiteness.
47. **C** — A logic error lets the program run but produces wrong results, for example using > instead of >= or adding the wrong variable. A syntax or compilation error would stop the program from running at all.
48. **A** — Brute force tries every possible candidate in turn. It is simple and guaranteed to succeed for a finite search space, but slow when there are many candidates, here up to 1,000.
49. **A** — Testing runs the program on chosen inputs and compares the results with what is expected, revealing that something is wrong. Debugging is the follow-up work of locating the cause and correcting it.
50. **A** — Iteration (a loop such as WHILE or FOR) repeats steps while a condition is true. Selection chooses between paths once, sequence runs steps one after another, and decomposition is a design technique, not a control structure.
51. **D** — An algorithm guarantees a correct result for every valid input. A heuristic, such as 'try a simpler case first' or 'pick the nearest city next', is a strategy that usually helps but carries no such guarantee.
52. **B** — Bottom-up design starts from basic building blocks, such as a date-validation function, and assembles them into larger parts. Top-down design does the reverse: it starts with the whole problem and refines it into sub-tasks.
53. **C** — Divide and conquer breaks a problem into smaller copies of itself, as merge sort sorts two halves and then merges them. Brute force tries every candidate, and working backwards starts from the goal.
54. **B** — Guess and check improves each guess using feedback from the last one. Brute force differs because it tries every candidate in a fixed order without using feedback to skip any.
55. **B** — Pólya lists this question under devising a plan: a related solved problem can suggest a method. Understanding the problem asks about the unknown, the data and the condition; looking back checks the result.
56. **A** — A small labelled circle is an on-page connector. It joins two parts of a flowchart without drawing a long arrow; the flow jumps to the matching label. Start and end use an oval.
57. **D** — Pseudocode is structured plain text, so long algorithms are easy to write, revise and turn into code. It has no fixed syntax, so it cannot be executed or compiled, and it is flowcharts that use standard symbols.
58. **D** — A well-defined problem has an initial state, allowed operations and a goal state. The goal state is the one that meets the requirements, for example all disks on the target peg in the Tower of Hanoi.
59. **D** — "Better" names no measurable goal, so there is no way to tell when the problem is solved. The other three state the input, the required output and a clear test for a correct answer.
60. **C** — A greedy method makes the locally best choice at each step, such as taking the largest coin that fits. It is fast but does not always give the overall best answer. Trying every choice is brute force.
61. **C** — The base case, such as factorial(0) = 1, is answered without another recursive call. Without it, the function would keep calling itself and never finish, breaking finiteness.
62. **D** — If 0 were the sentinel and a student genuinely scored 0, input would stop early. A value such as −1 cannot be a real mark, so it signals the end without hiding data. The sentinel is never processed as data.
63. **C** — A loop invariant holds before the loop starts and after every pass. When the loop ends with k = n, it gives the required result, which is how a dry run is turned into an argument that the loop is correct.
64. **A** — √2 has infinitely many non-repeating digits, so no one can carry out the step exactly in finite time. That is effectiveness. Definiteness is satisfied, because the instruction is unambiguous.
65. **A** — x changes 5 → 9 → 17 → 33 over the three passes (each pass doubles x and subtracts 1). 9 and 17 are the values after one and two passes; 65 would need a fourth pass.
66. **A** — Processing transforms inputs into outputs: comparing the balance with the amount and subtracting it are processing steps. The typed amount is an input; the cash and receipt are outputs.
67. **C** — 72 fails the first test (72 >= 80 is false), then passes the second (72 >= 65 is true), so grade becomes B and the remaining tests are skipped. In an ELSE IF chain only the first true branch runs.

68. **C** — Work backwards: before lunch she had 300 + 200 = 500. That was half of her money, so she started with 500 × 2 = 1,000. Doubling 300 first and then adding 200 gives 800, which undoes the steps in the wrong order.
69. **C** — range(2, 20, 3) gives 2, 5, 8, 11, 14, 17. The next value, 20, is not included, so the body runs 6 times. 18 is the size of the gap 2–20, not a count of steps.
70. **C** — Checking the result against the original problem is the look-back step. Solving for x was carrying out the plan; choosing to isolate x was devising it.
71. **A** — (a, b) goes (48, 18) → (18, 12) → (12, 6) → (6, 0). The loop runs 3 times and stops when b is 0, leaving a = 6, the GCD. Counting the starting pair as a pass gives the wrong count 4.
72. **A** — Every match removes one team, and 31 teams must be removed to leave a champion, so there are 31 matches (16 + 8 + 4 + 2 + 1). 16 counts only the first round and 5 is the number of rounds.
73. **A** — Work backwards by dividing by 3 four times: 162 → 54 → 18 → 6 → 2. Stopping after one, two or three divisions gives 54, 18 or 6.
74. **B** — idx tracks the position of the smallest value. It moves to 2 when nums[2] = 1 is found, and no later value is smaller. 1 is the smallest value itself, and 3 is its position counted from 1.
75. **B** — Each character is put in front of r: N, SN, CSN, TCSN. So the string is reversed. Writing r = r + ch would rebuild NSCT unchanged.
76. **C** — The number of pairs is 10 × 9 ÷ 2 = 45. 90 counts each pair twice (A with B and B with A), and 100 also compares each student with themselves.
77. **C** — The rule changes between 5,000 (no discount) and 5,001 (discount). Testing both catches the common slip of coding >= instead of >. 4,999 and 5,000 should both get no discount, so they never test where the discount starts.
78. **B** — The loop adds 4, 7 and 2, then stops on −1 without processing it, so total = 13 and count = 3. Adding −1 as data gives 12 4; skipping it and going on to 5 gives 18 4.
79. **D** — The pattern is n(n − 1)/2: each of 12 people shakes 11 hands, and every handshake is shared by two people, so 12 × 11 ÷ 2 = 66. 132 forgets to halve, and 144 includes people shaking their own hand.
80. **D** — Work backwards: 14:10 − 2 h 45 min = 11:25, the actual departure. It left 20 minutes late, so the scheduled time was 11:25 − 20 min = 11:05. 11:45 adds the delay instead of subtracting it.
81. **D** — f(7) = 7 + f(5) = 7 + 5 + f(3) = 7 + 5 + 3 + f(1), and f(1) = 1 is the base case. Total 16. 28 adds every number 1 to 7, and 15 forgets the base case value.
82. **D** — Work backwards: if x remain after a sale, there were 2x + 1 before it. 0 → 1 → 3 → 7. Check: 7 − 4 = 3, 3 − 2 = 1, 1 − 1 = 0. 15 would need a fourth customer.
83. **A** — The moves satisfy T(n) = 2T(n − 1) + 1 with T(1) = 1, giving 1, 3, 7, 15, 31, which is $2^n - 1$. 32 is 2^5 without subtracting 1.
84. **D** — The two assignments run one after another, so b = a + b uses the new a, which already equals b. b therefore doubles each pass: 2, 4, 8, 16, 32, 64. 13 is what a Fibonacci-style simultaneous update a, b = b, a + b would give.
85. **A** — Count by size: 16 of 1 × 1, 9 of 2 × 2, 4 of 3 × 3 and 1 of 4 × 4, so 16 + 9 + 4 + 1 = 30. 16 counts only the unit squares; 29 forgets the whole grid.
86. **A** — Both branches do the same thing, so the test changes nothing and students below 50 are wrongly told they passed. The symbols are fine, and >= 50 matches a normal pass rule.
87. **D** — Sequence matters: each step must come after the steps it depends on. Pouring depends on boiling, so steps 2 and 3 must swap. The algorithm still ends after four steps.
88. **B** — 15 passes n % 3 == 0 first and returns Fizz before the FizzBuzz test is reached. In a chain of conditions the most specific case must come first. Testing 15 first fixes every multiple of both.
89. **C** — count = 0 sits inside the loop, so it is wiped on every pass. The last element, −1, is negative, leaving count = 1. Moving count = 0 above the loop gives 3.
90. **D** — 4 + 1 + 1 is a correct way to pay 6, but 3 + 3 uses only two coins. Greedy is a heuristic: it is optimal for some coin sets but not for every set.
91. **B** — After sorting, equal values sit next to each other, so checking neighbours finds every duplicate. Without sorting, duplicates can be far apart.

92. **C** — Every original test had its largest value at the end, so 'return the last element' passed. [9, 2, 5] puts the maximum first; the buggy function returns 5 instead of 9.
93. **D** — m starts at 0, which is larger than every value in an all-negative list, so the function returns 0 instead of −2. Starting with m = nums[0] fixes it.
94. **B** — 1900 is divisible by 4, so the function says leap, but it is a century year not divisible by 400, so it is not a leap year. 2000, 2024 and 2023 all get the right answer by luck.
95. **B** — The two changes multiply: 1.2 × 0.8 = 0.96, so the original was 960 ÷ 0.96 = 1,000. Working backwards means undoing each step: 960 ÷ 0.8 = 1,200, then 1,200 ÷ 1.2 = 1,000.
96. **A** — Splitting an empty string on a space gives [''], one empty piece, so the function reports one word in a text with none. The definition must say what counts as a word, including empty text and repeated spaces.
97. **B** — Binary search discards half the list on the assumption that values are in order. Here mid = 2 holds 1 < 9, so the search moves right and never looks at index 1, where 9 sits. Sorting first, or using linear search, fixes it.
98. **B** — When i = 0 the code reads s[len(s) − 0] = s[5] in a 5-character string, which does not exist. The mirror of index i is len(s) − 1 − i; this off-by-one crashes before any comparison is made.
99. **B** — Removing every factor of 2 leaves 1 only when n had no other factor, so the plan is right for all n, including n = 1 = 2^0. The other plans pass some tests but fail others.
100. **D** — avg is recalculated on every pass, and the last pass uses the full total, so non-empty lists get the right result (wasteful but correct). For an empty list the loop never runs, so return avg fails with UnboundLocalError.
101. **B** — A constraint restricts which solutions are acceptable, for example "n ≤ 1000", "finish within 2 seconds" or "no two exams at the same time". The output is what is asked for, and given data are inputs; neither is a constraint by itself.
102. **B** — A functional requirement says what the system must do: a behaviour or service, such as downloading a result card. Load time, availability and screen support describe how well or under what conditions it works, so they are non-functional.
103. **D** — It sets a quality level for how the system behaves (speed), not a new behaviour, so it is a non-functional performance requirement in Sommerville's classification.
104. **A** — Decomposition splits a big problem into smaller, more manageable parts. Each part is solved on its own and the results are combined. Removing irrelevant data is abstraction, a related but different skill.
105. **B** — An assumption is something you take as true when the problem does not state it. Assumptions are allowed, but you must write them down, because a different assumption can change the answer.
106. **B** — Implicit constraints come from the real-world meaning of the data. A count of students can never be negative, even if the problem never says so. The other options are explicit constraints written in the problem.
107. **B** — Hard constraints must always be satisfied, or the solution is invalid. Soft constraints are preferences: you try to satisfy them, but a timetable that breaks one is still valid, just less good.
108. **B** — A good requirement is verifiable. "Fast" has no number attached, so developers and testers will disagree on whether it is satisfied. It should be rewritten as something like "95% of searches return within 2 seconds".
109. **A** — Pólya's first questions are: What is the unknown? What are the data? What is the condition? The unknown is what you must find, the data are what you are given, and the condition links them.
110. **A** — Edge cases sit at the boundaries of the input space: zero, empty, one element, the maximum allowed size, negative values. Many logic errors appear only there, so analysis should list them before coding.
111. **B** — A dependency means B consumes A's result, so A comes first in the solution order. Independent sub-problems can be solved in any order or in parallel; dependent ones cannot.
112. **D** — Abstraction removes irrelevant detail, such as the colour of a bus in a speed-distance question, so you can focus on the quantities and relationships that determine the answer. Splitting into parts is decomposition.
113. **D** — Performance, availability and portability are all quality attributes, so the first list is entirely non-functional. Each other list includes at least one behaviour the system must provide, such as registering or choosing a test city, which is functional.
114. **C** — A verifiable requirement gives a measurable target and the conditions under which it is measured, so a test can pass or fail. "User-friendly", "quickly" and "relevant" are subjective unless they are turned into numbers.
115. **C** — A precondition states what the algorithm may assume about its input; binary search, for example, requires a sorted list. A postcondition describes what must be true of the output when the algorithm finishes.

116. **D** — An over-constrained problem has no solution. The fix is a decision about the requirements, not more searching: identify which constraints are soft and agree which to relax. Dropping a hard constraint at random produces an invalid solution.
117. **B** — Time = distance ÷ speed = 90 ÷ 60 = 1.5 hours, and 8:15 + 1:30 = 9:45 am. The 32 passengers are irrelevant information; spotting and ignoring it is part of problem analysis.
118. **D** — With n = 10^5, n $\log_2$ n is about 1.7 × 10^6 operations, well inside 10^8. n^2 is 10^10, about 100 times too many, and n^3 and 2^n are far worse. Reading input limits this way tells you which algorithms are worth considering.
119. **C** — Split by the last digit. If it is 0: 9 choices for the first digit and 8 for the middle = 72. If it is 2, 4, 6 or 8: the first digit cannot be 0 or the last digit (8 choices) and the middle has 8 choices, so 4 × 64 = 256. Total 72 + 256 = 328.
120. **C** — A ends at 2. B runs 2–5 and C runs 2–3 in parallel. D waits for the later of them, B at 5, and ends at 7. 8 hours adds every task in sequence, which ignores that B and C can overlap.
121. **C** — Decompose into normal and overtime hours: 40 × 500 = 20,000 and 6 × 750 = 4,500, total 24,500. 23,000 ignores overtime, and 34,500 pays every hour at the overtime rate.
122. **D** — List by x: x = 2 allows y = 0–3 (4 pairs), x = 3 allows 3, x = 4 allows 2 and x = 5 allows 1. Total 4 + 3 + 2 + 1 = 10. 21 counts all pairs with x + y ≤ 5 and ignores x ≥ 2.
123. **A** — Perimeter 20 means length + width = 10, but 1 × 9 gives area 9 and 5 × 5 gives 25. Without one more fact the area is not fixed. Assuming a square, which gives 25, adds a constraint the problem never stated.
124. **D** — "abcdefghijklm" has 13 characters, one more than the 12 allowed. The others have 6, 12 and 9 characters, all inside the 6–12 range, so they must be accepted.
125. **D** — The average is sum ÷ count, and an empty list has count 0, so the division fails. The specification should say what to return in that case (an error, 0 or None). The other inputs are ordinary cases.
126. **C** — Statement (1) gives age + 5 = 20, so Ali is 15. Statement (2) only compares Ali with Sara and gives no number, so it adds nothing. Recognising when you already have enough data is a core analysis skill.
127. **C** — The minimum number of rooms equals the largest number of meetings running at once. From 10:30 to 11:00 the 9:30, 10:00 and 10:30 meetings overlap (3), and from 11:00 the 10:00, 10:30 and 11:00 meetings overlap (still 3). No moment has 4, so 3 rooms suffice.
128. **A** — Testing each independent part separately needs 3 × 4 = 12 cases. Testing every combination of the three parts would need 4^3 = 64. Independence is what lets decomposition cut the testing effort.
129. **B** — First digit: 9 choices (1–9). Then 9 choices remain for the second (0 is back in play), 8 for the third and 7 for the fourth: 9 × 9 × 8 × 7 = 4,536. 5,040 is 10 × 9 × 8 × 7, which ignores the no-leading-zero rule.
130. **C** — There are C(8, 5) = 56 committees in total. The only ones breaking the rule have 4 CS members: C(4, 4) × C(4, 1) = 4. So 56 − 4 = 52. Note that 5 CS is impossible because there are only 4 CS students.
131. **A** — From (1), x is 3 or −3, so (1) alone leaves two answers. (2) alone allows any positive number. Together they rule out −3 and give x = 3.
132. **A** — Concluding cause from timing assumes no other factor (exam season, a viral post, a price change) explains the rise. Finding hidden assumptions like this is part of analysing a problem before acting on it.
133. **D** — Starting at 2:30 pm at the earliest and lasting 3 hours means ending at 5:30 pm or later, which breaks R3. Any two of R1–R3 alone can be met, and R4 is about capacity, not time, so it conflicts with none of them.
134. **D** — Constraints tell you what is affordable. 2^20 = 1,048,576, and multiplying by 20 gives about 2.1 × 10^7 operations. A small n is often a hint that an exponential search is the intended solution.
135. **B** — Each task must come after all the tasks it depends on. R first, then D and U in either order, then C, then T. The other orders place C before D or U, or D before R.
136. **A** — Multiples of 3: 16. Multiples of 5: 10. Multiples of both (15, 30, 45): 3. By inclusion–exclusion, 16 + 10 − 3 = 23. 26 double-counts the common multiples, 27 counts the numbers divisible by neither, and 3 counts those divisible by both.
137. **A** — With a repeated maximum, "second largest" could mean 9 (second element in sorted order) or 7 (second distinct value). The spec does not say which. The other lists have distinct values, so every reading gives the same answer.
138. **C** — 1000 = 3 × 333 + 1, so one unit is left over. Whole-number shares and an exact total cannot both hold here. Analysis should ask the stakeholder: does one person pay 334, or is rounding allowed?

139. **C** — B is older than C, and C is not the youngest, so someone is younger than C, and it must be A. The order from youngest is A, C, B, so B is oldest. This also satisfies "A is not the oldest".
140. **A** — The hash-set plan is O(n) expected time, about 10^6 operations. Nested loops are $O(n^2)$, about 5 × 10^11 pairs, and subsets are exponential. The input size constraint rules out the obvious brute force.
141. **C** — A postcondition describes the required result after the solution runs. A precondition is what must hold before it starts. Together they define what a correct solution must do.
142. **C** — Stakeholders include anyone who uses, pays for, maintains or is affected by the solution. Their needs become requirements, so missing a stakeholder often means missing a requirement.
143. **A** — Pólya separates the unknown (the two numbers), the data (30 and 6) and the condition that links them (sum 30, difference 6). Adding the equations is part of a plan, not of the problem.
144. **B** — An explicit constraint is stated in the problem itself. A non-negative age is implicit: it follows from meaning, not from the text. "Look neat" is a vague preference, and a personal preference is not a constraint.
145. **A** — Requirements elicitation uses interviews, questionnaires, observation and workshops with users. Testing, compiling and reviewing code all happen after requirements have been gathered.
146. **C** — A phrase that a reasonable reader can take in two ways is ambiguous. The deadline matters, so the analyst should confirm the actual date instead of guessing.
147. **A** — Scope draws the boundary of the problem, for example "result cards for undergraduate students only, not postgraduates". Without it, requirements keep growing. Language and variable names are later design choices.
148. **B** — A test case gives an input and the result the requirement says should follow. Without the expected result, nobody can decide whether the test passed or failed.
149. **D** — Asking "why?" several times moves from the symptom, such as late results, to the underlying cause, such as a manual data-entry step. Fixing the root cause stops the problem from returning.
150. **B** — A fishbone diagram puts the problem at the head and lists possible causes on the bones, grouped under headings such as people, process and equipment. A timeline of tasks is a Gantt chart.
151. **A** — Verification checks the solution against its specification. Validation checks that the specification itself matches what users really need. Whether users wanted it at all is a validation question.
152. **D** — Constraints, such as n ≤ 1,000, limit acceptable solutions. Assumptions, such as "every month has 30 days", fill gaps in the problem and should be written down and checked, because a wrong assumption gives a wrong answer.
153. **B** — An acceptance criterion, such as "a result card downloads as a PDF within 5 seconds", turns a requirement into something that can pass or fail. It is how both sides agree the requirement is met.
154. **C** — A traceability matrix maps each requirement to the design elements and test cases that address it. A requirement with no test is a visible gap, and a changed requirement shows exactly what must be re-checked.
155. **C** — Adding the two counts includes every shared item twice. Subtracting |A ∩ B| once leaves each item counted exactly once. If A and B do not overlap, the formula reduces to |A| + |B|.
156. **C** — "At most 3" means 3 or fewer, so 0 is allowed as well. Reading it as "exactly" or forgetting the zero case are common mistakes when turning wording into constraints.
157. **A** — Every multiple of 6 is even, so divisibility by 2 is necessary. But 4 is even and not a multiple of 6, so it is not sufficient. Divisibility by both 2 and 3 is necessary and sufficient.
158. **B** — A structure chart is a tree: the whole problem at the top and its sub-problems below. It shows what the parts are, not the order of execution, which is what a flowchart shows.
159. **D** — MoSCoW splits requirements into Must, Should, Could and Won't have this time. The last group is deliberately left out of the current release so the team can focus. "Could have" is the lowest included priority.
160. **B** — Feasibility analysis happens before building. It asks whether the solution is technically possible, affordable and achievable in time. Compiling and testing come much later.
161. **C** — Equivalence partitioning assumes all values in a class behave alike, so one value per class is enough. Boundary value analysis instead picks values at the edges, such as 0, 1, 100 and 101.
162. **C** — Sommerville splits non-functional requirements into product (speed, reliability), organisational (company standards and processes) and external (laws, regulators, ethics). A law comes from outside the organisation, so it is external.

163. **A** — The requirement comes from the client organisation's own policy, so it is organisational. It does not describe how the product behaves (product) or come from law (external), and it is not a service the system offers (functional).

164. **D** — Requirements should say what is needed, such as "find a record by roll number within 1 second". Choosing a linked list is a design decision, and fixing it early rules out better structures for no stated reason.

165. **C** — The tank must lose 500 − 300 = 200 litres, and 200 ÷ 20 = 10 hours. The tap colour and the clock time do not affect the answer; dropping them is abstraction. 15 uses 300 ÷ 20, and 25 is the time to empty the tank.

166. **A** — Resetting a password is a service the system provides, so it is functional. Availability is a quality measured over time, so it is non-functional.

167. **D** — 29 February exists only in leap years, so the function must decide what a non-leap-year birthday means. The other dates occur every year and follow the normal rule.

168. **A** — Average speed is distance ÷ time, so both are needed: 120 ÷ 2 = 60 km/h. Either statement alone leaves the other quantity unknown.

169. **B** — There are 5 choices for the first letter, 4 for the second and 3 for the third: 5 × 4 × 3 = 60. $125 = 5^3$ allows repeats; 10 counts unordered groups of three letters.

170. **D** — First make the units match: 2.5 kg = 2,500 g, which is 5 lots of 500 g, so 5 × 180 = Rs 900. Rs 450 multiplies 180 by 2.5 as if the price were per kilogram.

171. **A** — $n^2 = 2.5 \times 10\text{^}7$, which fits. $n^3 = 1.25 \times 10\text{^}11$ and 2^n are far too large. O(n log n) fits easily, but it is faster than $O(n^2)$, so it is not the slowest acceptable choice.

172. **D** — The longest chain is A → C → E = 9 h. The chain A → B → D → E takes 8 h, so B (or D) has 1 hour of slack. A, C and E lie on the longest chain, so any delay to them delays the finish.

173. **D** — Split the units into slabs: 100 × 10 = 1,000, then 100 × 15 = 1,500, then 50 × 20 = 1,000, total Rs 3,500. Charging all 250 units at Rs 20 gives 5,000, and at Rs 15 gives 3,750.

174. **D** — Cricket or football = 25 + 20 − 8 = 37, so 40 − 37 = 3 play neither. Forgetting to subtract the 8 who play both gives 45, more than the class, which shows double counting.

175. **C** — There are 4! = 24 orders in all. A and B are together in 2 × 3! = 12 of them (treat AB as one block, in either order). So 24 − 12 = 12 orders keep them apart.

176. **C** — If n were odd, n^2 would be odd, so (1) forces n to be even. If n + 3 is odd, n = odd − 3 is even, so (2) also settles it. Each statement answers the question by itself.

177. **A** — Convert to the same unit: 450 cm ÷ 30 = 15 tiles along one side and 360 cm ÷ 30 = 12 along the other, so 15 × 12 = 180. Treating a tile as 0.3 m^2 instead of 0.09 m^2 gives 54.

178. **A** — The numbers are 54, 63, 72, 81 and 90. Missing 90, where the second digit is 0, gives 4. 9 counts every two-digit number with digit sum 9 and ignores the "greater than 50" constraint.

179. **B** — 60 falls in both B (60–79) and C (40–60), so the specification gives two grades for one mark. The other boundaries are clean: 40 is C, 80 is A and 39 is F.

180. **B** — They meet again after the least common multiple of 12 and 18, which is 36 minutes, so 7:36. Adding the intervals gives 7:30, and multiplying them (216 minutes) gives a far later meeting.

181. **D** — Count the complement: codes with no 7 at all number $9^4 = 6{,}561$, so 10,000 − 6,561 = 3,439 contain at least one 7. 4,000 counts a 7 in each of four positions and so counts codes with several 7s more than once. 2,916 is the count with exactly one 7.

182. **D** — 3y must leave an even remainder, so y is even: y = 2, 4, 6, 8 gives x = 12, 9, 6, 3. y = 0 or y = 10 would make y or x zero, and the problem requires positive values. Allowing zeros gives 6.

183. **D** — (1) alone fails: x = −3, y = 1 gives $x^2 > y^2$ but x < y. (2) alone says nothing about y. Together, x > 0 and $x^2 > y^2$ mean |y| < x, so y < x: the answer is always yes.

184. **B** — Each 10 km costs Rs 600. The first 10 km is split three ways (200 each), the second two ways (300 each), and Ali pays the last 600 alone: 200 + 300 + 600 = 1,100. Rs 900 splits the fare in proportion to distance (30 out of 60 rider-km), which is a different rule.

185. **B** — Divisible by at least one: 50 + 33 + 20 − 16 − 10 − 6 + 3 = 74 (by inclusion–exclusion), so 100 − 74 = 26 are divisible by none. Forgetting to add back the 3 multiples of 30 gives 29.

186. **B** — 50 ÷ 5 = 10 gaps, and a straight fence with posts at both ends has one more post than gaps, so 11. This fencepost error comes from not modelling the problem carefully.

187. **A** — "Slow" is vague, so the first job is to turn it into something measurable: which screen, how many seconds, what device and network. Only then can the cause be found.
188. **A** — A guest without an account cannot log in, let alone use two-factor authentication, so R1 and R2 cannot both hold as written. The client must decide, for example by exempting guest checkout from R1.
189. **C** — Halving the time only works if the second worker adds the same output and the work splits cleanly. A slower helper, or two people crowding one wall, breaks it.
190. **B** — A → B → C → A is a circular dependency: each task waits for another in the cycle, so none can start. The sub-problems must be redrawn, for example by moving shared work into a new task they all depend on.
191. **C** — Statement (2) is just (1) multiplied by 2, so it adds no new information. With one real equation and two unknowns, many prices fit: pen Rs 10 and pencil Rs 20, or pen Rs 20 and pencil Rs 5.
192. **D** — The odd/even rule holds only from January to July. August (8) is even but has 31 days, so the function wrongly returns 30. April (30), July (31) and June (30) happen to fit the rule.
193. **A** — 10! × 10 ≈ 3.6 × 10^7, which fits. 15! × 15 ≈ 2 × 10^13 does not, and 20! and 100! are far larger. Factorial growth means even a small rise in n makes brute force impossible.
194. **B** — Average speed = total distance ÷ total time = 240 ÷ (2 + 3) = 48 km/h. He spends longer at 40 km/h, so the slower speed carries more weight. A simple mean only works when equal times are spent at each speed.
195. **D** — Within groups, B wins: 98% against 95% for mild cases and 40% against 30% for severe. A's overall figure is high because 400 of its 500 patients were mild. Comparing totals hides a factor that changes the picture (Simpson's paradox).
196. **C** — Any 3 of 4 people must include R or S. Taking R forces P, taking S forces Q, and the 3-person teams PQR, PQS, PRS and QRS each break a rule. The constraints over-constrain the problem, which should be reported.
197. **B** — Prefix sums take O(n) to build and O(1) per query, about 2 × 10^5 steps in total. Looping over each range can cost 10^5 × 10^5 = 10^10 steps. The constraints, not just correctness, decide which plan is acceptable.
198. **A** — If daily sales were identical, February would still total 28/31 ≈ 90% of January, roughly 10% less. Normalising per day may show the drop is just the shorter month, not a change in customers.
199. **D** — R must be in seat 2 or 3, and P–Q need two neighbouring seats. If R is in 2, P–Q take 3–4 and S is in 1, which is banned. So R is in 3, P–Q take 1–2, and S is left with seat 4. That is the only arrangement.
200. **C** — With 24 students, each of the 12 months could hold exactly 2, so 3 in one month is not guaranteed. The 25th student must push some month to 3. This is the pigeonhole principle: 2 × 12 + 1.
201. **A** — A proposition is a declarative sentence with a definite truth value. "Lahore is the capital of Punjab" is true. A command and a question have no truth value, and x + 2 = 5 is an open sentence whose truth depends on x.
202. **D** — Deduction moves from general premises to a conclusion that must follow. Induction generalises from examples, so its conclusions are probable, not certain. Analogy and abduction (inference to the best explanation) are also non-certain.
203. **B** — The contrapositive negates both parts and swaps them: ¬q → ¬p. It is logically equivalent to p → q. q → p is the converse and ¬p → ¬q is the inverse; neither is equivalent to the original.
204. **A** — ∧ means AND. The conjunction is true in exactly one of the four rows of its truth table, the row where both p and q are true.
205. **D** — A conditional only makes a promise when p is true. It is broken only if p is true and q is false. When p is false the conditional is true whatever q is (it is vacuously true).
206. **D** — A tautology is true in every row of its truth table. p ∨ ¬p is the standard example. A statement false in every row is a contradiction, and one that is sometimes true and sometimes false is a contingency.
207. **C** — Validity is about form: if the premises were true, the conclusion would have to be true. A valid argument can still have false premises. Truth of premises is checked separately, and a true conclusion alone does not make an argument valid.
208. **A** — Soundness = validity + true premises. A sound argument's conclusion is therefore certainly true. "All birds can fly; a penguin is a bird; so a penguin can fly" is valid but not sound, because the first premise is false.
209. **B** — In "All A are B; all B are C; so all A are C", B is the middle term: it links the two premises and then drops out. The subject (A) and predicate (C) of the conclusion are the minor and major terms.
210. **D** — p ↔ q ("p if and only if q") is true when both are true or both are false. It is false when they differ, which is exactly when exclusive OR is true.

211. **C** — The converse swaps the two parts: q → p. It is not equivalent to the original, since the ground can be wet from a hose. The second option is the inverse and the third is the contrapositive.
212. **A** — The form is p → q, q, therefore p, which is invalid: the library might also open on Saturday. Denying the antecedent is p → q, ¬p, therefore ¬q. Modus tollens is a valid form, not an error.
213. **A** — The negation of "for all x, P(x)" is "there exists x with not P(x)". One failure is enough to make "all passed" false. "No student passed" is much stronger than the negation and is a classic trap.
214. **A** — De Morgan's laws: ¬(p ∧ q) ≡ ¬p ∨ ¬q, and ¬(p ∨ q) ≡ ¬p ∧ ¬q. Negation flips AND to OR (and vice versa) and negates each part. A truth table confirms the four rows match.
215. **C** — Induction generalises from observed cases, so its conclusion can be wrong (black swans exist). The other three apply a general rule to a specific case, which is deduction, and their conclusions follow necessarily.
216. **B** — XOR is true when the inputs differ: one true, one false. Inclusive OR (∨) is also true when both are true, which is the difference between the two.
217. **C** — Each variable is true or false, so n variables give 2^n combinations: $2^4 = 16$ rows. 8 is for 3 variables, and 4 confuses the number of variables with the number of rows.
218. **A** — p → q is T → F, which is false, so X is false whatever q → r is. For Y, p ∨ q is true and r is true, so T → T is true.
219. **B** — This is modus ponens: p → q and p give q. The second option reverses the rule (the converse), which is false: 9 is divisible by 3 but not by 6.
220. **D** — This is the classic Barbara syllogism: All A are B, all B are C, so all A are C. The premises do not tell you whether every logical thinker is a programmer, so the second and third options do not necessarily follow.
221. **D** — The students who are athletes are also disciplined, so at least some students are disciplined. Nothing is said about students who are not athletes, so "all students" goes too far, and the other two conclusions are not supported.
222. **B** — A conditional is false only when its left side is true and its right side is false. p must be true and q ∨ r false, which needs q and r both false. That is exactly one row.
223. **B** — p → q is false only when p is true and q is false; ¬p ∨ q is false in exactly that row too, so they match in all four rows. p ∨ ¬q is equivalent to q → p, the converse.
224. **B** — Contrapositive: negate both parts and swap them (¬q → ¬p). It is equivalent to the original. The second option is the converse and the third is the inverse; neither follows from the original.
225. **D** — ¬(r → c) ≡ r ∧ ¬c. The original promise is broken only when it rains and the match still goes ahead. The negation of a conditional is not another conditional, which is the common mistake.
226. **A** — A cannot be a knight, because a knight would not call himself a knave. So A is a knave and his statement is false. "Both are knaves" being false while A is a knave means B is a knight.
227. **D** — The antecedent 2 + 2 = 5 is false, and a conditional with a false antecedent is true whatever the consequent is (vacuous truth). It is a proposition because both parts have definite truth values.
228. **D** — When p is true the expression reduces to q, which is true in 2 of the 4 rows. When p is false it reduces to r, also true in 2 of 4. Total 2 + 2 = 4 rows.
229. **C** — Chain the clues from tallest: S > P > Q > R. Every clue fits, and R sits at the bottom. Writing the relations as one ordered chain avoids mixing up "taller" and "shorter".
230. **C** — This is modus tollens: p → q and ¬q give ¬p. If the server were up, the page would have loaded; it did not, so the server is not up. It is a valid form, so a conclusion does follow.
231. **C** — The first is modus tollens (p → q, ¬q, so ¬p), which is valid. A truth-table check finds a row where the premises are true and the conclusion false for each of the others.
232. **C** — "Some engineers are poets" means at least one person is both, so that person is a poet who is an engineer: "some poets are engineers" follows by simple conversion. The two "some" premises do not have to share members, so nothing links engineers with singers.
233. **C** — A statement and its contrapositive always agree in every row of the truth table. The converse and the inverse each differ from p → q in two rows, and ¬(p ∧ q) equals ¬p ∨ ¬q, not ¬p ∧ ¬q.
234. **B** — The first is the modus ponens form written as one statement; it is true in all four rows. The others each have a false row: the second when p is false and q true, the third when p is false and q true, the fourth whenever p and q differ.

235. **B** — Try both cases for A. If A is a knight, B is a knave, so B's claim is false and C differs from A: C is a knave. If A is a knave, B is a knight, so A and C match: C is a knave. A and B differ between the cases, but C is a knave in both.

236. **A** — "Not s unless r" means "if not r then not s", whose contrapositive is $s \to r$. Registering is necessary, not sufficient, so the statement does not promise that registration lets you sit.

237. **B** — "p is necessary for c" means $c \to p$: you cannot have the certificate without passing. It does not say passing is enough. The other three options all express $p \to c$ in different words.

238. **A** — T takes seat 3. S cannot be in seats 2 or 4, so S is at an end, and Q takes the other end. P and R fill seats 2 and 4, and R is right of P, so P is in seat 2. Q and S could swap ends, but seat 2 is fixed.

239. **D** — In symbols the statement is $\forall c\, \exists s\, F(c, s)$. Negating flips each quantifier and negates the inside: $\exists c\, \forall s\, \neg F(c, s)$, meaning there is a course in which no student failed.

240. **C** — From p and $p \to q$ we get q, then from $q \to r$ we get r, which contradicts $\neg r$. So no row makes all four true. Each of the other sets has a satisfying row, so it is consistent.

241. **A** — The inverse negates both parts without swapping them: $\neg p \to \neg q$. Swapping gives the converse $(q \to p)$, swapping and negating gives the contrapositive $(\neg q \to \neg p)$, and $p \wedge \neg q$ is the negation of $p \to q$.

242. **A** — A proposition must be either true or false. "x + 3 = 7" is an open sentence: it is true when x = 4 and false otherwise, so it has no truth value until x is fixed. It is not a command, it is not always false, and numeric statements such as "2 + 2 = 4" are propositions.

243. **D** — $p \vee q$ is false in exactly one row, where p and q are both false. AND is false in three rows, the conditional is false only when p is true and q is false, and the biconditional is false whenever p and q differ.

244. **C** — A contingency is neither always true nor always false. A tautology is true in every row, a contradiction is false in every row, and a fallacy is a flawed argument pattern, not a type of statement.

245. **C** — p and $\neg p$ can never both be true, so $p \wedge \neg p$ is false in every row. The other three are tautologies: each is true whether p is true or false.

246. **D** — Hypothetical syllogism chains two conditionals: if p leads to q and q leads to r, then p leads to r. Modus ponens needs p itself as a premise, disjunctive syllogism works on an OR, and contraposition rewrites one conditional as $\neg q \to \neg p$.

247. **B** — Disjunctive syllogism: at least one of p, q is true, and p is not, so q must be. Modus tollens works on a conditional, addition goes from p to $p \vee q$, and simplification goes from $p \wedge q$ to p.

248. **C** — A conditional $a \to b$ is equivalent to $\neg a \vee b$. With $a = \neg p$ this gives $\neg\neg p \vee q$, which is $p \vee q$. $\neg p \vee q$ is the form of $p \to q$, not $\neg p \to q$.

249. **D** — "p only if q" means p cannot happen without q, so whenever p is true q is true: $p \to q$. The trap is reading "if q" and writing $q \to p$. "p if and only if q" would be the biconditional.

250. **B** — The biconditional means the implication holds in both directions, so it is the conjunction $(p \to q) \wedge (q \to p)$. Joining them with OR gives a tautology, $p \wedge q$ misses the row where both are false, and the last option is exclusive OR, the negation of $p \leftrightarrow q$.

251. **A** — "Some are expensive" claims at least one expensive book exists. Its negation says there is none: no books are expensive. "Some are not expensive" can be true at the same time as the original, so it is not the negation.

252. **B** — Abduction starts from an observation and picks the most plausible explanation for it. The conclusion is likely, not guaranteed, so it is not deduction; it does not assume what it sets out to prove, and it does not rewrite a conditional.

253. **B** — $\exists$ is the existential quantifier, read "there exists". $\forall$ is the universal quantifier ("for all"), $\neg$ is negation and $\oplus$ is exclusive OR.

254. **B** — "Not everything has property P" means at least one thing lacks it: $\exists x\, \neg P(x)$. $\forall x\, \neg P(x)$ ("nothing has P") is far stronger than the negation and is the usual trap.

255. **C** — Push the negation inward one quantifier at a time, flipping each: $\neg\exists x$ becomes $\forall x\, \neg$, then $\neg\forall y$ becomes $\exists y\, \neg$. The result is $\forall x\, \exists y\, \neg P(x, y)$. The order of the variables does not change.

256. **C** — A categorical syllogism has exactly two premises (major and minor) and one conclusion, linked by a middle term that appears in both premises.

257. **D** — The middle term "mammals" is never used to cover all mammals, so the two premises only place cats and dogs somewhere inside the same large group. That does not link cats to dogs. The other options name errors about conditionals or about assuming the conclusion.

258. **C** — The form is $p \to q$, $\neg p$, so $\neg q$, which is denying the antecedent. The office could be closed for some other reason. Modus tollens is the valid form $p \to q$, $\neg q$, so $\neg p$.
259. **D** — NAND (not both) can express NOT (p NAND p), and from NOT and NAND you can build AND and OR, so it is functionally complete. AND, $\leftrightarrow$ and $\oplus$ used alone cannot produce $\neg p$ from p (AND and $\leftrightarrow$ only ever give p or true from a single p, $\oplus$ only p or false), so none of them is complete.
260. **A** — A universal negative converts simply: if no A is a B, then no B can be an A, because that would make some A a B. The other three statements contradict "No A are B" or do not follow from it.
261. **C** — "Some S are not P" is exactly what is needed to refute "All S are P", so whenever one is true the other is false. That is the contradictory relation. Contraries are "All S are P" and "No S are P".
262. **C** — Validity depends only on the form: the conclusion cannot be false if all premises are true. Soundness also needs true premises, so this argument is valid but unsound. Its conclusion may still happen to be true; a false premise does not make it false.
263. **C** — A universal claim is refuted by a single case that breaks it. 2 is prime and even, so the claim is false. Listing odd primes cannot prove a universal claim, and facts about odd numbers that are not prime say nothing about whether every prime is odd.
264. **D** — Absorption says that $p \vee (p \wedge q)$ and $p \wedge (p \vee q)$ both reduce to p. The distributive law spreads $\wedge$ over $\vee$, De Morgan's laws negate a conjunction or disjunction, and the idempotent law says $p \vee p \equiv p$.
265. **C** — A truth table needs one row per combination of distinct variables, 2^n rows for n variables. There are 5 distinct variables, so $2^5 = 32$ rows. Repeating p does not add rows.
266. **A** — $p \leftrightarrow q$ is true only when p and q match, and here they differ, so it is false. $p \to q$ is true because the antecedent is false, $p \vee q$ is true because q is true, and $\neg p \wedge q$ is $T \wedge T$ = true.
267. **B** — A chain of exclusive ORs is true when an odd number of its inputs are true. Rows with exactly one true variable (3 rows) or all three true (1 row) give 4 true rows.
268. **A** — The conditional is false only when $(p \vee q)$ is true and r is false. $p \vee q$ is true in 3 of the 4 combinations of p and q, and each needs r false, so 3 rows are false.
269. **C** — This is disjunctive syllogism: from $p \vee q$ and $\neg q$, conclude p. Since the USB option is ruled out, the report must be in Google Drive.
270. **A** — Chaining $p \to q$ and $q \to r$ gives $p \to r$ (hypothetical syllogism). The premises never say Ali does study, so "Ali passes" does not follow. The other two options are the converse and the inverse of the valid conclusion.
271. **D** — Every snake sits inside the reptile circle, and the reptile circle does not overlap the mammal circle, so no snake can be a mammal. The other options contradict the premises or reverse "All snakes are reptiles".
272. **A** — The unemployed graduates may or may not include doctors: a Venn diagram can be drawn either way. Since the middle term "graduates" is only partly covered in the second premise, no conclusion links doctors and unemployment.
273. **A** — The contrapositive of $p \to q$ is $\neg q \to \neg p$. Its converse swaps the two parts: $\neg p \to \neg q$, which is the inverse of the original statement.
274. **B** — "g only if s" means going requires sunshine, so $g \to s$. It does not promise that I will go whenever it is sunny ($s \to g$), and it is not a two-way statement.
275. **D** — "Not every studier passes" is $\neg\forall x\,(S(x) \to P(x))$, which becomes $\exists x\,\neg(S(x) \to P(x))$ and then $\exists x\,(S(x) \wedge \neg P(x))$: some student studied and did not pass. $\exists x\,(S(x) \to \neg P(x))$ is already true for any student who did not study, so it says too little.
276. **B** — A fails at $x = 1$, because $1^2 = 1$ is not greater than 1, so A is false. B is true because $x = 2$ gives $4 = 4$. One counterexample is enough to falsify a $\forall$ claim, and one witness is enough to make a $\exists$ claim true.
277. **B** — A says every number has a larger one, which is true (take $y = x + 1$). B says a single number is larger than every number, including itself, which is false. Swapping the order of $\forall$ and $\exists$ changes the meaning.
278. **A** — If A were a knave, then at least one of them would be a knave, so A's statement would be true, which a knave cannot say. So A is a knight, and the statement is true: since A is not a knave, B must be.
279. **D** — If A is a knight, B is a knight, but then B's claim that they differ would be false, which is impossible. So A is a knave and B is a knave too (A's claim is false). B's claim that they differ is then false, which fits B being a knave.
280. **B** — The order from the front is Sara, Hina, Zara, Ayesha, which uses all four people and every clue. Hina is second.

281. **A** — Position from bottom = total − position from top + 1 = 35 − 12 + 1 = 24. Forgetting the +1 gives 23, the usual slip.
282. **C** — ¬(p → q) is p ∧ ¬q, so the expression is (p ∧ ¬q) ∨ q. Distributing gives (p ∨ q) ∧ (¬q ∨ q) = (p ∨ q) ∧ T = p ∨ q. It is false when p and q are both false, so it is not a tautology.
283. **B** — Factor out p: p ∧ (q ∨ ¬q) = p ∧ T = p. Whatever q is, the expression is true exactly when p is true.
284. **D** — p ⊕ q is true (they differ) and ¬r is true, so the biconditional T ↔ T is true. p → (q ∨ r) is T → F = false, (p ∧ ¬q) → r is T → F = false, and ¬(p ∨ r) is ¬T = false.
285. **B** — For p → q the contrapositive is ¬q → ¬p. Here p = "not even" and q = "not divisible by 4", so ¬q → ¬p is "divisible by 4 → even". The last option just restates the original, and the second is false (6 is even but not divisible by 4).
286. **A** — Affirming the consequent has the form p → q, q, so p. Here q (divisible by 5) is affirmed for 25 and p (ends in 0) is wrongly concluded; 25 ends in 5.
287. **C** — From q → r and ¬r, modus tollens gives ¬q. From p → q and ¬q, modus tollens gives ¬p. A truth table confirms there is no row where all three premises are true and ¬p is false.
288. **D** — With R → M, R gives M by modus ponens, and M → E then gives E. Both missing links are needed: one to trigger the first conditional and one to connect postponement to an empty stadium.
289. **D** — Take p, q, r all false: (p → q) → r is T → F = false, but p → (q → r) is true because p is false. So the conditional is not associative. The other three pairs match in all 8 rows.
290. **A** — A and B contradict each other, so exactly one of them is truthful. Since only one person tells the truth, C must be lying, so C broke the vase. Then A's claim is false and B's is true, which fits.
291. **A** — The lazy employees cannot be managers, since no manager is lazy. So at least those employees are not managers. Nothing is said about the other employees, and nothing about managers outside the employee group.
292. **D** — The argument generalises from a large random sample, which is induction. Its conclusion is well supported but could still be false if an untested bulb is faulty.
293. **D** — "Reliable" and "never fails" mean nearly the same thing, and each is offered as the reason for the other. No independent evidence is given, which is circular reasoning (begging the question).
294. **C** — Three classmates cannot represent all candidates nationwide, so the general conclusion is not justified by the evidence. This is a hasty generalisation, a weak inductive argument.
295. **B** — B cannot be a knight, since a knight cannot call himself a knave, so B is a knave and not all three are knaves. If A is a knight, B is the only knave and C is a knight. If A is a knave, there are at least two knaves but not three, so C is a knight. C is a knight either way; A is not fixed.
296. **A** — If A were a knave, the statement would be false, and a conditional is false only when its "if" part is true, meaning A is a knight. That contradicts A being a knave, so A is a knight. The statement is then true with a true "if" part, so B is a knight.
297. **C** — The rule is (Programmer ∧ Python) → Loops. Modus tollens with ¬Loops gives ¬(Programmer ∧ Python): Hina is not both. She may be a programmer who doesn't know Python, or know Python without being a programmer.
298. **B** — D must be in position 2 or 3, and the pair A-B needs two adjacent seats with C to its left. The only arrangement is C, D, A, B, so A is in position 3.
299. **B** — The rule p → q is broken only by a card with p true and q false. The Lahore card could hide a non-blue colour, and the green card could hide Lahore. Karachi is irrelevant, and blue cannot break the rule whatever city it hides.
300. **D** — Both conditionals hold when the values never drop from true to false along p, q, r. The valid patterns are FFF, FFT, FTT and TTT, so 4 assignments.
301. **B** — A diamond holds a yes/no (true/false) test and has one flow line in and two labelled flow lines out. Rectangles are processes, parallelograms are input/output and ovals mark start or end.
302. **C** — Input and output steps use the parallelogram. A rectangle is for a process or calculation, a diamond for a decision and a small circle is an on-page connector.
303. **C** — The oval is the terminator symbol. Every flowchart begins with a Start terminator and finishes with a Stop or End terminator. Calculations go in rectangles and conditions in diamonds.

304. **A** — Structured programming builds every algorithm from three structures: sequence (steps in order), selection (IF/CASE) and iteration (loops). Input-process-output describes a system's data flow, not its control flow.
305. **D** — Finiteness (from Knuth's list of algorithm properties) means the algorithm must always terminate. Definiteness is about each step being unambiguous, and effectiveness is about each step being basic enough to carry out.
306. **A** — Pseudocode mixes plain English with programming-style keywords (IF, WHILE, OUTPUT) to describe an algorithm. It has no fixed syntax and is not compiled or run. The diagram with standard shapes is a flowchart.
307. **B** — When the number of iterations is known in advance, a count-controlled FOR loop is the natural choice: FOR i ← 1 TO 12. WHILE and REPEAT loops suit conditions that depend on data, such as reading until a sentinel.
308. **B** — Definiteness requires every step to be precisely and unambiguously defined. "A few" and "if needed" can be read in different ways, so two people could follow the step differently.
309. **A** — A WHILE loop tests its condition before the body, so the body may run zero times. A post-test loop runs the body first and tests afterwards, so the body runs at least once. Both kinds can loop forever and both can use counters.
310. **D** — A sentinel (for example -1 when all real marks are 0 or more) tells the loop to stop reading. The loop must test for the sentinel before processing it, otherwise the sentinel gets counted as data.
311. **A** — A binary decision has one entry and two exits, labelled Yes/No or True/False, one for each outcome of the test. A process box has one entry and one exit.
312. **D** — CASE chooses one branch out of several according to a value, so it is selection with more than two outcomes. It does the same job as a chain of IF ... ELSE IF tests on one variable.
313. **D** — A small circle with a matching label (A, B ...) is an on-page connector. It joins two points of the same flowchart without drawing a long arrow. A terminator is the start/end oval.
314. **B** — Böhm and Jacopini showed that sequence, selection and iteration are sufficient for any flowchart-expressible algorithm, so unrestricted GOTO is not needed. Selection alone cannot repeat work an input-dependent number of times, so iteration is required.
315. **C** — A loop invariant is a statement about the variables that holds before the first iteration and after each one (CLRS uses it to prove correctness). Combined with the exit condition, it shows the loop's result is right. It is a condition, not a constant variable.
316. **B** — With short-circuit AND, the right operand is evaluated only if the left one is true. Putting the guard b ≠ 0 first means a / b is skipped whenever b is 0. Operator precedence decides grouping, not which operand is evaluated first.
317. **D** — Statements in a sequence run in order. y becomes 4 × 2 = 8, then x is overwritten with 8 + 1 = 9. The output is 9, 8.
318. **B** — 67 >= 80 is false, so the first branch is skipped. 67 >= 60 is true, so grade becomes B and the remaining ELSE is skipped. Only one branch of an IF ... ELSE IF chain runs.
319. **C** — The loop adds 1 + 2 + 3 + 4 + 5. In pseudocode, FOR i ← 1 TO 5 includes both ends, so the total is 15. Stopping at 4 would give 10.
320. **D** — i takes the values 2, 4, 6, 8 and 10. The upper bound 10 is included, so five values are printed.
321. **A** — i goes 1 → 2 → 4 → 8 → 16 → 32 → 64. That is six doublings; when i reaches 64 the test i < 50 fails. Doubling loops run about log2(limit) times, not limit/2 times.
322. **B** — The inner loop runs i times on each pass of the outer loop: 1 + 2 + 3 + 4 = 10. It would be 16 only if the inner loop always ran 4 times.
323. **A** — x goes 10 → 7 → 4 → 1 → -2. The test runs after each subtraction; it first succeeds when x is -2, so -2 is printed.
324. **C** — After a ← b, both hold 9 and the original 5 is lost. b ← a then copies 9 back. A correct swap needs a temporary variable: temp ← a, a ← b, b ← temp.
325. **A** — Each pass adds the last digit and then removes it: 2, then 7, then 0, then 4. The sum of the digits is 2 + 7 + 0 + 4 = 13. Reversing the digits would need s ← s * 10 + n MOD 10.
326. **B** — The process box runs for i = 1, 2, 3 and 4, multiplying p by each, so p = 1 × 2 × 3 × 4 = 24. After i becomes 5 the decision answers Yes and p is printed. 120 would need i = 5 to be multiplied in too.
327. **D** — best moves to 2 when A[2] = 9 beats 3. At i = 4, A[4] = 9 is not strictly greater than 9, so best stays 2. The output is the index of the first maximum. Using >= would give 4. 9 is the maximum value, not its index.

328. **C** — Multiples of 3 up to 20 are 3, 6, 9, 12, 15 and 18, so c = 6. The ELSE IF only runs when i is not a multiple of 3, so 15 is not counted in d: d counts 5, 10 and 20, giving 3.
329. **C** — k = 1, 2, 3 make total 6; k = 4 is skipped by continue; k = 5 and 6 bring it to 17; k = 7 is added because 17 is not > 20, giving 24; k = 8 is skipped; at k = 9 the test total > 20 is true, so the loop breaks with k still 9.
330. **D** — This is Euclid's GCD algorithm. (a, b) goes (48, 18) → (18, 12) → (12, 6) → (6, 0): three passes. The loop stops when b is 0 and a holds the GCD, 6.
331. **A** — With i < n the loop stops before i reaches n, so A[n] (here 3) is never added: 4 + 1 + 7 = 12. Using i <= n makes the last pass include A[n], giving 15. This is a classic off-by-one error.
332. **A** — i takes the values 10, 7, 4, 1, -2, -5 and so on. It never equals 0, so the test i ≠ 0 stays true for ever. Using i > 0 as the condition would make the loop stop.
333. **D** — Testing the narrower condition x > 100 first, then x > 0, reproduces all three outcomes, including the boundaries x = 100 (small) and x = 0 (neg). Boundary values are where rewritten conditions usually break.
334. **B** — The test for -1 comes after the process boxes, so -1 is added to sum (4 + 8 - 1 = 11) and counted (3). The fix is to test v = -1 immediately after input, before updating sum and count, which would give 12 / 2 = 6.
335. **C** — Before the loop p = 1 = 2^0. Each pass doubles p and adds 1 to i, so p = 2^i still holds. When the loop exits, i = n, so the invariant gives p = 2^n, which proves the loop correct.
336. **D** — Loop A steps i through 1..64, so it runs 64 times. Loop B takes i through 1, 2, 4, 8, 16, 32, 64 (seven values that pass i <= 64) before i becomes 128. That is log2(64) + 1 = 7, so B grows logarithmically.
337. **C** — Short-circuit AND evaluates left to right. After the last element, i = n + 1 and A[n + 1] ≠ key is evaluated first, which reads outside the array. Writing i <= n AND A[i] ≠ key stops before the bad access.
338. **C** — In an IF ... ELSE IF chain only the first true branch runs. 90 >= 50 is true, so C is assigned and the B and A tests are never reached. Testing >= 85 first, then >= 70, then >= 50 fixes it.
339. **B** — P runs its inner statement n × n = 100 times. Q runs 1 + 2 + ... + 10 = n(n + 1)/2 = 55 times. The difference is 100 - 55 = 45. Both are $O(n^2)$, but the exact counts differ.
340. **A** — count is never updated inside the loop, so count < 10 stays true and the loop never ends. An algorithm must terminate after a finite number of steps. Adding count ← count + 1 restores finiteness.
341. **D** — A rectangle is the process box, used for calculations and assignments. The diamond is for decisions, the parallelogram for input or output, and the oval for Start and Stop.
342. **C** — Flow lines connect the symbols and show the direction of control from one step to the next. Storage, decisions and the end point each have their own symbols.
343. **D** — The predefined process symbol stands for a named routine, such as "Sort list", whose steps are drawn in a separate flowchart. It keeps the main chart short and supports modular design.
344. **B** — Knuth allows an algorithm to take no input (for example, printing the first 10 primes) but requires at least one output, because an algorithm that produces nothing is of no use.
345. **D** — Effectiveness (Knuth) means each operation is simple and doable, not something vague like "find the best route by intuition". Speed is about efficiency, output is a separate property, and algorithms need not be written in code.
346. **A** — A trace table has one column per variable and one row per step. Filling it in by hand shows exactly how the values change and is the standard way to find a logic error or predict output.
347. **D** — A counter increases by a fixed step (usually 1) to record how many times something happens. An accumulator adds varying values (such as total ← total + price), a sentinel marks the end of input, and a flag records a yes/no state.
348. **A** — A flag is a Boolean variable that records whether some event has happened. It starts FALSE and is switched to TRUE when the event occurs.
349. **A** — The same algorithm (for example, linear search) can be written as many programs in C++, Python or Java. The algorithm is the logic; the program is one concrete, executable version of it.
350. **D** — Top-down design starts with the whole task, such as "Process results", then splits it into parts like "Read marks", "Compute grade" and "Print report", refining each further. It produces modular, readable solutions.
351. **B** — A FOR loop tests its counter before each pass, like a pre-test loop. If the start value is already past the end value, the test fails at once and the body is skipped.

352. **A** — The prompt must run at least once and the number of attempts is unknown, which is exactly what a post-test loop provides. A FOR loop needs a known count, and IF or CASE cannot repeat anything.
353. **D** — Each of the m outer passes triggers n inner passes, so the inner body runs m × n times. This only holds when the inner count does not depend on the outer variable.
354. **D** — A WHILE loop repeats as long as its condition holds. If nothing in the body moves the condition towards false, it repeats forever. A condition that is true at the start only means the body runs at least once.
355. **C** — A loop variant (for example n − i) that is a non-negative integer and drops each pass cannot fall forever, so the loop must stop. An invariant proves the result is correct, not that the loop ends.
356. **B** — NOT binds tightest, then AND, then OR, just as unary minus, × and + do in arithmetic. So the expression groups as (NOT a) OR (b AND c). Use brackets in your own pseudocode to avoid relying on this.
357. **C** — CASE picks a branch by matching one expression against listed constants, such as a menu choice of 1, 2 or 3. Range tests and conditions mixing several variables are better as IF ... ELSE IF, and repetition needs a loop.
358. **C** — break leaves the innermost enclosing loop immediately. continue is the statement that skips the rest of the current iteration and moves to the next one.
359. **C** — break exits only the innermost loop that contains it. Control returns to the outer loop, which carries on with its next pass, starting the inner loop afresh.
360. **A** — In a deterministic algorithm every step is fixed by the input and the current state, so repeated runs on the same input behave identically. Randomised algorithms are the main exception.
361. **C** — Arbitrary jumps produce "spaghetti code" whose paths are hard to trace. Sequence, selection and iteration, each with one entry and one exit, are enough for any algorithm and keep the logic readable.
362. **A** — Testing before the body means the body may run zero times, which is the WHILE pattern. If the diamond came after the body, the body would always run at least once (post-test).
363. **B** — This is the dangling-else problem. Languages such as C, C++ and Java attach an ELSE to the closest IF that has none, so x ← 2 runs when a is true and b is false. END IF markers or braces remove the ambiguity.
364. **C** — Single entry, single exit means each block can be treated as one step and nested inside another without tangled jumps. That is what lets any structured program be read and checked piece by piece.
365. **A** — t keeps the old a (1). Then a takes b (2), b takes c (3) and c takes t (1). The three values rotate left: 2 3 1.
366. **D** — 14 MOD 4 = 2, so the CASE matches the branch for 2 and outputs C. The OTHERWISE branch can never run here, because n MOD 4 is always 0, 1, 2 or 3.
367. **D** — Counting down from 20 in steps of 4 gives 20, 16, 12, 8, 4. The next value, 0, is below the end value 1, so the loop stops after outputting 4.
368. **B** — i takes the values 1, 3, 5 and 7, so p = 1 × 3 × 5 × 7 = 105. 5040 would be 7!, using every number from 1 to 7, and 945 would wrongly include 9.
369. **C** — REPEAT ... UNTIL is a post-test loop, so the body runs once before the test: x becomes 10. Then 10 > 5 is true, so the loop stops and 10 is output, even though x was already above 5 before the loop.
370. **B** — Each pass moves the last digit of n onto the end of r. n goes 1230 → 123 → 12 → 1 → 0 while r goes 0 → 3 → 32 → 321. The trailing zero of 1230 becomes a leading zero, which disappears.
371. **A** — After each pass (a, b) becomes (1, 1), (1, 2), (2, 3), (3, 5), (5, 8), (8, 13). After the 6th pass a = 8, the 6th Fibonacci number. 13 is b, not a.
372. **C** — 91 is not divisible by 2, 3, 4, 5 or 6, so d climbs to 7. 91 MOD 7 = 0 (91 = 7 × 13), so isPrime becomes FALSE and d stays 7. The loop then stops because isPrime is FALSE.
373. **D** — 15 MOD 3 = 0 but 15 MOD 5 = 0 too, so NOT (x MOD 5 = 0) is false and the first test fails. The ELSE IF test is true, so B is output. Only one branch of an IF ... ELSE IF chain ever runs.
374. **B** — The inner loop runs 5, 4, 3, 2 and 1 times as i goes from 1 to 5, because j starts at i. 5 + 4 + 3 + 2 + 1 = 15. 25 would need j to start at 1 every time.
375. **D** — n goes 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1. That is 8 changes, and steps is increased once per change. 16 is the largest value reached, not the step count.
376. **C** — range(10, 0, -3) starts at 10 and steps by −3 while the value is still greater than the stop value 0: 10, 7, 4, 1. The next value, −2, is past the stop, and the stop itself is never included.
377. **A** — Odd values of i hit continue and are skipped. Even values 2, 4, 6 and 8 are added, giving 20. When i = 10, the break runs before 10 can be added.

378. **B** — The break ends only the inner loop, as soon as i * j > 6. Row i = 1 counts j = 1..4 (4), i = 2 counts j = 1..3 (3), i = 3 counts j = 1..2 (2) and i = 4 counts j = 1 (1). 4 + 3 + 2 + 1 = 10.
379. **C** — Each adjacent pair is compared once: 5 > 1 swaps, 5 > 4 swaps, 5 > 2 swaps, 5 < 8 stays. The largest value 5 bubbles right, giving [1, 4, 2, 5, 8]. A fully sorted array needs more passes.
380. **C** — The loop never stops early, so every match overwrites pos. 3 is found at index 2 and again at index 4, and the last match wins: 4. Returning the first match would need a flag check or an exit.
381. **C** — 12 ≠ 18 and 12 < 18, so b ← 6. Now 12 > 6, so a ← 6. Then a = b = 6 and 6 is output. Repeated subtraction like this finds the greatest common divisor.
382. **B** — i goes 100 → 33 → 11 → 3 → 1, using integer division by 3. That is 4 passes. The loop then stops because 1 > 1 is false.
383. **A** — The inner loop starts at i and steps by i. i = 1 adds 1 + 2 + 3 + 4 = 10, i = 2 adds 2 + 4 = 6, i = 3 adds 3 and i = 4 adds 4. 10 + 6 + 3 + 4 = 23.
384. **D** — This is fast exponentiation. n = 5 (binary 101): r becomes 3, then x = 9 with no multiply, then x = 81 and r = 3 × 81 = 243. $3^5 = 243$.
385. **B** — 137 DIV 60 = 2 whole hours, and 137 MOD 60 = 137 − 120 = 17 minutes left over. DIV drops the fraction, so 2.28 cannot appear.
386. **B** — Starting at 0 assumes some element exceeds 0. When every value is negative, none does, and the starting 0 is wrongly returned. Starting from an actual element works for all inputs.
387. **B** — REPEAT runs the body once, then repeats while c is still false. Writing the body once and then looping on NOT c reproduces that exactly, including the guaranteed first run.
388. **B** — The FOR loop includes its end value, so the test is i <= n, and the counter moves by the step after the body. For n = 5 both versions run the body with i = 1, 3, 5.
389. **A** — No number is both less than 5 and greater than 10, so the first condition is always false and the P branch is dead code. x >= 5 gives Q and x < 5 gives R.
390. **B** — Before the loop s = 0 and i = 0, and 0 × 1 / 2 = 0. Each pass raises i by 1 and adds the new i to s, which keeps s equal to 1 + 2 + ... + i. When the loop ends with i = n, the invariant gives the final answer.
391. **A** — The increment sits inside the IF, which only runs for even i. i starts at 1 (odd), so i ← i + 1 never runs and the condition i <= 10 stays true forever. Moving the increment after END IF fixes it.
392. **C** — The loop runs while i^2 <= 1000. $31^2 = 961$ passes and $32^2 = 1024$ fails, so i takes the values 1 to 31: 31 runs. The count grows like √n, not n/2.
393. **A** — At exactly 5000 the customer should get the discount, but > 5000 is false, so they don't. At every other listed value the buggy and correct versions agree. Boundary values catch most selection bugs.
394. **D** — X needs a > 0 and b > 0, Y needs a > 0 and b <= 0, and Z needs a <= 0. The first set covers all three. Full branch coverage of a nested IF needs a case for each path.
395. **B** — With a = 5 and b = 9: a becomes 14, then b = 14 − 9 = 5, then a = 14 − 5 = 9. The sum holds both values, so neither is lost. The temporary-variable version is the clearer choice in practice.
396. **C** — Starting at 0 and incrementing before output gives 1, 2, 3, 4; when i reaches 4 the test 4 < 4 fails. The order of the update and the output, together with the bound, decides every value.
397. **D** — For each i, the inner loop doubles j from 1 until j >= i, which takes $\lceil \log_2 i \rceil$ passes. For i = 1..8 that is 0, 1, 2, 2, 3, 3, 3, 3, which sums to 17.
398. **D** — n MOD 2 is the lowest binary digit, and n DIV 2 shifts it away. Adding each lowest digit counts the 1 bits: for n = 13 (1101) the output is 3.
399. **A** — Every outcome of a decision needs its own labelled exit. Without a No path, the chart does not say what happens when n <= 0, so it breaks definiteness and may not reach Stop.
400. **A** — The two loops compare every pair (i, j) with i < j, not just neighbours. found becomes TRUE as soon as any two positions hold the same value, which is a duplicate check that takes about $n^2/2$ comparisons.
401. **C** — Any base-b system uses b digit symbols, from 0 to b - 1. Hexadecimal uses 0-9 for zero to nine and A-F for ten to fifteen, which makes 16 symbols. Eight symbols (0-7) belong to octal.
402. **A** — A byte is 8 bits, so it can hold 2^8 = 256 different patterns. Four bits form a nibble, which is exactly one hexadecimal digit.

403. **A** — Abstraction keeps the essential features and leaves out irrelevant detail, for example a metro map that shows stations and lines but not street layouts. Breaking a problem down is decomposition, finding similarities is pattern recognition and writing steps is algorithm design.
404. **A** — Since 16 = 2^4, every hex digit stands for one 4-bit group: 1101 0110 becomes D6. The hardware still stores bits; hex is only a shorter way for people to write them and takes no less memory.
405. **D** — Each bit has two states and the bits are independent, so n bits give 2 × 2 × ... × 2 = 2^n patterns. As unsigned numbers these are 0 to 2^n - 1: the largest value is 2^n - 1, but the count of patterns is 2^n.
406. **D** — Two's complement has one representation of zero and ordinary binary addition gives correct signed results, so subtraction is just adding the negated value. Its positive range is about half that of unsigned binary with the same width.
407. **B** — 7 bits give 2^7 = 128 codes, numbered 0 to 127. The 256-code sets are 8-bit extended ASCII variants such as Latin-1.
408. **D** — UTF-8 is a variable-length encoding of Unicode: code points 0-127 take one byte with the same values as ASCII, and other characters (Urdu, Chinese, emoji) take 2 to 4 bytes. A fixed 2-byte scheme describes the old UCS-2 encoding.
409. **B** — An ADT is specified by its values and operations, for a stack push, pop, peek and isEmpty with last-in-first-out behaviour. Whether it is built on an array or a linked list is an implementation choice hidden behind that interface.
410. **C** — A high-level language hides the most hardware detail. Assembly gives readable names to machine instructions, machine code is the binary those instructions become, and logic gates are the circuits that execute it. Each level hides the one below.
411. **B** — 8 = 2^3, so each octal digit stands for one 3-bit group. That is why 110 101 in binary is 65 in octal. Hexadecimal digits use 4 bits.
412. **C** — In sign-magnitude, 0000 0000 is +0 and 1000 0000 is -0. Two's complement and biased forms have exactly one zero, and unsigned binary has no sign at all. (One's complement also has two zeros.)
413. **C** — Single precision uses 1 sign bit, an 8-bit exponent stored with a bias of 127, and a 23-bit fraction (24 bits of precision with the hidden leading 1). Double precision uses 1, 11 and 52.
414. **D** — A fraction terminates in base 2 only if its denominator (in lowest terms) is a power of 2. 1/10 has a factor 5, so in binary it is 0.000110011... repeating, and must be rounded. Fractions like 0.5 or 0.375 are exact.
415. **B** — Procedural abstraction lets you use a procedure through its name, inputs and output without knowing its internal steps. Data abstraction is the same idea for data types (ADTs). Decomposition is splitting a problem into parts.
416. **A** — Overflow means the true result is outside the representable range. That can only happen when adding two positives (result looks negative) or two negatives (result looks positive). A carry out of the top bit alone is normal in signed addition, for example -1 + 1, and adding opposite signs can never overflow.
417. **A** — Use place values 8, 4, 2, 1: 1×8 + 0×4 + 1×2 + 1×1 = 11. Reading the bits the other way round (1101) gives 13.
418. **D** — 25 = 16 + 8 + 1, so the bits for 16, 8 and 1 are set: 11001. Repeated division by 2 gives remainders 1, 0, 0, 1, 1, read from bottom to top. 10011 is the same remainders read top to bottom (19).
419. **D** — Each hex digit is worth 16 times the digit to its right. 2F = 2 × 16 + 15 = 32 + 15 = 47, because F stands for 15.
420. **B** — Split into 4-bit groups from the right: 1101 0110. 1101 = 13 = D and 0110 = 6, so the result is D6.
421. **D** — 157 ÷ 8 = 19 remainder 5; 19 ÷ 8 = 2 remainder 3; 2 ÷ 8 = 0 remainder 2. Reading the remainders bottom to top gives 235. Check: 2×64 + 3×8 + 5 = 157.
422. **D** — +20 is 00010100. Invert the bits (11101011) and add 1 to get 11101100. Check: -128 + 64 + 32 + 8 + 4 = -20. 10010100 is the sign-magnitude form and 11101011 is the one's complement.
423. **D** — The leftmost bit has weight -128: -128 + 32 + 16 + 2 + 1 = -77. Read as unsigned it would be 179. -51 is what the sign-magnitude reading gives.
424. **A** — n-bit two's complement covers -2^(n-1) to 2^(n-1) - 1, so 8 bits give -128 to 127. There is one more negative value than positive because zero takes one of the non-negative patterns. -127 to 127 is the sign-magnitude range.

425. **B** — 109 + 54 = 163, and 163 in binary is 10100011. Work column by column from the right, carrying 1 whenever a column sums to 2 or 3.
426. **C** — 8 bits give only 2^8 = 256 patterns, which is too few. 9 bits give 2^9 = 512, enough for 300. In general you need the smallest n with 2^n ≥ the number of items.
427. **A** — 100 × 100 = 10,000 pixels, each using 24 bits = 3 bytes, so 30,000 bytes. 240,000 is the size in bits, not bytes.
428. **A** — 7 + 1 = 8, but 4-bit two's complement only reaches 7. The bit pattern 1000 has value -8, so two positives produced a negative: signed overflow. The register stays 4 bits wide, so no fifth bit appears.
429. **C** — Left of the point: 4 + 0 + 1 = 5. Right of the point the weights are 1/2, 1/4, 1/8: 0.5 + 0 + 0.125 = 0.625. Total 5.625.
430. **B** — 12 = 1100 and 10 = 1010. XOR gives 1 where the bits differ: 0110 = 6. AND would give 1000 = 8 and OR would give 1110 = 14.
431. **B** — Roads connect cities in arbitrary ways, can form cycles and carry a cost, which is exactly a weighted graph. Shortest-path algorithms such as Dijkstra's work directly on it.
432. **C** — 8-bit unsigned values run 0-255 and arithmetic wraps modulo 256. 250 + 10 = 260, and 260 - 256 = 4. Six increments reach 0 (after 255) and four more reach 4.
433. **C** — Convert everything to decimal: 80 hex = 128, 177 octal = 127, 1111111 binary = 127 and 126 is already decimal. The hex value is largest even though it has the fewest digits.
434. **A** — Counting stops only needs the stations and their connections, so track length is irrelevant and can be abstracted away. If the goal changed to shortest distance, track length would become essential.
435. **A** — 0.1 and 0.2 are stored as the nearest binary fractions, and their sum is 0.30000000000000004, which is not the same double as 0.3. Rounding to 2 decimal places gives the double nearest 0.3, so the second comparison is True. Compare floats with a tolerance or rounding, not ==.
436. **C** — 1010 in 4-bit two's complement is -8 + 2 = -6. Sign extension copies the sign bit into the new high bits, giving 11111010, which is also -6. Padding with zeros would turn it into +10.
437. **B** — The point of an ADT is that callers rely on the operations and their behaviour, not the representation. If push, pop and isEmpty keep the same signatures and LIFO behaviour, the calling code does not change.
438. **B** — Count the 1s: 11010110 has 5 (odd), so a bit must have flipped. The others have 4, 4 and 0 ones, all even. Parity detects any single-bit error but cannot detect two flipped bits or locate the error.
439. **D** — With 24 significant bits every integer up to 2^24 = 16,777,216 is exact. 2^24 + 1 would need 25 significant bits, so it rounds to 16,777,216. Powers of two such as 2^32 are stored exactly.
440. **C** — In base b, 13 means b + 3 and 20 means 2b. So b + 3 + 5 = 2b, giving b = 8. Check: 13 octal = 11, 11 + 5 = 16 = 20 octal.
441. **A** — A nibble is 4 bits, half a byte. One hex digit describes exactly one nibble. A byte (also called an octet) is 8 bits, and a word is the processor's natural data size, often 32 or 64 bits.
442. **C** — Place values grow to the left: 1, 2, 4, 8, and so on. So the leftmost bit (the MSB) carries the largest weight, and the rightmost bit (the LSB) is worth only 1.
443. **D** — Lossless compression (ZIP, PNG, FLAC) removes redundancy without losing information, so decompression gives back the identical data. Discarding detail is lossy compression, and no method can guarantee a fixed reduction for every input.
444. **B** — JPEG throws away fine image detail to reach small sizes, so the decoded image is not identical to the original. PNG, ZIP and FLAC are lossless: they rebuild the original data exactly.
445. **C** — Big-endian machines store the most significant byte at the lowest address; little-endian machines (such as x86) store the least significant byte first. It matters when binary data moves between systems or is read byte by byte.
446. **A** — A parity bit makes the count of 1s even (or odd). If one bit flips, the count no longer matches and the error is detected. Parity cannot say which bit flipped, so it cannot correct the error.
447. **D** — One's complement negates a number by flipping all its bits. Adding 1 after inverting is two's complement, and setting only the sign bit is sign-magnitude. One's complement, like sign-magnitude, has two zeros (all 0s and all 1s).
448. **B** — The stored exponent equals the true exponent + 127, so 2^0 is stored as 127 (01111111). 1023 is the bias for double precision's 11-bit exponent, and 255 (all 1s) is reserved for infinity and NaN.

449. **A** — Double precision uses 1 + 11 + 52 = 64 bits. The 11-bit exponent (bias 1023) gives a far wider range than single precision, and 52 fraction bits give about 15 to 17 significant decimal digits.
450. **A** — An all-1s exponent is reserved. With a fraction of zero it means infinity; with any non-zero fraction it means NaN, the result of operations such as 0/0. The largest finite number has an exponent field one below all 1s.
451. **B** — In BCD, 59 is stored as 0101 1001, one nibble per decimal digit. It wastes some bit patterns (1010 to 1111 are unused) but makes decimal display and exact decimal arithmetic easy, which is why calculators and clocks use it.
452. **B** — In a Gray code, moving from one value to the next flips only one bit (000, 001, 011, 010, ...). This avoids glitches in rotary encoders, where several bits changing at slightly different moments could be misread.
453. **C** — A dictionary maps keys to values, for example roll number to student name. The ADT promises insert, look up and delete by key. Whether it is built with a hash table or a balanced tree is an implementation detail hidden by the abstraction.
454. **D** — Amounts such as Rs 0.10 have no exact binary floating-point form, so repeated additions drift by tiny errors. Storing 10 paisa as the integer 10 keeps every sum exact. Floats can be negative and usually have a larger range than integers.
455. **A** — The internal (physical) level covers storage structures and indexes. The conceptual level describes the whole database's entities and relationships, and the external level gives each user group its own view. Each level hides the details of the one below.
456. **D** — The binary prefix kibi means 2^10 = 1,024. A kilobyte (kB) in the SI sense is 1,000 bytes, 8,192 is the number of bits in a KiB, and 1,048,576 bytes is one mebibyte (MiB).
457. **C** — Each bit moves to a position worth twice as much, so the value is multiplied by 2: 0101 (5) becomes 1010 (10). A right shift halves the value, dropping any remainder.
458. **D** — Copying the sign bit keeps a negative two's complement number negative, so an arithmetic right shift divides signed values by 2 (rounding down). A logical shift treats the bits as unsigned. For non-negative values both shifts give the same result.
459. **D** — Unicode is the character set: for example, U+0627 is the Arabic letter alif. UTF-8, UTF-16 and UTF-32 are encodings that turn code points into bytes. UTF-8 uses 1 to 4 bytes and is the most common on the web.
460. **A** — Floating point spends bits on an exponent, so it can hold both tiny and huge values. The price is relative precision: near 1 the gaps are tiny, but near 10^16 a double cannot even tell n from n + 1. Fixed point keeps equal spacing over a small range.
461. **D** — 24 bits are shared equally across three channels, so each channel gets 8 bits and can take 256 levels (0 to 255). Together they give 2^24, about 16.7 million, colours.
462. **B** — The rules of chess depend only on which piece sits on which square, so the model keeps exactly that and drops physical details. Choosing what to keep for the task is the core of abstraction.
463. **B** — Hamming distance counts differing positions, which equals the number of 1s in their XOR. Error-detecting and error-correcting codes rely on keeping valid codewords a minimum Hamming distance apart.
464. **D** — A carry out of the MSB alone is normal in two's complement: -1 + 1 produces one and the answer 0 is correct. Overflow happens when the carry into the sign bit and the carry out of it disagree, which is the same as two same-sign operands giving a result of the opposite sign.
465. **A** — Convert each hex digit to 4 bits: 3 = 0011 and C (12) = 1100, giving 0011 1100. 00111010 is 3A and 11000011 is C3, with the digits reversed.
466. **A** — 100 ÷ 16 = 6 remainder 4, so the hex digits are 6 then 4: 0x64 = 6 × 16 + 4 = 100. A0 is 160, and 46 is the digits read in the wrong order (70).
467. **A** — In base 8, 17 means 1 × 8 + 7 = 15. Reading it as decimal gives 17, which ignores the base.
468. **C** — 65 + 32 = 97. The fixed gap of 32 means a single bit (value 32) separates upper and lower case, so case conversion is just flipping that bit.
469. **B** — BCD encodes each decimal digit in its own nibble: 5 = 0101 and 9 = 1001. 0011 1011 is pure binary for 59, and 1001 0101 puts the digits in the wrong order.
470. **B** — Each hex digit stands for exactly 4 bits, so 64 ÷ 4 = 16 digits. Eight digits cover only 32 bits.
471. **C** — The runs are four A, three B, two C and one D, written count first: 4A3B2C1D. Dropping the single D loses data, and A4B3C2D1 uses the opposite order to the rule stated.

472. **B** — ord('7') is 55 and ord('0') is 48, so their difference is the digit value 7. Adding 32 to the code of 'C' (67) gives 99, which is 'c'. This is how parsers turn digit characters into numbers and switch letter case.

473. **B** — Little-endian stores the least significant byte first, so addresses 100 to 103 hold 78, 56, 34, 12. A big-endian machine would put 0x12 at address 100. Bytes are never split into reversed nibbles, so 0x87 is wrong.

474. **A** — +5 is 00000101; inverting every bit gives 11111010. 11111011 is the two's complement form (invert and add 1), and 10000101 is sign-magnitude.

475. **C** — 37 is 0100101 in 7 bits. Sign-magnitude sets the leftmost bit to 1 for negative, giving 1 0100101. 11011011 is the two's complement of -37, and 11011010 is its one's complement.

476. **D** — AND keeps a bit only where both inputs are 1, so the mask 00001111 clears the high nibble and keeps the low nibble 1100. 10101111 is the OR result and 10100011 is the XOR result.

477. **A** — << 2 multiplies by $2^2 = 4$, so 13 becomes 52. >> 1 divides by 2 and drops the remainder, so 13 becomes 6 (1101 becomes 110).

478. **B** — Multiply the fraction by 2 repeatedly: 0.375 → 0.75 (0), 0.75 → 1.5 (1), 0.5 → 1.0 (1). Reading the integer parts top to bottom gives 0.011 = 1/4 + 1/8. 0.101 is 0.625.

479. **D** — Size = 10 × 44,100 samples × 2 bytes = 882,000 bytes. 7,056,000 is the size in bits (not divided by 8), and 1,764,000 would be stereo.

480. **C** — 10010110 is 150 and 00111011 is 59, so the difference is 91, which is 01011011 (64 + 16 + 8 + 2 + 1). Checking in decimal is the quickest way to catch borrow mistakes.

481. **D** — n-bit two's complement covers -2^(n-1) to 2^(n-1) - 1. Seven bits reach only -64 to 63, while 8 bits reach -128 to 127, which contains -100 to 100. 201 counts the values, not the bits.

482. **C** — Split into pairs: FF = 255, 80 = 8 × 16 = 128, 00 = 0. Full red plus half green gives orange. Reading 80 as decimal 80 forgets that the pairs are hexadecimal.

483. **C** — 0x40400000 is 0 10000000 10000000000000000000000. Sign 0 means positive, the exponent 128 - 127 = 1 gives ×2, and the fraction .1 in binary gives a significand of 1.5. So the value is 1.5 × 2 = 3.0.

484. **A** — Gray = b XOR (b >> 1) = 0110 XOR 0011 = 0101. 0011 is only the shifted value, and 1001 is the bitwise inverse of 0110.

485. **D** — An arithmetic right shift by 1 divides by 2 and rounds toward negative infinity. 7 >> 1 is 3, and -7 >> 1 is -4, not -3, because -3.5 rounds down. Treating it as truncation toward zero gives the -3 trap.

486. **B** — One parent per node and a single root is exactly the definition of a tree, so reporting lines and 'who is above whom' questions become simple tree walks.

487. **C** — Hex 45 is 4 × 16 + 5 = 69. The other three all equal 45: 32 + 8 + 4 + 1, 2 × 16 + 13, and 5 × 8 + 5. The trap is reading the hex digits as if they were decimal.

488. **C** — Stored as a number, 03001234567 loses its leading zero, and it is also larger than a 32-bit integer can hold. Phone numbers are identifiers, not quantities, so a string models them correctly.

489. **C** — Each flip changes the number of 1s by one, so two flips change it by two and the parity stays even. A single parity bit detects any odd number of flipped bits but misses every even number.

490. **B** — Each setting needs one bit, so 8 settings fit exactly in one byte. Bitwise AND, OR and XOR with masks read and change individual settings. The other choices use 32 to 50 times more memory.

491. **B** — len(s) counts characters, so 'café' has 4. In UTF-8, c, a and f are one byte each, but é (U+00E9) needs 2 bytes, so the encoded length is 5. Code that assumes one byte per character breaks on such text.

492. **C** — C8 = 11001000 = 200 unsigned. In two's complement the top bit is worth -128, so the value is -128 + 64 + 8 = -56 (or 200 - 256). The bits never say how to read them; the type does.

493. **A** — Strings sort character by character: '1' comes before '9', and '10' is a prefix of '100', so the order is '10', '100', '9'. To sort numerically, store the codes as integers or sort with key=int.

494. **D** — A fraction is exact in binary only if its denominator, in lowest terms, is a power of 2. 0.75 = 3/4, 0.5 = 1/2 and 2.5 = 5/2 qualify. 0.2 = 1/5 has a factor of 5, so its binary expansion repeats forever and is rounded.

495. **A** — A matrix needs 10^6 × 10^6 = 10^12 cells regardless of edges. With about 50 million friendships, an adjacency list stores roughly 10^8 entries, a tiny fraction. Matrices suit dense graphs or very fast edge-existence checks.

496. **B** — 0111 1111 1111 1111 + 1 = 1000 0000 0000 0000, and in 16-bit two's complement that pattern is -32768. The value wraps from the top of the range to the bottom, which is how integer-overflow bugs appear.

497. **A** — Near 10^16, neighbouring doubles are 2 apart, so 1e16 + 1 rounds back to 1e16 (ties go to the even significand). Subtracting 1e16 then gives 0.0. Large floats lose small additions entirely.

498. **D** — The range is 125 °C, which is 250 steps, so there are 251 possible values including both ends. 2^7 = 128 is too few and 2^8 = 256 is enough, so 8 bits are needed.

499. **D** — Little-endian puts the low byte first, so the value is 0x1234 = 1 × 4096 + 2 × 256 + 3 × 16 + 4 = 4660. Reading the bytes in file order gives 0x3412 = 13330, the classic endianness bug.

500. **C** — 8-bit two's complement holds -128 to 127. 64 + 64 = 128 is out of range and wraps to 10000000, which reads as -128. The others give 127, -128 and -1, which all fit.

501. **C** — In an arithmetic sequence you add the same common difference d each time, for example 4, 9, 14, 19 (d = 5). A constant ratio defines a geometric sequence.

502. **D** — 3, 6, 12, 24 multiplies by 2 each time, a constant ratio. 3, 6, 9, 12 adds 3 (arithmetic), 1, 4, 9, 16 are squares and 1, 1, 2, 3 is Fibonacci.

503. **A** — After the starting values 0 and 1, every term is F(n) = F(n-1) + F(n-2): 1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5 and so on.

504. **B** — The first term has no differences added, the second has one, so the nth has n - 1: a + (n - 1)d. a × d^(n - 1) is the shape of a geometric nth term, and a + nd is off by one.

505. **C** — Constant first differences mean a linear rule; constant second differences mean a quadratic rule, and the second difference equals 2a. Exponential sequences never settle to constant differences at any level.

506. **B** — Going from specific examples to a general rule is inductive reasoning. Its conclusions are likely but not certain, so the rule still needs testing or proof. Deduction goes from general rules to specific conclusions.

507. **D** — A recurrence defines each term from earlier terms and needs a starting value. The other three are closed forms: you can compute any term directly from n.

508. **B** — A short sequence can be extended by many rules. Doubling gives 8, and differences of +1, +2, +3 give 7. Look for more terms or context, and pick the simplest rule consistent with everything given.

509. **B** — Pair the first and last terms (1 + n), the second and second-last (2 + n - 1) and so on: n/2 pairs each worth n + 1. n(n - 1)/2 is the sum up to n - 1, which is the handshake count.

510. **C** — S(n) = a(r^n - 1)/(r - 1). For 1 + 2 + 4 + 8 (a = 1, r = 2, n = 4) it gives (16 - 1)/1 = 15. a × r^(n - 1) is just the nth term, not the sum.

511. **B** — The differences are 2, 3, 4, 5, so the nth triangular number is 1 + 2 + ... + n = n(n + 1)/2. The next term is 15 + 6 = 21.

512. **B** — When terms jump up and down, check the odd and even positions separately. Here the odd positions are 2, 4, 6 (+2) and the even positions are 50, 45, 40 (-5).

513. **B** — Induction needs a base case and an inductive step (k → k + 1). Together they reach every positive integer, like dominoes. Checking many cases is only evidence, and assuming what you want to prove is circular.

514. **A** — The last digits are 7, 9, 3, 1, then 7 again (because 7 × 1 = 7). So the cycle length is 4, and the last digit of 7^n depends only on n mod 4.

515. **C** — With t(n) = 4n - 1 you get t(1000) = 3999 in one step. The recurrence t(n) = t(n - 1) + 4 would need 999 steps from t(1). Generalising to a closed form is the payoff of pattern recognition.

516. **C** — Generalisation follows pattern recognition: once you see what several cases share, you write one parameterised rule or algorithm, for example a function sum(n) instead of separate code for n = 5 and n = 10.

517. **D** — The common difference is +4, so the next term is 15 + 4 = 19.

518. **A** — Each term is 3 times the previous one, so the next is 54 × 3 = 162. Adding the last difference (36) would give 90, which ignores the ratio.

519. **D** — These are the squares 1^2, 2^2, 3^2, 4^2, 5^2, so the next is 6^2 = 36. The differences 3, 5, 7, 9, 11 confirm it: 25 + 11 = 36.

520. **D** — The differences are 1, 2, 3, 4, 5, so the next difference is 6 and the next term is 17 + 6 = 23. Treating it as Fibonacci (12 + 17) gives 29, which the earlier terms do not support.

521. **C** — a = 7 and d = 5, so t(50) = 7 + 49 × 5 = 7 + 245 = 252. Using 50 × 5 instead of 49 × 5 gives 257, the usual off-by-one slip.

522. **A** — From the third term on, each term is the sum of the two before it: 2 + 5 = 7, 5 + 7 = 12, 7 + 12 = 19, so the next is 12 + 19 = 31.

523. **B** — The differences 3, 5, 7, 9 rise by 2, so the next difference is 11 and the next term is 27 + 11 = 38. The rule is $n^2 + 2$.

524. **A** — Each term is double the previous plus 1: 2 × 2 + 1 = 5, 5 × 2 + 1 = 11, and so on. So 47 × 2 + 1 = 95. Simply doubling gives 94.

525. **A** — Convert to positions: 1, 3, 6, 10, 15. The gaps are 2, 3, 4, 5, so the next gap is 6 and the next position is 21, which is U.

526. **D** — Each new square shares one side, so it adds 3 sticks: the rule is 3n + 1. For n = 25 that is 76. Treating squares as separate (4n = 100) ignores the shared sides.

527. **C** — Two sequences are interleaved. Odd positions: 1, 3, 5, 7 (+2). Even positions: 10, 20, 30 (+10). The next term is in an even position, so it is 40. 9 would be next only for the odd-position sequence.

528. **C** — The last digits of powers of 7 cycle 7, 9, 3, 1 with length 4. 2026 mod 4 = 2, so 7^2026 ends like 7^2 = 49, in 9.

529. **C** — The 20th term is 4 + 19 × 3 = 61. The sum is n/2 × (first + last) = 10 × (4 + 61) = 650. 1300 forgets to halve.

530. **A** — Each person shakes 11 hands, giving 12 × 11 = 132, but that counts every handshake twice, so the total is 132 / 2 = 66. In general n(n - 1)/2.

531. **A** — The terms follow n(n + 1): 1×2, 2×3, 3×4, 4×5, 5×6, 6×7, 7×8, i.e. 2, 6, 12, 20, 30, 42, 56. The sixth term should be 42, so 40 is wrong. The differences 4, 6, 8, 10, 12, 14 confirm it.

532. **D** — Row i has 2i - 1 stars: 1, 3, 5, ..., 19. The sum of the first n odd numbers is n^2, so 10 rows give 100 stars. 19 is only the last row.

533. **A** — The doubling pattern breaks: the count is C(n, 4) + C(n, 2) + 1, which gives 15 + 15 + 1 = 31 for n = 6. This is the standard warning that a pattern seen in a few cases is a guess until it is proved.

534. **D** — 3n + 2 gives 5, 8, 11, 14, matching every term. A formula must be checked against all given terms, not only the first one or two.

535. **B** — The terms are 2, 6, 12, 20, 30, ... Adding 2n each step builds 2(1 + 2 + ... + n) = n(n + 1). Check: 4 × 5 = 20.

536. **D** — Last digits of powers of 3 cycle 3, 9, 7, 1, and each full cycle sums to 20, which ends in 0. 100 terms are exactly 25 full cycles, so the total ends in 0.

537. **C** — 1681 = 41 × 41, so one counterexample disproves the general claim. Inductive guesses from examples need a proof; even 40 confirming cases were not enough.

538. **A** — Each count is one less than a power of two: 2^n - 1. This matches the recurrence M(n) = 2M(n - 1) + 1. For 10 discs: 2^10 - 1 = 1023.

539. **D** — The ratios are ×2, ×3, ×4, ×5, so the next is ×6: 120 × 6 = 720. These are the factorials 1!, 2!, ..., 6!.

540. **B** — Count by size: 1×1 gives 16, 2×2 gives 9, 3×3 gives 4 and 4×4 gives 1, so 30 in total. In general an n × n grid has $1^2 + 2^2 + ... + n^2$ squares.

541. **A** — Cube numbers are n^3: $1^3 = 1$, $2^3 = 8$, $3^3 = 27$, $4^3 = 64$. 1, 4, 9, 16 are squares, 3, 6, 9, 12 are multiples of 3, and 1, 3, 9, 27 are powers of 3.

542. **A** — Each term divided by the one before gives 27/81 = 1/3. The sequence shrinks, so the ratio is a fraction, not 3. -54 is the first difference, which is not constant here.

543. **A** — Each term is the previous one times -2: 3 × -2 = -6, -6 × -2 = 12. The negative ratio makes the signs alternate. The differences (-9, 18, -36) are not constant, so it is not arithmetic.

544. **D** — Deduction runs from a general, proven rule to a particular conclusion, so the answer 1275 is certain. Induction runs the other way, from cases to a conjectured rule.

545. **D** — A universal claim says 'for every n', so a single counterexample disproves it. Confirming examples can never prove such a claim on their own, but one failure always refutes it.

546. **A** — The level of difference that becomes constant equals the degree of the polynomial: constant first differences mean linear, second mean quadratic, third mean cubic. Geometric sequences never reach constant differences.

547. **A** — 1 = 1, 1 + 3 = 4, 1 + 3 + 5 = 9: the sums are perfect squares. Each odd number adds an L-shaped border that grows a k × k square into a (k + 1) × (k + 1) square. n(n + 1)/2 is the sum of all numbers 1 to n.

548. **A** — The sum of squares is n(n + 1)(2n + 1)/6, so for n = 3 it gives 3 × 4 × 7 / 6 = 14 = 1 + 4 + 9. $[n(n + 1)/2]^2$ is the sum of cubes, and $n^3/3$ is only an approximation for large n.

549. **B** — If $|r| < 1$ the terms shrink toward 0 and the partial sums settle at $a / (1 - r)$; for example $1 + 1/2 + 1/4 + \ldots = 2$. If $|r| \geq 1$ the terms do not shrink, so the series has no finite sum.

550. **A** — Pattern recognition spots similarities between problems or parts of a problem. Decomposition comes before it (breaking the problem into parts), and generalisation and abstraction follow it.

551. **A** — P grows linearly (adds d each step) while Q grows exponentially (multiplies by r each step). Exponential growth always overtakes linear growth in the end, which is why compound interest beats simple interest.

552. **C** — Ratios of consecutive terms, such as $13/8 = 1.625$ and $21/13 \approx 1.615$, settle at $\varphi = (1 + \sqrt{5})/2 \approx 1.618$. The terms themselves grow without limit, but their ratio does not.

553. **B** — Rows sum to 1, 2, 4, 8, 16: row 4 is $1 + 4 + 6 + 4 + 1 = 16$. Each number feeds two numbers in the next row, so each row's sum doubles.

554. **C** — Each term needs the two before it, so t(1) and t(2) must be given. With 1, 1 it gives Fibonacci; with 2, 1 it gives the Lucas numbers. A recurrence of order k needs k starting values.

555. **D** — $t(n) = a + (n - 1)d = dn + (a - d)$, which is a straight line in n with slope d. Only the special case $d = 0$ is constant.

556. **A** — With 5 coefficients you can match any 5 values, so a perfect fit says nothing. A rule earns trust when it predicts terms it was not built from, or when it is derived from the problem's structure.

557. **C** — $10 \equiv 1 \pmod 9$, so $10^k \equiv 1$ and a number such as $351 = 3 \times 100 + 5 \times 10 + 1$ has the same remainder as $3 + 5 + 1$. That turns an observed pattern into a proven general rule.

558. **B** — Check the ten possible units digits: $2^5 = 32$, $3^5 = 243$, $7^5 = 16807$, and so on; each ends in its own digit. Units-digit cycles have length 1, 2 or 4, and 4 divides $5 - 1$, so the fifth power always returns to the start.

559. **D** — Each entry is the sum of the two above it, which is the same rule as $C(n, k) = C(n - 1, k - 1) + C(n - 1, k)$. So row 4 is 1, 4, 6, 4, 1 = C(4,0) to C(4,4).

560. **D** — Computing t(n) calls t(n - 1) and t(n - 2), and t(n - 1) calls t(n - 2) again, and so on. The call count grows like the Fibonacci numbers themselves. Storing results (memoisation) or a simple loop makes it linear.

561. **D** — The difference between terms is always 0 and the ratio is always 1, so it meets both definitions. This edge case is a common trap in 'which type is it' questions.

562. **C** — $(n + 1)^2 - n^2 = 2n + 1$, which is why the gaps between squares are the odd numbers 3, 5, 7, 9. Generalising the gap explains the pattern instead of only describing it.

563. **D** — The terms shrink toward 0, but not fast enough: grouping them ($1/3 + 1/4 > 1/2$, $1/5$ to $1/8 > 1/2$, ...) adds at least 1/2 again and again. Terms going to 0 does not guarantee a finite sum. $1 + 1/2 + 1/4 + \ldots$ is the series that approaches 2.

564. **C** — Each term is half the previous one, so the ratio is 1/2. The differences (-1/2, -1/4, -1/8) are not constant, so it is not arithmetic.

565. **A** — Each term is half the one before (ratio 1/2), so $8 / 2 = 4$. Subtracting 8 would treat it as arithmetic, but the differences 32, 16, 8 are not constant.

566. **C** — $125 = 5^3$ and $216 = 6^3$, so the next cube is $7^3 = 343$. Adding the same difference (91) gives 307, which ignores the cube pattern.

567. **B** — These are consecutive prime numbers. After 23, 24 to 28 are all composite ($25 = 5^2$, $27 = 3^3$), so the next prime is 29.

568. **C** — The common difference is -7, so $29 - 7 = 22$. 15 subtracts 14, skipping a term.

569. **B** — The first letter moves forward one step (A, B, C, D) and the second moves back one step from Z (Z, Y, X, W). Each pair's positions add to 27.

570. **D** — The sum of the first n odd numbers is n^2, so $15^2 = 225$. 120 is $1 + 2 + \ldots + 15$, and 240 is the sum of the first 15 even numbers.

571. **D** — It adds $1^2 + 2^2 + \ldots + 10^2$. By the formula $n(n + 1)(2n + 1)/6$ that is $10 \times 11 \times 21 / 6 = 385$. 55 is the plain sum 1 to 10, and 3025 is the sum of cubes.

572. **C** — From $t(n) = 7 + 4(n - 1) = 203$, we get $n - 1 = 49$, so $n = 50$. 49 counts the gaps, not the terms; this off-by-one is the usual trap.

573. **D** — The differences are 1, 4, 9, 16, the square numbers, so the next difference is 25 and $31 + 25 = 56$. 47 adds 16 again, and 62 doubles the last term.

574. **B** — As positions these are 26, 24, 21, 17, 12. The gaps grow: -2, -3, -4, -5, so the next gap is -6 and 12 - 6 = 6, which is F.

575. **A** — Each term is 3 times the previous one minus 1: 2 × 3 - 1 = 5, 5 × 3 - 1 = 14, and 122 × 3 - 1 = 365. 366 forgets the - 1.

576. **B** — The differences must rise steadily: 7, 9, 11, 13. So the missing term is 12 + 9 = 21, and 21 + 11 = 32 confirms it. The rule is $n^2 + 4n$.

577. **A** — This is geometric with a = 3 and r = 2. The sum of 10 terms is 3(2^10 - 1) / (2 - 1) = 3 × 1023 = 3069. 1536 is the 10th term itself, and 3072 forgets the - 1.

578. **D** — Each vertex joins to n - 3 others by a diagonal, and each diagonal is counted twice, giving n(n - 3)/2. For n = 12 that is 12 × 9 / 2 = 54. 66 counts every pair, including the 12 sides, and 108 forgets to halve.

579. **B** — The total is 1 + 2 + ... + 20 = 20 × 21 / 2 = 210. 420 forgets to halve, and 400 treats it as a full 20 × 20 square.

580. **D** — The last digits of 2^1, 2^2, 2^3, 2^4 are 2, 4, 8, 6, and then the cycle repeats every 4. 100 is a multiple of 4, so 2^100 ends like 2^4, in 6.

581. **A** — Days repeat every 7, so only 100 mod 7 matters. 100 = 14 × 7 + 2, so the answer is 2 days after Monday: Wednesday.

582. **C** — range(1, 6) gives n = 1 to 5, and $n^2 - n$ gives 0, 2, 6, 12, 20 (the pronic numbers n(n - 1)). The list with 30 would need n = 6, which range excludes.

583. **C** — f(n) adds the nth odd number 2n - 1 to f(n - 1), starting from 1. So f(n) is the sum of the first n odd numbers, which is n^2, and f(12) = 144. 169 would be f(13).

584. **C** — Read 111221 aloud: three 1s, two 2s, one 1, which gives 31 22 11 = 312211. 13112221 is the term after that. The pattern is about describing digits, not arithmetic.

585. **C** — Trace t: start 1; n = 2 gives 4; n = 3 gives 11; n = 4 gives 26; n = 5 gives 57; n = 6 gives 120. range(2, 7) stops at 6, so 57 is one step short, and 247 is one step too many.

586. **B** — $n^2 - n + 2$ gives 2, 4, 8, 14. The differences 2, 4, 6 rise by 2, which marks a quadratic. 2^n fits the first three terms but gives 16, which shows why you must test every term.

587. **B** — Every term is 3 times the previous one, so after 81 comes 243, and 243 × 3 = 729 confirms it. 242 is one short.

588. **D** — 121, 144, 169 and 196 are 11^2, 12^2, 13^2 and 14^2. 186 is not a perfect square ($13^2 = 169$ and $14^2 = 196$ lie either side of it).

589. **C** — The outputs 3, 5, 7 rise by 2 for each step in n, so the rule is linear with slope 2: 2n + 1 gives 3, 5, 7. Replacing special cases with one parameterised rule is generalisation in code.

590. **A** — The checks suggest a pattern, but the proof comes from the structure: of any two consecutive integers one is even, so their product is even. That argument covers every n, which no finite list of checks can.

591. **B** — The left end is either one vertical domino (leaving 2 × (n - 1)) or two stacked horizontal dominoes (leaving 2 × (n - 2)). So T(n) = T(n - 1) + T(n - 2): 1, 2, 3, 5, 8, 13, 21. The structure gives a Fibonacci rule.

592. **C** — Calls satisfy C(n) = C(n - 1) + C(n - 2) + 1 with C(0) = C(1) = 1, giving 1, 1, 3, 5, 9, 15. So f(5) makes 15 calls, and the count grows exponentially with n.

593. **C** — Every shortest path is 3 rights and 3 downs in some order, so the count is the number of ways to choose where the 3 rights go among 6 moves: C(6, 3) = 20. Generalising, an m × n grid has C(m + n, m) paths.

594. **D** — Plan B pays 10 × 2^(n - 1). In month 11 it pays 10,240 against A's 11,000; in month 12 it pays 20,480 against 12,000. Exponential growth starts slowly but overtakes linear growth.

595. **D** — 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1 is 8 steps. The rule always seems to reach 1, but nobody has proved it for all starting values, a reminder that observed patterns are not proofs.

596. **B** — $x^2 + x + 1$ gives 1, 3, 7, 13. The differences 2, 4, 6 rise steadily, which signals a quadratic. 2^(x+1) - 1 matches the first three rows and fails only at x = 3, so stopping early would pick it.

597. **A** — 2 + 4 + ... + 2n = 2(1 + 2 + ... + n) = 2 × n(n + 1)/2 = n(n + 1). n = 1 already shows the student's error: the sum is 2, not 1. n^2 is the sum of the first n odd numbers.

598. **B** — Row 4 (the widest) needs 0 leading spaces, and each row above it needs one more, so row i needs h - i spaces: 3, 2, 1, 0. h - i + 1 shifts every row one space right, so the base would not start at the margin.

599. **B** — Powers of 2 modulo 7 cycle with length 3: 2, 4, 1, 2, 4, 1, ... 50 = 3 × 16 + 2, so 2^50 has the same remainder as 2^2, which is 4.

600. **B** — The nth line crosses all n - 1 earlier lines and so splits n existing regions, adding n. Regions = 1 + (1 + 2 + ... + n) = 1 + n(n + 1)/2, so 10 lines give 1 + 55 = 56. Doubling (2, 4, 8, ...) breaks at 3 lines.

601. **C** — A percentage is a number out of 100. 3/4 = 0.75, and 0.75 × 100 = 75%. Multiplying the decimal by 100 is the standard conversion.

602. **D** — Divide both terms by their highest common factor, 6: 12 ÷ 6 = 2 and 18 ÷ 6 = 3, giving 2:3. 6:9 is equivalent but not fully simplified, and 3:2 reverses the order.

603. **D** — A probability always lies between 0 (impossible) and 1 (certain), inclusive. 1.2 is greater than 1, so it cannot be a probability. 0, 0.5 and 1 are all valid.

604. **A** — The first position has 4 choices, the next 3, then 2, then 1, so there are 4! = 4 × 3 × 2 × 1 = 24 orders. 16 is 4^2, which wrongly allows a book to be used twice.

605. **C** — Mutually exclusive events cannot happen together, so P(A and B) = 0 and the addition rule reduces to P(A or B) = P(A) + P(B). The product P(A) × P(B) is the joint probability of independent events, a different idea.

606. **B** — A permutation counts ordered arrangements and a combination counts unordered selections. For the same n and r, nPr = nCr × r!, so the permutation count is never smaller than the combination count.

607. **C** — The difference between two percentages is measured in percentage points: 25 − 20 = 5 points. The relative change is 5 ÷ 20 = 0.25, a 25% increase. Mixing up points and percent is a classic trap.

608. **C** — Independence means one event does not change the probability of the other, so the multiplication rule gives P(A and B) = P(A) × P(B). For two fair coins, P(both heads) = 1/2 × 1/2 = 1/4.

609. **A** — 'At least one' covers many overlapping cases (exactly one, exactly two and so on). Its complement, 'none', is a single case that is usually easy to calculate, and P(event) + P(not event) = 1 always holds.

610. **C** — Direct proportion means y = kx for a constant k. Doubling x gives k(2x) = 2y, so y doubles. Halving would describe inverse proportion.

611. **B** — The median is the middle value once the data is sorted, so one extreme value moves it by at most one position. The mean (sum divided by count) is pulled towards the outlier, and the midpoint of the extremes depends directly on it.

612. **D** — The second change uses a new base. 100 rises to 110, and 10% of 110 is 11, so the price falls to 99. Overall the multiplier is 1.1 × 0.9 = 0.99, a net 1% fall.

613. **B** — Choosing 3 items to take is the same as choosing the 7 items to leave, so C(n, r) = C(n, n − r). Both C(10, 3) and C(10, 7) equal 120. P(10, 3) = 720 counts ordered selections.

614. **A** — When one choice can be made in m ways and a second choice in n ways, the pair can be made in m × n ways. Here that is 3 × 4 = 12 outfits. Addition applies when you pick one item from either group, not one from each.

615. **B** — The total work (workers × days) is fixed, so workers and days are inversely proportional. Doubling the workers halves the days, assuming every worker keeps the same rate.

616. **A** — Adding probabilities only works for mutually exclusive events, and sixes on different rolls can occur together. The correct value is 1 − (5/6)^4 = 671/1296 ≈ 0.518, not 2/3. With 7 rolls the adding method would even give more than 1.

617. **D** — 15% of 240 = 0.15 × 240 = 36. A quick check: 10% is 24 and 5% is 12, and 24 + 12 = 36.

618. **A** — Percentage change = (new − old) ÷ old × 100 = (100 − 80) ÷ 80 × 100 = 25%. Dividing by the new value (100) gives the wrong answer of 20%.

619. **C** — The ratio has 3 + 5 = 8 parts, so one part is 1,200 ÷ 8 = 150. The larger share is 5 × 150 = 750 and the smaller is 3 × 150 = 450.

620. **B** — There are 6 × 6 = 36 equally likely outcomes. Six give a total of 7: (1,6), (2,5), (3,4), (4,3), (5,2), (6,1). So P = 6/36 = 1/6. 1/11 wrongly treats the 11 possible totals as equally likely.

621. **D** — Order does not matter in a committee, so use combinations: C(7, 3) = 7! ÷ (3! × 4!) = 35. 210 is P(7, 3), which counts ordered selections.

622. **B** — LEVEL has 5 letters, with L appearing twice and E appearing twice. Arrangements = 5! ÷ (2! × 2!) = 120 ÷ 4 = 30. 120 ignores the repeats and 60 divides by only one of them.

623. **C** — The price is multiplied by 0.8 and then by 0.9, giving 0.72 of the original. The overall discount is 1 − 0.72 = 0.28, or 28%. Adding the two discounts to get 30% ignores the changed base.

624. **A** — Average speed = total distance ÷ total time = 120 ÷ (60/30 + 60/60) = 120 ÷ 3 = 40 km/h. Averaging the two speeds to get 45 is wrong because more time is spent at the slower speed.

625. **B** — P(first red) = 5/8 and, with one red gone, P(second red) = 4/7. Multiply: 5/8 × 4/7 = 20/56 = 5/14. 25/64 would be right only with replacement, and 15/28 is the chance of one red and one blue.

626. **C** — Use the complement. P(no heads) = (1/2)^3 = 1/8, so P(at least one head) = 1 − 1/8 = 7/8. 3/8 is the probability of exactly one head.

627. **D** — The amount of acid does not change: 30% of 20 = 6 litres. The new volume is 30 litres, so the acid share is 6 ÷ 30 = 20%.

628. **B** — New price = original × 1.25, so original = 150,000 ÷ 1.25 = 120,000. Taking 25% off 150,000 (giving 112,500) is wrong because the 25% was based on the smaller original price.

629. **B** — Simple interest = principal × rate × time = 5,000 × 0.08 × 3 = 1,200. 1,298.56 is the compound-interest figure, and 6,200 is the final amount, not the interest.

630. **D** — Order matters and digits cannot repeat, so count 10 × 9 × 8 × 7 = 5,040, which is P(10, 4). 10,000 allows repeats, and 210 is C(10, 4), which ignores order.

631. **A** — Each change is measured against its own starting value. Rise: 10 ÷ 40 = 25%. Fall: 10 ÷ 50 = 20%. The same absolute change gives different percentages because the bases differ.

632. **C** — Use a weighted average: (30 × 60 + 20 × 70) ÷ 50 = (1,800 + 1,400) ÷ 50 = 3,200 ÷ 50 = 64. The answer sits closer to 60 because section A is larger.

633. **C** — Per 1,000 students: 10 have the condition and 9 of them test positive. Of the 990 healthy students, 99 test positive. So P(condition | positive) = 9 ÷ (9 + 99) = 1/12 ≈ 8.3%. The low base rate outweighs the test's accuracy.

634. **D** — Use the complement. P(all different) = (7 × 6 × 5) ÷ 7^3 = 210/343 = 30/49. So P(at least two share) = 1 − 30/49 = 19/49 ≈ 0.39.

635. **A** — Make B the same in both ratios. The LCM of 3 and 4 is 12, so A:B = 8:12 and B:C = 12:15. Combined, A:B:C = 8:12:15.

636. **A** — Each person shakes 9 hands, giving 10 × 9 = 90, but every handshake involves two people, so each was counted twice. The correct count is 90 ÷ 2 = 45 = C(10, 2).

637. **D** — The two 'six' events can happen together (6 and 6), so adding counts that outcome twice. Using the complement: 1 − $(5/6)^2$ = 1 − 25/36 = 11/36. Equivalently, 1/6 + 1/6 − 1/36 = 11/36.

638. **B** — Total work = 6 × 10 = 60 worker-days. In 4 days, 6 × 4 = 24 are done, leaving 36. Four workers need 36 ÷ 4 = 9 more days, so the job takes 4 + 9 = 13 days.

639. **D** — Growth compounds: 50,000 × 1.1 × 1.1 = 60,500. Year two's 10% is taken on 55,000, which adds 5,500 rather than 5,000.

640. **A** — All teams: C(9, 2) = 36. Teams with no woman: C(5, 2) = 10. So teams with at least one woman = 36 − 10 = 26. Direct check: one woman 4 × 5 = 20, two women C(4, 2) = 6, total 26.

641. **B** — 0.125 = 125/1000. Dividing top and bottom by 125 gives 1/8. A quick check: 1 ÷ 8 = 0.125.

642. **B** — An event and its complement cover every outcome, so P(not A) = 1 − P(A) = 1 − 0.35 = 0.65.

643. **B** — 0! is defined as 1. There is exactly one way to arrange zero objects (do nothing), and the definition keeps formulas such as n!/(n − r)! correct when r = n.

644. **D** — Expected value = (1 + 2 + 3 + 4 + 5 + 6) / 6 = 21 / 6 = 3.5. It need not be a value the die can actually show; it is the long-run average.

645. **A** — The mode is the most frequent value. The median is the middle value after sorting, the mean is the total divided by the count, and the range is a measure of spread, not an average.

646. **A** — Inverse proportion means x × y stays constant. If x is multiplied by 3, y must be divided by 3 to keep the product the same.

647. **A** — Range = maximum − minimum. It is a simple measure of spread. The other options describe the mode, the median and the mean.

648. **D** — Each coin has 2 outcomes, so by the multiplication principle there are 2 × 2 = 4: HH, HT, TH and TT. Writing only 3 (two heads, one head, no heads) merges HT and TH, which are different outcomes.

649. **B** — The ratio splits the class into 2 + 3 = 5 equal parts, and boys hold 2 of them, so boys are 2/5 of the class. 2/3 compares boys with girls, not boys with the whole class.

650. **C** — Knowing B happened shrinks the sample space to B, so we divide the probability of both events by P(B). Dividing by P(A) gives P(B | A) instead.

651. **C** — This is the general addition rule. Adding P(A) and P(B) counts the overlap twice, so it is subtracted once. P(A) + P(B) alone is only correct when the events are mutually exclusive.

652. **B** — Odds of 1:3 mean 1 favourable outcome for every 3 unfavourable ones, so 4 equally likely outcomes in total. The probability is 1/(1 + 3) = 1/4.

653. **B** — Fix one person's seat to remove the n equivalent rotations; the other n − 1 people can then be arranged in (n − 1)! ways. Equivalently n!/n = (n − 1)!.

654. **B** — Simple interest is always calculated on the original principal. Compound interest adds each period's interest to the balance, so the base grows and interest is earned on interest.

655. **B** — Adding 5 to every value shifts the whole data set up by 5, so the mean rises by 5. The largest and smallest values both rise by 5, so their difference (the range) does not change.

656. **A** — For each of the n elements there are 2 choices, in or out of the subset. By the multiplication principle that gives 2 × 2 × … × 2 = 2^n subsets.

657. **A** — Choose the first item in n ways, the second in n − 1 ways, and so on for r positions: n(n − 1)…(n − r + 1) = n!/(n − r)!. Dividing again by r! gives combinations, where order does not matter.

658. **B** — Without replacement, the bag has one fewer ball, and one fewer of the colour drawn, for the second draw. So P(second | first) differs from P(second), which is the definition of dependence.

659. **A** — Percentage change is always measured against the starting value, so the difference is divided by the old amount. Dividing by the new amount is a common mistake that gives the wrong base.

660. **B** — Complementary events cannot happen together (mutually exclusive) and between them cover every outcome (exhaustive), so their probabilities add to exactly 1.

661. **D** — Mutually exclusive means P(A and B) = 0. Independence needs P(A and B) = P(A) × P(B), which is positive here. The two conditions conflict, so such events are dependent: if A happens, B certainly does not.

662. **D** — n!/(n − r)! counts ordered selections. Any one group of r items can be ordered in r! ways, so each group was counted r! times. Dividing by r! counts each group once.

663. **D** — Adding the three sets counts pairwise overlaps twice, so they are subtracted. But that subtracts the triple overlap three times after adding it three times, leaving it at zero, so it is added back once.

664. **C** — Independence means P(A and B) = P(A) P(B). Dividing by P(B) gives P(A | B) = P(A): learning that B happened does not change the chance of A. P(A and B) = 0 describes mutually exclusive events.

665. **B** — 12.5% = 1/8, so 12.5% of 640 = 640 ÷ 8 = 80. Spotting common fraction equivalents (12.5% = 1/8, 25% = 1/4) speeds up such questions.

666. **A** — The discount is 15% of 500 = 75, so the price is 500 − 75 = 425. Equivalently, 500 × 0.85 = 425. 75 is only the discount, not the price.

667. **A** — The total is 4 + 8 + 15 + 16 + 23 + 42 = 108, and there are 6 values, so the mean is 108 / 6 = 18. 15.5 is the median.

668. **A** — Sorted: 1, 3, 5, 7, 9, 11. With an even count, the median is the mean of the two middle values: (5 + 7) / 2 = 6. Forgetting to sort gives 9 (the middle of the unsorted list).

669. **A** — The multiples of 4 are 4, 8, 12, 16 and 20: 5 numbers out of 20 equally likely ones. 5/20 = 1/4.

670. **D** — 4 cm × 50,000 = 200,000 cm. Since 100,000 cm = 1 km, that is 2 km.

671. **D** — 5! / 3! = (5 × 4 × 3!) / 3! = 5 × 4 = 20. It equals P(5, 2), the number of ordered pairs chosen from 5 items.

672. **C** — BANANA has 6 letters with A repeated 3 times and N repeated 2 times, so the count is 6! / (3! × 2!) = 720 / 12 = 60.

673. **D** — For circular arrangements, fix one person and arrange the rest: (5 − 1)! = 4! = 24. 120 = 5! counts each seating 5 times, once per rotation.

674. **A** — Amount = 10,000 × 1.1^2 = 12,100, so the interest is 12,100 − 10,000 = 2,100. Simple interest would give only 2,000; the extra 100 is interest on the first year's interest.

675. **C** — The posts are different, so order matters: 8 choices for president, then 7 for secretary, giving 8 × 7 = 56 = P(8, 2). 28 = C(8, 2) would treat the two posts as the same.

676. **A** — P(king) = 4/52 and P(heart) = 13/52, but the king of hearts is in both, so P = 4/52 + 13/52 − 1/52 = 16/52 = 4/13. 17/52 counts the king of hearts twice.

677. **D** — The difference is 7 − 4 = 3 parts = 21, so 1 part = 7. The total is 4 + 7 = 11 parts = 77. Ali has 28 and Babar 49.

678. **D** — Rates add: A does 1/6 and B does 1/3 of the job per day, so together 1/6 + 2/6 = 1/2 per day. The job takes 1 ÷ 1/2 = 2 days. Averaging the times (4.5) is a common error.

679. **D** — 1 km/h = 1000 m / 3600 s = 5/18 m/s, so 72 × 5/18 = 20 m/s. 259.2 comes from multiplying by 3.6 instead of dividing.

680. **C** — Cost = 1,150 / 1.15 = 1,000. A 25% profit needs 1,000 × 1.25 = 1,250. Adding 10% to 1,150 (1,265), another 15% (1,322.5) or 25% to 1,150 (1,437.5) all use the selling price as the base by mistake.

681. **D** — The hundreds digit can be 1–9 (9 ways), the tens digit any of the other 9 digits including 0, and the units digit any of the remaining 8: 9 × 9 × 8 = 648. 720 = 10 × 9 × 8 wrongly allows a leading zero.

682. **C** — There are 2^4 = 16 equally likely sequences, and C(4, 2) = 6 of them have exactly 2 heads. 6/16 = 3/8. 1/16 is the chance of one specific sequence such as HHTT.

683. **A** — Choose 2 of the 4 women in C(4, 2) = 6 ways and 1 of the 5 men in 5 ways: 6 × 5 = 30. 84 = C(9, 3) counts all committees with no restriction.

684. **C** — The solution contains 2 L of acid. Adding x L of pure acid gives (2 + x)/(20 + x) = 0.25, so 2 + x = 5 + 0.25x and x = 4. Check: 6/24 = 25%.

685. **A** — Each toss of a fair coin is independent; the coin has no memory. P(tails) on the 6th toss is 1/2. Believing a result is 'due' is the gambler's fallacy.

686. **D** — Pass rates must be compared as percentages of each class size: 36/40 = 90% and 45/60 = 75%. Class B has more passes only because it is larger.

687. **D** — Let cost = 100, so price = 125. The cost is 25 below the price, and 25/125 = 20%. The base changes from cost to price, so the percentage is not 25% again.

688. **C** — There are 36 equally likely ordered outcomes. A total of 2 needs (1,1) only, so 1/36. A total of 7 comes from (1,6), (2,5), (3,4), (4,3), (5,2), (6,1): 6/36 = 1/6.

689. **B** — After one ace is drawn, 3 aces remain among 51 cards: P = 4/52 × 3/51 = 12/2652 = 1/221. The student's 1/169 assumes the first card is put back.

690. **B** — The equally likely outcomes are HH, HT, TH and TT. One head and one tail happens in 2 of them, so P = 2/4 = 1/2. The student's three 'results' are not equally likely.

691. **C** — Average speed = total distance / total time. Equal times: (40 + 60) km / 2 h = 50 km/h. Equal distances give 48 km/h, because more time is spent at the slower speed. 2 h at 40 and 1 h at 60 gives 140/3 ≈ 46.7 km/h.

692. **C** — Shop A: 1.1 × 1.1 = 1.21, a 21% rise. Shop B: 1.20. The second 10% is taken on an already increased price, so successive percentages multiply rather than add.

693. **C** — Each of the 6 vertices connects to 3 non-adjacent vertices, but each diagonal has two ends, so 6 × 3 / 2 = 9. Formula check: C(6, 2) − 6 sides = 15 − 6 = 9.

694. **C** — All committees: C(8, 3) = 56. Committees containing both X and Y need 1 more member from the other 6: 6 ways. Subtract: 56 − 6 = 50.

695. **B** — Treat Sara and Hina as one block: 4 units can be arranged in 4! = 24 ways. Inside the block they can swap places (2 ways), so the total is 24 × 2 = 48.

696. **A** — Use the complement. P(no faulty) = C(8, 3)/C(10, 3) = 56/120 = 7/15. So P(at least one faulty) = 1 − 7/15 = 8/15.

697. **C** — Equally likely families: BB, BG, GB, GG. 'At least one boy' removes GG, leaving 3 cases, of which only BB has two boys. P = 1/3.

698. **D** — Let boys = 3k and girls = 2k. After 10 girls join, 3k = 2k + 10, so k = 10. The original group had 3k + 2k = 5k = 50 (30 boys and 20 girls).

699. **B** — Simple: 1,000 × 0.10 × 3 = 300. Compound: 1,000 × 1.1^3 − 1,000 = 331. The difference is 331 − 300 = 31, which is interest earned on earlier interest.

700. **C** — An algorithm is a finite, well-defined sequence of steps that turns input into the required output and terminates. It exists independently of any language: the same algorithm can be written as pseudocode, a flowchart or code.

701. **A** — Knuth lists finiteness, definiteness, input, output and effectiveness as properties of an algorithm. Finiteness means it always terminates after a finite number of steps; a procedure that can loop forever is not an algorithm in this sense.
702. **D** — Definiteness means every step is precisely and unambiguously specified. 'A few' could mean two or ten, so two people following the step could do different things.
703. **B** — A dry run is a manual execution on paper (or in your head) using chosen test values. You follow each step exactly as written, which exposes logic errors before or without running the program.
704. **B** — A trace table has one column per variable (plus conditions or output if useful). Each new row shows the values after a step or loop iteration, so you can see exactly how the state changes.
705. **B** — Writing a = b destroys a's old value. Storing it first (temp = a; a = b; b = temp) keeps it so it can be copied into b. Without the temporary variable, both end up holding b's value.
706. **A** — An accumulator builds up a running result, such as a sum, by adding each new value. A counter increases by a fixed amount (usually 1) to count events, a flag records a yes/no condition, and a sentinel marks the end of input.
707. **A** — Stepwise refinement, described by Niklaus Wirth, starts with a high-level outline and refines each step until it is simple enough to code directly. It keeps each sub-problem small and manageable.
708. **C** — As used in CLRS, a loop invariant is a property that holds before the first iteration and is preserved by every iteration. Combined with the exit condition, it proves the loop produces the right result. Variables can and usually do change while the invariant stays true.
709. **B** — Edge cases sit at the boundaries of valid input: no elements, exactly one element, and values that break hidden assumptions (such as negatives when the code starts from 0). Random typical inputs rarely expose these bugs.
710. **C** — A termination argument uses a variant: a value such as high − low or n that strictly decreases (or increases) on each iteration and cannot pass a fixed bound. When it reaches the bound, the loop condition becomes false.
711. **B** — A flag is a Boolean that records whether something happened. If a whole pass ends with swapped still False, no pair was out of order, so the list is sorted and further passes can be skipped.
712. **B** — The Böhm–Jacopini result shows that sequence (one step after another), selection (if/else) and iteration (loops) are sufficient to express any computable algorithm.
713. **A** — A sentinel is a special value that cannot be valid data and signals the end of input. The loop checks for it after each read; it must not be included in the sum or count.
714. **C** — CLRS uses three parts. Initialisation shows the invariant holds before the first iteration, maintenance shows each iteration preserves it, and termination uses the invariant plus the reason the loop stopped to prove the final result.
715. **D** — Dijkstra noted that testing shows the presence of bugs, not their absence. Passing 20 cases is evidence, not proof; correctness for all inputs needs a reasoning argument such as a loop invariant, or exhaustive testing of a finite input space.
716. **A** — Trace: x = 5 + 3 = 8; y = 8 − 3 = 5; x = 8 − 5 = 3. The values have been swapped without a temporary variable, so it prints 3 5.
717. **A** — range(1, 5) gives 1, 2, 3, 4 (the end value 5 is excluded). total becomes 1, 3, 6, 10. Choosing 15 wrongly includes 5.
718. **B** — i takes the values 1, 2, 4, 8, 16, 32, 64 at the start of each iteration; each is below 100, so the body runs 7 times. After the seventh run i = 128 and the loop stops.
719. **D** — m starts at 3. Trace x = 3 (not > 3), 8 (update, m = 8), 2 (no), 9 (update, m = 9), 4 (no). That is 2 updates; the initial assignment before the loop is not counted.
720. **C** — Each iteration adds the last digit (n % 10) and removes it (n // 10). s = 4, 7, 9, 10 as n goes 1234 → 123 → 12 → 1 → 0. The code sums the digits.
721. **D** — The digits are taken from the right: 0, 0, 2, 1. r goes 0 → 0 → 2 → 21. Leading zeros do not survive in an integer, so the result is 21, not 0021.
722. **C** — Trace (a, b): (48, 18) → (18, 12) → (12, 6) → (6, 0). The loop runs 3 times and stops when b = 0, leaving a = 6, the GCD of 48 and 18.
723. **D** — The inner loop runs i times: 0 + 1 + 2 + 3 = 6. Choosing 16 assumes the inner loop always runs 4 times.

724. **A** — Compare neighbours left to right: 5>1 swap → [1,5,4,2,8]; 5>4 swap → [1,4,5,2,8]; 5>2 swap → [1,4,2,5,8]; 5<8 no swap. One pass moves the largest unsorted value to the end but does not finish sorting.
725. **D** — low = 0, high = 9 → mid = 4 (16 < 23, low = 5). mid = (5 + 9) // 2 = 7 (56 > 23, high = 6). mid = (5 + 6) // 2 = 5 (23, found). Integer division rounds the midpoint down.
726. **A** — The average is total ÷ N, so the total must be built first with an accumulator. Sorting and finding the maximum are not needed, and printing must come last.
727. **C** — Multiples of 3 from 1 to 15 are 3, 6, 9, 12 and 15. The first branch catches 15 (divisible by both 3 and 5) and does nothing, so the elif counts only 3, 6, 9 and 12: 4. Order of conditions matters.
728. **D** — x goes 10 → 7 → 4 → 1 → −2. When x = 1 the condition is still true, so 3 is subtracted once more, giving −2; then x > 0 is false and the loop ends.
729. **C** — f(5) = f(4) + f(3), and so on down to f(1) = f(0) = 1, giving f values 1, 1, 2, 3, 5, 8, so f(5) = 8. The call counts follow calls(n) = 1 + calls(n−1) + calls(n−2) with calls(0) = calls(1) = 1: 3, 5, 9, 15.
730. **D** — m starts at 0, an assumed value instead of the first element. If every element is negative, none is greater than 0, so the function returns 0 instead of −2. The fix is m = a[0].
731. **A** — range(1, len(a)) starts at index 1, so a[0] = 4 is skipped: 7 + 1 + 6 = 14. Starting at 0 visits every index and gives 18.
732. **C** — found is overwritten on every iteration, so the function only reports whether the LAST element equals t. For [1, 2, 3] and 2, found becomes True at 2 but is reset to False at 3. The fix is to return True as soon as x == t (or use found = found or x == t).
733. **B** — i takes the values 0, 3, 6, 9, 12, 15 and so on. It is never exactly 10, so i != 10 is always true. Using i < 10 as the condition would guarantee termination.
734. **A** — a = b sets a to 7 and the old value 4 is lost. b = a then copies 7 back into b. Both are 7. A temporary variable (or Python's a, b = b, a) is needed.
735. **D** — Before the first iteration k = 0 and total = 0, the sum of no elements. Each iteration adds a[k] and then increments k, so the invariant 'total = a[0] + … + a[k − 1]' holds again. At exit k = len(a), giving the full sum.
736. **D** — j starts at i + 1, so each unordered pair is compared once: 9 + 8 + … + 1 + 0 = 45 = n(n − 1)/2. Sorting first and comparing neighbours would need far fewer comparisons for large n.
737. **B** — For n = 1, range(2, 1) is empty, the loop never runs and the function returns True, but 1 is not prime. A guard such as 'if n < 2: return False' fixes it.
738. **B** — Trace table (a, b): start (1, 1); after iteration 1 (1, 2); after 2 (2, 3); after 3 (3, 5). The simultaneous assignment uses the old a and b on the right-hand side.
739. **C** — Each pair (i, n − 1 − i) is checked twice: once from each end. Checking only the first half, range(n // 2), covers every pair once; the middle character of an odd-length string never needs checking. Tested on '', 'a', 'ab', 'abba', 'abca', 'racecar' and more.
740. **D** — Knuth allows an algorithm to have zero or more inputs (some need none) but requires one or more outputs, since an algorithm that produces nothing has solved nothing.
741. **B** — Pseudocode is a structured, language-independent way of writing steps. It lets you design and check the logic before choosing a language. It is not executed or compiled.
742. **B** — A diamond marks a decision with yes/no branches. Rectangles are processes, parallelograms are input/output, and ovals (terminators) mark start and end.
743. **B** — A counter increases by a fixed amount (usually 1) to record how many times something happened. An accumulator adds varying values, such as total = total + x.
744. **C** — Knuth's effectiveness property says each operation must be simple enough to be done exactly in a finite amount of time. Definiteness is about steps being unambiguous; finiteness is about the whole algorithm ending.
745. **D** — Reading values is input and displaying the result is output. The calculation that turns inputs into the result, area = length × width, is the process.
746. **C** — Selection chooses a path based on a condition. Repetition is iteration (loops), a fixed order is sequence, and storing many values is a data structure such as an array, not a control structure.
747. **B** — Abstraction removes irrelevant detail, for example modelling a city map as nodes and roads as edges. This keeps the algorithm simple and general.

748. **B** — A precondition states what the algorithm assumes about its input, such as 'the list is sorted' for binary search. A postcondition describes what is true after it finishes; a loop invariant holds on each iteration.
749. **A** — When you cannot know how many times the loop will run, you repeat while a condition holds (a while loop). Count-controlled (for) loops suit a known number of repetitions.
750. **A** — A post-test loop runs the body first and then tests the condition, so the body executes at least once. A pre-test while loop tests first, so its body can run zero times.
751. **D** — A loop ends only when its condition becomes false. If the body never changes the variables in the condition, or changes them in the wrong direction, the condition stays true forever.
752. **D** — Boundary testing checks values just outside and exactly on each limit, since off-by-one mistakes appear there. 0 and 101 must be rejected; 1 and 100 must be accepted.
753. **C** — The base case stops the recursion. Each recursive call must move the input closer to it; otherwise the calls never end and the program runs out of stack space.
754. **C** — Refinement continues until every step is precise and primitive enough to implement directly. A fixed number of levels has no meaning; the level of detail needed depends on the step.
755. **D** — With short-circuit and, if the left test is False the right test is never evaluated. Checking the bound first prevents an index error when i equals len(a).
756. **D** — If the target is the last item or is absent, linear search checks every item once: n comparisons. On average, when the target is present, it checks about n/2.
757. **A** — For each of the n outer iterations the inner body executes m times, so the total is n × m. Adding them would count each loop once only.
758. **C** — In base 10, n % 10 is the remainder on division by 10, which is the last digit; n // 10 is the integer quotient, which drops the last digit. Digit-processing loops repeat these two steps.
759. **C** — Partial correctness says: if the algorithm terminates, its output is correct. Total correctness adds that it does terminate on every valid input. A loop that never ends can be partially correct but not totally correct.
760. **B** — A variant such as high − low in binary search can only decrease a finite number of times before reaching its bound, so the loop must stop. The quantity that stays true on each iteration is the invariant, not the variant.
761. **B** — Turing showed there is no general procedure that decides, for every program and input, whether it halts. Individual loops can still be proved to stop, for example with a decreasing variant.
762. **D** — At the start, low..high covers the whole list. Each step discards only a half that cannot contain the target, so the invariant holds. When low > high the range is empty, so the target is absent.
763. **A** — b = 4 × 2 = 8, then a = 8 − 1 = 7. The old value of a (4) is gone once a is reassigned, so the output is 7 8.
764. **C** — p takes the values 1, 2, 6, 24 as i goes 1 to 4. range(1, 5) stops before 5, so the result is 4! = 24, not 5! = 120.
765. **A** — The even numbers are 8, 6 and 4, so c ends at 3. 18 is their sum, which the code does not compute.
766. **A** — n goes 5, 3, 1, then −1, which ends the loop. Each value is added to the right of s before n changes, giving '531'.
767. **B** — Each character is placed in front of r: 'c', 'oc', 'doc', 'edoc'. Prepending builds the string in reverse order.
768. **D** — // is integer (floor) division: 10 // 3 = 3. % gives the remainder: 10 % 3 = 1. Check: 3 × 3 + 1 = 10.
769. **D** — total becomes 50 + 70 + 90 = 210 and count becomes 3. In Python, / always returns a float, so 210 / 3 prints 70.0.
770. **D** — m becomes 2 at index 1. At index 3 the value is also 2, but 2 < 2 is False, so pos stays 1. A strict < keeps the first position of the minimum.
771. **B** — i = 0: 4 ≤ 10, so i = 1. i = 1: 9 ≤ 10, so i = 2. i = 2: 15 ≤ 10 is False, so the loop stops with i = 2, the index of the first value over 10.
772. **B** — Remainders on dividing by 2 are 1, 0, 1, 0, 0, 1 as n goes 37, 18, 9, 4, 2, 1. Each is put in front of bits, so the result is the binary form 100101 (32 + 4 + 1 = 37). Appending instead would give 101001.
773. **C** — c counts pairs where i + j is even. For i = 0: j = 0, 2, 4 (3). For i = 1: j = 1, 3 (2). For i = 2: j = 0, 2, 4 (3). Total 8 out of 15 pairs.
774. **C** — i = 1 inserts 3 before 7: [3, 7, 5, 1]. i = 2 inserts 5 between 3 and 7: [3, 5, 7, 1]. range(1, 3) stops before i = 3, so 1 is not yet placed.

775. **A** — n goes 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1. That is 8 changes, so steps = 8. Counting the values (9) instead of the moves between them is a common slip.

776. **D** — The loop visits every index without stopping. last is set to 1 when the first 8 is found and then overwritten with 3 at the second 8, so it ends holding the last position.

777. **D** — best holds the largest value seen so far and is appended after every element: 3, 3, 4, 4, 5. It is a running maximum, so it never decreases.

778. **A** — (20, 8) → (12, 8) → (4, 8) → (4, 4). The loop runs 3 times and stops when a equals b, so it prints the GCD 4 and the count 3.

779. **C** — h(13) = h(6) + '1', h(6) = h(3) + '0', h(3) = h(1) + '1', h(1) = h(0) + '1' = '1'. Building back up: '1', '11', '110', '1101'. The recursive call comes before the digit, so the digits appear in the correct (most significant first) order.

780. **B** — break only leaves the inner loop. For i = 1 to 4, the inner loop adds j = 1 to i: 1, 3, 6 and 10. The total is 1 + 3 + 6 + 10 = 20.

781. **B** — b = a makes both names refer to the same list, so b.append(4) changes it to [1, 2, 3, 4]. a = a + [5] builds a NEW list and rebinds only a, so a has 5 items while b keeps 4.

782. **A** — 1 + 9 = 10 < 13, so i moves (steps = 1). 3 + 9 = 12 < 13, so i moves (steps = 2). 4 + 9 = 13 breaks before steps is increased, so the output is 4 9 2.

783. **A** — Pass 1: a = 2 + 5 = 7, then b = 7 − 5 = 2. Pass 2: a = 7 + 2 = 9, then b = 9 − 2 = 7. Each line uses the value just updated by the line before it.

784. **C** — For an empty list, len(a) is 0, so sum(a) / len(a) raises ZeroDivisionError. A robust version checks for an empty list first. This is why the empty input belongs in every edge-case test set.

785. **D** — range(start, stop) stops before stop, so range(1, 10) gives 1 to 9. To include 10 the stop value must be 11. This is a classic off-by-one error.

786. **C** — The test only checks lower-case vowels, so the capital A in 'Apple' is missed and the function returns 1 instead of 2. Converting with ch.lower() fixes it.

787. **C** — The line total = 0 runs on every iteration and wipes out the running sum. After the loop, total holds only 0 + 6. Initialising an accumulator must happen once, before the loop.

788. **A** — The new b is a % b, which is less than the old b and never negative. A non-negative integer cannot decrease for ever, so b reaches 0 and the loop ends.

789. **C** — i is increased before it is used, so a[0] is skipped. On the last pass i becomes 3 and a[3] does not exist. Swapping the two lines in the body fixes it.

790. **D** — The check for −1 comes after the updates, so the sentinel is processed: (70 + 80 − 1) / 3 ≈ 49.67. Testing for the sentinel first, before updating, gives 150 / 2 = 75.

791. **B** — Before the loop, result = 1 = x^0 and i = 0. Each pass multiplies result by x and adds 1 to i, so result = x^i stays true. When the loop ends, i = n, so result = x^n.

792. **B** — For n = 0 the condition n > 0 is False at once, so the loop never runs and 0 is returned, but 0 has one digit. A do-while style loop, or a special case for 0, fixes it.

793. **D** — Searching for 0: mid = 1, high = 1; mid = 0, high = 0; then low = high = 0, mid = 0 and 1 > 0 sets high = 0 again. The range never shrinks, so the loop runs forever.

794. **A** — Both first and second start at a[0] = 9. No later value beats 9, and 4 and 7 never beat second = 9, so 9 is returned instead of 7. Initialising second to a value smaller than every element fixes it.

795. **A** — For n > 0, n // 2 is a smaller non-negative integer, so n strictly decreases and must reach 0. That decreasing quantity is the loop variant.

796. **A** — The else branch returns True after checking only the first pair. For [1, 3, 2], 1 ≤ 3, so it returns True without ever comparing 3 and 2. The return True belongs after the loop.

797. **C** — With a, b, c = 1, 2, 3, the first sequence gives 2, 3, 1 as required. Saving a in t first keeps its old value for c after a has been overwritten.

798. **A** — remove deletes the first 2, shifting later items left while the loop's position moves on. The second 2 slides into the position just checked, so it is skipped. Never change a list's length while looping over it; build a new list instead.

799. **C** — Conclusion indicators such as 'therefore', 'so', 'hence' and 'thus' introduce the claim being argued for. 'Because', 'since' and 'given that' usually introduce premises, the reasons offered in support.

800. **C** — An assumption is a premise the author takes for granted without stating it. If the assumption is false, the conclusion no longer follows, which is why finding it is the quickest way to test an argument.
801. **A** — Validity is about structure: it is impossible for the premises to be true and the conclusion false. A valid argument can still have false premises; validity alone says nothing about whether the premises are true.
802. **B** — A sound argument is valid and also has true premises, so its conclusion is guaranteed to be true. 'All cats can fly; Ali is a cat; so Ali can fly' is valid but unsound, because the first premise is false.
803. **A** — Two things moving together (correlation) does not show that one causes the other. Here a third factor, hot weather, drives both. Before accepting a causal claim, look for confounding variables.
804. **D** — Ad hominem ('to the person') dismisses a claim because of who says it, for example their grades, age or background, rather than examining the evidence or reasoning.
805. **C** — A straw man replaces the real claim with an exaggerated, weaker version that is easier to attack. The lead proposed testing one module, not stopping all feature work.
806. **D** — A false dilemma (false dichotomy) offers only two choices when more exist. Profiling and optimising the slow parts, caching or scaling the hardware are all ignored.
807. **A** — A hasty generalisation draws a broad conclusion from a sample that is too small or unrepresentative. Two phones cannot represent thousands of models from many manufacturers.
808. **D** — Sunk costs are already spent and cannot be recovered whatever you decide. Rational decisions compare future costs and benefits only; 'we have already spent six months on it' is not a reason on its own to continue.
809. **C** — Opportunity cost is the value of the best alternative you give up when you choose. Here the revision you could have done is the price of taking the freelance task.
810. **D** — A trade-off means gaining on one criterion by giving up something on another. The classic space-time trade-off spends extra memory to save computation time.
811. **A** — Confirmation bias is the tendency to look for, notice and remember evidence that supports what we already believe, while discounting evidence against it.
812. **C** — Inductive arguments generalise from observations, so even with true premises the conclusion is only probable. They are judged as strong or weak. Deductive arguments are judged as valid or invalid.
813. **D** — The form 'If P then Q; Q; therefore P' is the fallacy of affirming the consequent. A database fault or a bad deployment could also cause the error. The valid forms are modus ponens (P, so Q) and modus tollens (not Q, so not P).
814. **A** — Such an option is dominated (in Pareto terms). Whatever positive weights you choose, the dominating option scores at least as well, so the dominated one can be removed safely. Being worst on one criterion is not enough, because it may win on others.
815. **D** — The conclusion is the claim the other statements support; here it is signalled by 'So'. The first two sentences are premises. The last option is not stated in the argument at all.
816. **C** — Popularity is not evidence of fitness for a specific task. The framework should be judged on criteria such as the project's requirements, the team's skills and its documentation.
817. **B** — A = 0.5×6 + 0.3×9 + 0.2×7 = 3 + 2.7 + 1.4 = 7.1. B = 0.5×8 + 0.3×6 + 0.2×8 = 4 + 1.8 + 1.6 = 7.4. B wins because price carries the most weight.
818. **B** — Expected value = Σ probability × outcome = 0.6 × 100 + 0.4 × (−20) = 60 − 8 = 52. Ignoring the loss gives 60, and 80 is simply 100 − 20.
819. **D** — The conclusion only follows if server capacity is the bottleneck. If the real cause is an unindexed database query or large images, a bigger server will not help. Negating the assumption breaks the argument, which is the test for a required assumption.
820. **A** — A controlled comparison isolates the cache as the cause by keeping other factors equal. The other options are an appeal to authority, a confounded result (lower traffic) and an unmeasured impression.
821. **B** — Self-selection offers an alternative explanation: strong students both choose the app and score well, so the app may not cause the higher scores. Tuition could also matter, but 'some users' is too weak to explain the overall gap.
822. **A** — A slippery slope claims that a small first step will inevitably lead to an extreme outcome, without evidence for each link in the chain.
823. **C** — The 6 months are a sunk cost: they are gone whatever you choose. The decision should rest on future effort, risk and quality, which here favour switching unless the custom library offers something the other cannot.

824. **B** — EV = 0.5 × 40 + 0.3 × 10 + 0.2 × (−30) = 20 + 3 − 6 = 17. Treating the loss as a gain gives 29, and a simple average of 40, 10 and −30 (6.67) ignores the probabilities.
825. **D** — An array gives constant-time access by index, which matches the main need. Its O(n) middle insertion matters little if inserts are rare. A linked list needs O(n) to reach the i-th node and stores extra pointers.
826. **A** — Each item contributes 500 − 300 = 200 towards fixed costs, so break-even = 50,000 ÷ 200 = 250 items. Dividing by the price alone (100) ignores the variable cost.
827. **B** — Circular reasoning (begging the question) assumes what it sets out to prove. The source's reliability needs independent support, such as its track record or the checks it uses.
828. **B** — The premises say every CS student takes DSA, not that only CS students do. Sara could be an SE or IT student taking DSA, so the conclusion does not follow. This mirrors affirming the consequent.
829. **D** — The argument looks only at the visible successes. To judge the claim you need the success rate among all dropouts compared with graduates; the many unsuccessful dropouts are missing from the sample.
830. **B** — Before: 50 ÷ 10,000 × 1,000 = 5. After: 30 ÷ 5,000 × 1,000 = 6, a 20% rise. The raw fall in reports is explained by fewer users, so the manager's claim ignores a changed base.
831. **C** — A: easy 90%, hard 20%, overall 100/150 ≈ 66.7%. B: easy 85%, hard 10%, overall 175/250 = 70%. A wins in each group, yet B wins overall because most of its tests were easy. This is Simpson's paradox: compare like with like.
832. **B** — A = 0.2×6 + 0.5×9 + 0.3×7 = 1.2 + 4.5 + 2.1 = 7.8. B = 0.2×8 + 0.5×6 + 0.3×8 = 1.6 + 3.0 + 2.4 = 7.0. The ranking flips, so the decision is sensitive to the weights and they need justifying.
833. **A** — This is modus tollens: if Friday then approval; not approval; therefore not Friday. It is valid. The premises say nothing about failure or about other days.
834. **D** — V2 is dominated: V1 is cheaper, has higher uptime and better support, and V3 is cheaper with equal uptime and better support. V1 and V3 involve a real trade-off (lower cost against higher uptime), so weights are needed to choose between them.
835. **C** — This is post hoc reasoning. Scope changes, absences or new requirements could explain the missed deadlines. You would need to compare the work assigned and check for other changes before blaming one person.
836. **A** — EV(B) = 0.5 × 80 + 0.5 × 0 = 40, equal to A. Expected value alone cannot separate them; a risk-averse student with rent due would take A, while someone who can absorb a zero might take B.
837. **B** — 'A is necessary for B' means B cannot happen without A: if B then A. So every graduate passed the project. It does not make passing sufficient; other requirements may still apply.
838. **C** — X: $100 \times 100 \times \log_2 100 \approx 100 \times 100 \times 6.64 \approx 66{,}439$ steps. Y: $100^2 = 10{,}000$. Big-O describes growth for large n and hides constant factors, so the hidden assumption 'lower order means faster for every n' is false for small inputs.
839. **A** — Words such as since, because and given that introduce premises, the reasons offered for a claim. Here the premise is 'it runs on every page load' and the conclusion is 'we should cache this query'.
840. **B** — Modus ponens has the form: if P then Q; P; therefore Q. The second premise affirms the 'if' part (P), so the 'then' part (Q) must follow. It is valid.
841. **D** — The form is: if P then Q; not P; therefore not Q. This is denying the antecedent. Cheating is one way to fail, not the only way, so not cheating does not guarantee passing.
842. **A** — The cricketer is an authority on cricket, not on programming hardware. Citing someone's fame outside their field of expertise as proof is an appeal to an irrelevant authority.
843. **C** — A red herring introduces an irrelevant point that pulls attention away from the issue being discussed. The cloud bill does not explain or fix the slow database.
844. **A** — Tu quoque deflects criticism by accusing the critic of the same fault. Whether the reviewer's code had bugs has no bearing on whether Bilal's code has this bug.
845. **A** — Equivocation shifts the meaning of a key word mid-argument. 'Bug' means an insect in the first premise and a software defect in the second, so the premises do not connect.
846. **C** — Anchoring is relying too heavily on the first number heard. The client's 'two weeks' became a reference point, and the team adjusted too little away from it.
847. **A** — The availability heuristic judges how likely something is by how easily examples come to mind. A vivid, recent story feels more likely than it is, even when the data shows otherwise.

848. **A** — The planning fallacy is the tendency to underestimate how long a task will take, even when past tasks of the same kind ran over. The fix is to estimate from the record of similar tasks.
849. **D** — Groupthink happens when the wish for harmony stops a group from examining alternatives and objections. Asking someone to argue the other side is a common remedy.
850. **C** — A fact can be checked against evidence, and Python's first public release in 1991 is documented. The other statements are judgements that reasonable people can dispute.
851. **B** — In 'if P then Q', P is sufficient for Q. Being divisible by 4 guarantees the number is even, but it is not necessary: 6 is even and not divisible by 4.
852. **C** — A conditional is equivalent to its contrapositive: if P then Q equals if not Q then not P. The converse (if Q then P) and the inverse (if not P then not Q) are not equivalent.
853. **A** — Satisficing means choosing the first option that clears a set threshold. It saves search time and is reasonable when the extra gain from finding the best option is small. Maximising keeps searching for the optimum.
854. **C** — A confounding variable influences both of the things being compared, so it can create a correlation without either causing the other. Night shifts may drive both coffee drinking and commit activity.
855. **C** — A falsifiable claim makes a specific prediction that a test could contradict. Measuring the query time would confirm or refute the first claim; the others are vague or cannot be disproved.
856. **B** — Cost-benefit analysis weighs the gains of an option against what it costs, often converting both to money or time. The cheapest option is not automatically best if it delivers little benefit.
857. **C** — Decision trees use squares for decisions the decision-maker controls and circles for chance events. Each branch from a chance node carries a probability, and the probabilities sum to 1.
858. **C** — Occam's razor advises choosing the simpler explanation when competing ones explain the evidence equally well. It is a starting point for investigation, not proof.
859. **B** — The official specification or documentation is the primary source for a language's rules. Secondary and anonymous sources may be outdated or wrong, so they should be checked against it.
860. **A** — Under risk, outcome probabilities are known or can be estimated, so expected value applies. Under uncertainty they are unknown, so decision-makers use rules such as maximin or minimax regret instead.
861. **B** — The framing effect is when logically identical information leads to different choices depending on how it is worded. Gain framing ('stay online') tends to attract more approval than loss framing.
862. **D** — 'If and only if' joins two conditionals: if the password matches, the login succeeds (sufficient), and if the login succeeds, the password matched (necessary). This is a biconditional.
863. **D** — The premises are the reasons after 'because': the server is out of warranty and repairs cost more than a new one. 'We should replace it' is the conclusion they support.
864. **B** — Parts that work alone can still fail together, for example through mismatched interfaces or shared data. Assuming the whole inherits the properties of its parts is the fallacy of composition, which is why integration testing exists.
865. **D** — Expected value = 0.25 × 200 + 0.75 × 0 = 50. Multiply each outcome by its probability and add the products.
866. **A** — Chaining two conditionals (if P then Q, if Q then R) gives if P then R. This is a valid hypothetical syllogism. The other options reverse a conditional or assert something not given.
867. **A** — 0.5 × 7 + 0.3 × 5 + 0.2 × 10 = 3.5 + 1.5 + 2.0 = 7.0. The unweighted total would be 22.
868. **C** — X: 500 + 0.10 × 2,000 = 700. Y: 600 + 0.02 × 3,000 = 660. Y's higher base price is outweighed by its much smaller expected penalty.
869. **B** — Net monthly saving = 15,000 − 3,000 = 12,000. Payback = 120,000 / 12,000 = 10 months. Ignoring maintenance gives 8; adding it to the saving gives 6.7.
870. **C** — Opportunity cost is the value of the next-best alternative you give up, not the sum of all alternatives. Here that is the 75,000 offer.
871. **D** — The argument needs dark mode to be the cause of the growth and the effect to transfer to a different app and audience. Negate either part and the conclusion collapses.
872. **C** — If a new rule split work into more, smaller commits, the commit count rose without any more work being done. The measure changed, so it no longer supports the claim about productivity.

873. **C** — Apply the negation test: if test scores did not predict job performance, the highest score would be no reason to hire her. So the argument depends on that link.

874. **A** — The arrow probably runs the other way: incident-prone teams invest in monitoring, and monitoring finds incidents that would otherwise go unreported. Correlation alone cannot show which way causation runs.

875. **B** — A hard constraint removes options before scores are compared. The 8.2 laptop breaks the budget, so the best remaining option is the 7.9 laptop at 95,000.

876. **B** — The relative reduction is 50% (0.1 / 0.2), but the absolute reduction is 0.2% − 0.1% = 0.1 percentage points. Relative figures can make small absolute changes sound large.

877. **D** — Random selection gives every student an equal chance to be included, so the sample represents the population. Large self-selected or narrow samples are biased however big they are.

878. **D** — EV = 0.2 × 40 − 10 = 8 − 10 = −2. The fee is paid whatever happens. Forgetting it gives 8.

879. **B** — Per 1,000 emails: 100 spam × 0.95 = 95 flagged, and 900 genuine × 0.05 = 45 flagged. P(spam | flagged) = 95 / (95 + 45) ≈ 0.68. Ignoring the low base rate of spam leads to the 95% mistake.

880. **D** — Launch now: 0.3 × 100 + 0.7 × (−40) = 30 − 28 = 2. Prototype: −10 + 0.3 × 100 + 0.7 × 0 = 20, because the 40 loss is avoided. Forgetting the prototype cost gives 30.

881. **B** — C = 3.6 + 1.75 + 1.5 = 6.85. D = 2.4 + 2.8 + 1.75 = 6.95. E = 2.8 + 2.45 + 1.5 = 6.75. D wins narrowly, so the ranking is sensitive to the weights.

882. **A** — Maximin picks the option whose worst outcome is best. Worst cases: Small 50, Medium 40, Large 20, so Small wins. Choosing the best best case is maximax, and smallest regret is minimax regret.

883. **B** — Regret = best payoff in that column − the option's payoff. Low demand (best 50): Small 0, Medium 10, Large 30. High demand (best 120): Small 60, Medium 20, Large 0. Maximum regrets are 60, 20 and 30, so Medium minimises it.

884. **D** — Systematic data from many real sessions measures actual use. A single tweet is anecdotal, a belief is not evidence, and a competitor's feature says nothing about this product.

885. **D** — A rose 100% but by only 2 bugs; B rose 10% but by 5 bugs and now has 55. A big percentage on a tiny base can hide a small absolute change, so both views are needed.

886. **D** — The first is modus tollens (if P then Q; not Q; so not P), which is valid. The second affirms the consequent and the third denies the antecedent; decryption can fail for other reasons. The fourth assumes two 'some' groups overlap.

887. **A** — With imbalanced classes, a trivial model that always predicts the majority class matches the claimed accuracy while catching zero fraud. Measures such as recall on the fraud class are needed.

888. **A** — This is Goodhart's law: when a measure becomes a target, people optimise the measure rather than the goal. More lines do not mean more value and may mean more to review and maintain.

889. **D** — EV(A) = 100p. Setting 100p = 45 gives p = 0.45. Above 0.45, A has the higher EV; below it, B does. The current estimate of 0.5 is close to this point, so the choice is sensitive to it.

890. **A** — This is selection: the comparison groups were not alike to begin with. To judge tutoring, compare students' own scores before and after, or use a randomly assigned control group.

891. **C** — It is drawn from the first two statements and then used as a reason for the final recommendation, 'we should write tests for it first'. A claim that is both supported and supporting is an intermediate conclusion.

892. **C** — Choosing units for being extreme sets up regression to the mean: part of last week's slowness was random, so they would likely improve anyway. Comparing with similar unpatched servers isolates the patch's effect.

893. **B** — With 20 visitors, each conversion is worth 5 percentage points, so chance easily explains 10%. B must be tested on a sample comparable to A's before any claim about its rate is trusted.

894. **D** — EV(P) = 0.9 × 30 + 0.1 × (−80) = 27 − 8 = 19. EV(Q) = 15. EV ignores whether a single bad outcome is survivable; a loss of 80 exceeds the reserve of 50, so risk of ruin matters.

895. **D** — The first premise means all deployed code is tested. The deployed code that has bugs is therefore tested code, so some tested code has bugs. The other options claim more than the premises give.

896. **B** — Mean A = 140 / 5 = 28, median A = 10; mean and median B = 20. A is usually faster but has a bad tail; B is steady. Neither 'better for every workload' nor 'better in every respect' follows.

897. **B** — The missing link places the refund module inside the group the rule covers. With it, the argument becomes: all payment modules need review; refund is a payment module; so refund needs review.
898. **C** — This is outcome bias. With an 80% chance of success, failure happens one time in five even when the choice was right. Review whether the estimate and process were sound, not just the result.
899. **B** — A logic error lets the program run normally but produces the wrong result, because the code does something other than what the programmer intended. Syntax errors stop the program before it runs, and runtime errors crash it with a message.
900. **A** — Python parses the whole file before running it. A missing colon breaks the grammar, so the parser raises a SyntaxError and no line of the file runs. Syntax errors are found at parse or compile time, not at run time.
901. **B** — The line is valid syntax, so it parses. When it runs, dividing by zero raises ZeroDivisionError, which is a runtime error: it only appears when that line executes with a zero divisor.
902. **C** — An off-by-one error is a boundary mistake: a loop runs n+1 or n-1 times instead of n, or an index goes one past the end. It usually comes from mixing up < and <=, or forgetting that indexes start at 0.
903. **B** — A traceback lists the calls that were in progress when the exception was raised, most recent call last, with file names and line numbers. Read it from the bottom: the last line names the exception and the line just above shows where it happened.
904. **B** — A breakpoint tells the debugger to pause before the marked line runs. While paused you can read variable values, step into or over calls, and continue. It lets you check your expectations at a precise point without adding print statements.
905. **A** — Rubber duck debugging means explaining code aloud, line by line, to someone or something that cannot help. Forcing yourself to state what each line should do often exposes the line that does something else.
906. **A** — Debugging starts with reproducing the fault. Once you can trigger it on demand with a known input, you can test hypotheses and confirm that a fix works. Hiding the error or rewriting code blindly does not locate the cause.
907. **C** — A regression is a bug in behaviour that used to work, introduced by a later change. Regression testing re-runs earlier tests after every change so that such breakages are caught early.
908. **A** — Boundary value analysis tests values on and just outside each edge, because off-by-one errors live there. 17 and 61 should be rejected, while 18 and 60 should be accepted. Mid-range values rarely expose a wrong < or <=.
909. **A** — IndexError is raised when you access a sequence position that does not exist, for example a[3] on a three-element list (valid indexes 0 to 2). It is a common symptom of an off-by-one loop bound. An undefined variable gives NameError and a type clash gives TypeError.
910. **D** — A logic error is valid code that computes the wrong thing, so no tool can flag it without knowing your intent. You find it by comparing actual output with expected output on test cases and tracing the code. It can occur in any language.
911. **D** — git bisect marks one good and one bad commit, then repeatedly tests the commit halfway between them and halves the range. With n commits it needs about log2(n) tests, the same idea as binary search.
912. **C** — A Heisenbug changes or vanishes when you observe it, often because the observation alters timing or memory layout. Race conditions in concurrent code and reads of uninitialised memory in C are typical causes.
913. **D** — Binary floating point cannot represent 0.1 or 0.2 exactly, so 0.1 + 0.2 evaluates to 0.30000000000000004. Comparing floats with == is therefore unreliable; use abs(a - b) < eps or math.isclose(a, b). The is operator would be worse, because it checks identity.
914. **A** — A while loop stops only when its condition becomes false. If nothing in the body moves the loop variable towards that point, for example a forgotten i += 1, the loop never ends.
915. **D** — range(1, n) stops before n, so the loop adds 1 + 2 + 3 + 4 = 10 instead of 15. This is an off-by-one error; the fix is range(1, n + 1).
916. **D** — range(len(a) + 1) produces 0, 1, 2, 3. Indexes 0 to 2 print 3, 6 and 9, then a[3] is out of range and raises IndexError. Output that appears before a crash still appears, so the traceback follows the three numbers.
917. **D** — // is floor division, so 5 // 3 gives 1 and the fractional part is lost. Using / would return 1.6666666666666667. Choosing the wrong division operator is a classic logic error in averages.

918. **B** — A default argument value is created once, when the function is defined. Both calls share the same list, so the second call appends to the list that already holds 1. The usual fix is bucket=None and then creating a new list inside the function.

919. **C** — x = 0 is an assignment, not a comparison. It stores 0 in x and the expression's value is 0, which is falsy, so the else branch runs. The intended test was x === 0.

920. **A** — m starts at 0, so if every value is negative none of them beats 0 and the function returns 0 instead of -2. The other inputs all contain a value of at least 0, so they happen to give the right answer. The fix is to start with m = a[0].

921. **C** — [[0] * 2] * 2 copies the reference to one inner list, so both rows are the same object. Changing a[0][0] changes what a[1][0] shows too. Building rows with [[0] * 2 for _ in range(2)] creates two separate lists.

922. **D** — The check only looks for lowercase vowels. In "Education" the capital E is missed, so it counts u, a, i, o = 4 instead of 5. Calling ch.lower() before the test fixes it.

923. **C** — When one operand of + is a string, JavaScript converts the other to a string and concatenates, so "5" + 3 is "53". Converting first with Number(a) gives 8. This type-coercion bug is common with form input.

924. **C** — range(0, 10, 3) produces 0, 3, 6 and 9 and stops before 10, so the body runs 4 times. Listing the values is safer than dividing 10 by 3 in your head.

925. **B** — Posts stand at 0, 2, 4, ..., 20 m, which is 20 / 2 + 1 = 11 positions. 20 // 2 counts the 10 gaps, not the posts. This is the classic fencepost form of the off-by-one error.

926. **A** — Trace x after each pass: i = 0 gives 3*2 - 0 = 6, i = 1 gives 6*2 - 1 = 11, i = 2 gives 11*2 - 2 = 20. A trace table with columns i and x makes this quick and avoids skipping a step.

927. **D** — The base case only catches n == 0. Starting from -1, n goes -2, -3, ... and never reaches 0, so the calls continue until Python's recursion limit and a RecursionError. A safer base case is n <= 0, or validating the input.

928. **B** — Python reads the condition as (x == 1) or 2. x == 1 is False, but 2 is truthy, so the whole condition is true and "match" prints for every x. The intended test is x == 1 or x == 2, or x in (1, 2).

929. **D** — With lo < hi the loop stops when one candidate is left (lo == hi) without checking it, so a target at that final position is reported as missing. Changing the condition to lo <= hi checks the last candidate and the search becomes correct for every input.

930. **C** — A list of length 4 has 4 - 2 + 1 = 3 windows of size 2, starting at 0, 1 and 2. range(len(a) - k) gives only 0 and 1, so the last window [6, 8] is missed. The fix is range(len(a) - k + 1).

931. **B** — Division by zero needs len(a) == 0, which only the empty list gives. Lists that contain zeros or sum to zero still have a non-zero length, so they divide safely. The fix is to handle the empty case explicitly.

932. **A** — x = y overwrites x with 7 before its old value is saved, so y = x then copies 7 back. Both end as 7. A correct swap uses a temporary variable or Python's x, y = y, x.

933. **D** — total is reset only once, so the second row is added on top of 3, giving 3 + 7 = 10. Moving total = 0 inside the outer loop, before the inner loop, resets the accumulator for every row.

934. **B** — Removing an item while iterating shifts later items left. After the first 2 at index 1 is removed, the second 2 moves into index 1, and the loop goes on to index 2 (the 3), so it is skipped. Build a new list instead: [x for x in a if x != 2].

935. **C** — var is function-scoped, so all three callbacks share one i. The callbacks run only after the loop finishes, when i is 3. Declaring the loop variable with let creates a new binding per iteration and prints 0 1 2.

936. **C** — The message says + received a str and an int. Python does not convert automatically, so convert the number with str(total) or use an f-string such as f"Total: {total}". Reading the last line of the traceback points straight at the fix.

937. **A** — The loop exits when i reaches 10, so 10 is never printed. With i <= 10 it prints 2, 4, 6, 8, 10 and stops at 12. Simulating each candidate fix against the expected output is the reliable way to choose.

938. **B** — Each check halves the range of candidate stages: 1,024 → 512 → ... → 1, which takes log2(1024) = 10 checks. This is binary search applied to debugging, the same idea as git bisect.

939. **C** — Looking up a key that is not in a dictionary with square brackets raises KeyError. Use d.get('b') or check 'b' in d first if the key may be missing.

940. **B** — int() accepts a string argument, so the type is right, but 'abc' is not a valid number. A correct type with an unusable value raises ValueError.

941. **B** — items is None, and None has no append method. Calling a method or reading an attribute an object does not have raises AttributeError. It often means a variable was never given the object you expected.
942. **C** — Indentation defines blocks in Python, so inconsistent indentation is a syntax-level problem. IndentationError is a subclass of SyntaxError and is reported when the file is parsed.
943. **A** — Step into enters the called function. Step over runs the whole call and stops at the next line of the current function. Continue runs to the next breakpoint, and step out finishes the current function.
944. **C** — A watch keeps an expression, such as total or len(items), on screen and updates it at every pause. It saves you from adding print statements to track a value.
945. **D** — assert checks a condition the programmer believes must be true. If it is false, Python raises AssertionError, which stops the program close to where the assumption broke.
946. **C** — Java is statically typed, and the compiler checks types in every method, called or not. Assigning a String to an int is a compile-time error, so the program will not build.
947. **C** — A linter performs static analysis: it reads the code and warns about things such as unused variables, undefined names and suspicious comparisons. It finds some bugs before any test is run.
948. **D** — Python's logging levels run DEBUG < INFO < WARNING < ERROR < CRITICAL. DEBUG is the most detailed and is usually switched off in production.
949. **B** — Equivalence partitioning splits inputs into classes the program should treat the same way and tests one value from each class. Boundary value analysis instead tests the edges of those classes.
950. **A** — C does no bounds checking. Reading past the end is undefined behaviour, so the result depends on the compiler and memory layout. That is why such bugs can hide for a long time.
951. **A** — In ISTQB terms, a human error (mistake) introduces a fault (defect or bug) into the code, and executing the fault can cause a failure, a visible wrong behaviour.
952. **A** — A race condition occurs when the result depends on the timing of threads. Both threads can read the same old value, so one update overwrites the other. A lock or atomic operation fixes it.
953. **A** — A memory leak keeps memory reserved after it is no longer needed, for example by adding to a cache or list that is never cleared. Usage grows steadily until memory runs out.
954. **C** — A NullPointerException is thrown when code uses a null reference as if it pointed to an object. The other cases throw ArithmeticException, ArrayIndexOutOfBoundsException and NumberFormatException.
955. **D** — A minimal reproducible example strips away everything unrelated while keeping the bug. Cutting the code down often reveals the cause, and it lets others run and check the problem quickly.
956. **D** — A bare except with pass swallows all exceptions without a message. The program keeps going in a broken state, and the failure appears later far from its cause. Catch specific exceptions and log them.
957. **A** — [1, 2] == [1, 2] is True because the values match, but [1, 2] is [1, 2] is False because they are two separate list objects. Using is to compare values is a common hidden bug.
958. **B** — A Bohrbug is deterministic and reproducible, which makes it the easiest kind to debug. A Heisenbug changes or vanishes when you try to observe it, often because of timing.
959. **D** — As Dijkstra observed, testing can show the presence of bugs but never their absence. Tests only check the inputs they try, so coverage of edge cases and boundaries matters more than the count.
960. **C** — JavaScript numbers are 64-bit floating-point values, which represent integers exactly only up to 2^53 (Number.MAX_SAFE_INTEGER is 2^53 − 1). 2^53 + 1 rounds to 2^53. Use BigInt for larger exact integers.
961. **C** — counts[w] += 1 first reads counts[w], which does not exist yet, so the first word raises KeyError. Use counts[w] = counts.get(w, 0) + 1 or collections.Counter.
962. **B** — Python strings are immutable, so assigning to s[0] raises TypeError: 'str' object does not support item assignment. Build a new string instead, for example s = 'H' + s[1:].
963. **B** — range(5, 0, -1) stops before the end value 0, so 0 is never produced. Use range(5, -1, -1) to include 0. This is an off-by-one at the stop boundary.
964. **A** — The - operator has no string meaning, so JavaScript converts "10" to the number 10 and subtracts: 10 − 3 = 7. With + it would concatenate and give "103".
965. **C** — 4 % 2 is 0, and 0 == 1 is False. The condition tests for odd numbers. It should be n % 2 == 0. The code runs without error, so this is a logic error.

966. **A** — The first line rebinds the built-in name list to a list object. Calling list('abc') then tries to call that object and raises TypeError: 'list' object is not callable. Avoid naming variables after built-ins.
967. **A** — a / b divides two ints, so Java performs integer division and gets 2 before the result is widened to double, giving 2.0. Cast first: (double) a / b gives 2.5.
968. **C** — upper() returns a new string and leaves s unchanged, because strings are immutable. The result is discarded. The fix is s = s.upper().
969. **C** — b = a does not copy the list; both names refer to the same object. Appending through b changes what a shows. Use b = a.copy() or b = list(a) for a separate list.
970. **B** — The else branch returns False after checking only the first element, 3. The loop never reaches −1. return False belongs after the loop, so it runs only when no negative was found.
971. **C** — list.sort() sorts in place and returns None, so a = a.sort() replaces the sorted list with None. Use a.sort() on its own line, or a = sorted(a).
972. **D** — range(4) yields 0, 1, 2, 3, so the first multiplication is r *= 0 and r stays 0. The range should be range(1, n + 1), which multiplies 1 × 2 × 3 × 4 = 24.
973. **D** — There are three valid classes, so one representative from each is enough: 25 (F), 60 (C) and 85 (A). Values on the edges such as 49 and 50 belong to boundary value analysis, not partitioning.
974. **B** — leap(1900) returns True because 1900 % 4 == 0, but 1900 is a century year not divisible by 400, so it is not a leap year. 2024, 2023 and 2000 all give the correct answer by chance.
975. **B** — With no compare function, sort() converts elements to strings and orders them by character: "1" < "10" < "9". Pass a comparator, a.sort((x, y) => x - y), to sort numerically.
976. **B** — == converts types before comparing, and the empty string converts to the number 0, so 0 == "" is true. === compares without conversion, and a number is never strictly equal to a string, so it is false.
977. **D** — Python 3's round() uses round-half-to-even (banker's rounding): a value exactly halfway goes to the nearest even integer. So 2.5 becomes 2 and 3.5 becomes 4. Use decimal with ROUND_HALF_UP if halves must round up.
978. **B** — The lambdas capture the variable i, not its value at creation. They run after the loop ends, when i is 2, so all return 2. Binding the value with lambda i=i: i fixes it.
979. **D** — break leaves only the inner loop. For each of the 3 values of i, j = 0 adds 1 and j = 1 breaks, so found grows by 1 per outer pass: 3 in total.
980. **B** — Strings compare character by character, and "9" is greater than "1", so "9" is the maximum. Convert first, max(versions, key=int), to compare them as numbers.
981. **A** — a.copy() is a shallow copy: the outer list is new, but the inner lists are shared. Appending to b[0] changes the same inner list that a[0] refers to. copy.deepcopy(a) copies the inner lists too.
982. **C** — is_adult(18) returns False because 18 > 18 is false, but 18-year-olds are adults. Testing exactly on the boundary exposes the wrong operator; the fix is age >= 18.
983. **B** — range(1, len(s)) skips index 0, and 'b' only appears there. Starting the range at 0, or looping for ch in s, fixes it. count('banana', 'a') would still return 3, which is why the bug can hide.
984. **A** — On the first pass i = 0, so s[len(s) - i] is s[4], one past the last index of a 4-letter string, and IndexError is raised. The mirror of index i is len(s) − 1 − i.
985. **B** — The bottom frame of the traceback is parse, where row[1] fails for the row ['b']. The malformed input row is the root cause; parse should validate the row length or the data should be cleaned.
986. **D** — The sum of positives is 12, but it is divided by all 3 elements, giving 4.0. Counting positives (2) and dividing by that gives 6.0. The other changes do not touch the wrong divisor.
987. **D** — After 100 is good and 150 is bad, the first bad commit lies between 101 and 150. Bisection tests the midpoint of that range, about 125, halving the remaining suspects again.
988. **C** — The output is correct, so this is not a logic error. Pairwise checking takes $O(n^2)$ time: 50,000 × 49,999 / 2 ≈ 1.25 billion comparisons. Adding items to a set and checking membership is O(n) on average.
989. **D** — For n = 0 the condition n > 0 is false at the start, so the loop body never runs and c stays 0, though 0 has one digit. All positive inputs work. A fix is to return 1 when n == 0.
990. **B** — When visits is 0 (for example, after a tracking failure), sales / visits raises ZeroDivisionError, and the except returns 0 with no log. The error is hidden, so the dashboard shows a plausible but false value.

991. **C** — 0.1 has no exact binary form, so after ten additions x is 0.9999999999999999, not 1.0. The loop skips past 1.0 and runs forever. Use while x < 1.0, or count with integers.
992. **D** — range(n - i - 2) stops one comparison early, so the last pair in each pass is never compared and 1 never moves past index 2 into place. With n - i - 1, every adjacent pair in the unsorted part is compared.
993. **D** — Python does not allow a dictionary to change size while it is being iterated. Iterating over list(orders) takes a snapshot of the keys first, so deleting entries is safe. A comprehension that builds a new dict also works.
994. **A** — map calls parseInt(value, index). parseInt('1', 0) treats radix 0 as 10 and gives 1; parseInt('2', 1) has an invalid radix and gives NaN; parseInt('3', 2) is NaN because 3 is not a binary digit. Use map(Number) or map(s => parseInt(s, 10)).
995. **A** — Python's // floors towards negative infinity: -17 // 10 = -2, -2 // 10 = -1, and -1 // 10 = -1, so n never reaches 0. In C, integer division truncates towards zero, which is why code ported from C can break. Use abs(n) first.
996. **A** — For page 1, the slice is items[10:20], so items 0-9 are never shown, and the last page starts past the end of the list. Subtracting 1 from page converts the 1-based page number into a 0-based offset.
997. **B** — Big-O gives an asymptotic upper bound on growth: f(n) = O(g(n)) means f grows no faster than a constant times g for large n. It ignores machine speed and constant factors, so it compares approaches rather than timing them.
998. **B** — Keep the fastest-growing term and drop constant factors. For large n, $3n^2$ dominates 5n + 20, and the constant 3 is dropped, so the complexity is $O(n^2)$.
999. **D** — An array stores elements in contiguous memory, so the address of a[i] is base + i × element size. That is one calculation whatever the size of the array, so access is constant time, O(1).
1000. **A** — The usual order from slowest to fastest growth is O(1), O(log n), O(n), O(n log n), $O(n^2)$, $O(n^3)$, $O(2^n)$. Exponential growth eventually overtakes any polynomial, which is why $O(2^n)$ algorithms are impractical beyond small inputs.
1001. **B** — Each comparison halves the remaining range, so after k steps $n/2^k$ elements remain. The search ends when that reaches 1, after about $\log_2 n$ steps, so the worst case is O(log n).
1002. **B** — Θ (theta) is a tight bound. f(n) = Θ(g(n)) when $c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$ for large n. CLRS defines it this way; in everyday use people often write O when they mean Θ.
1003. **B** — Ω (omega) is an asymptotic lower bound: $f(n) \ge c \cdot g(n)$ for all large n. O is the upper bound and Θ means both hold. These bounds are separate from best, average and worst case, which describe which inputs you analyse.
1004. **B** — In the worst case the target is last or absent, so every one of the n items is compared once. Binary search would be faster, but it needs sorted data.
1005. **A** — Extra (auxiliary) space counts memory beyond the input. The loop uses only total and a loop counter, whatever n is, so it needs O(1) extra space. The input array itself is O(n) but is not counted as extra.
1006. **A** — Memoisation stores earlier results, which costs extra memory, so later calls skip the work and save time. Trading memory for speed, or the reverse, is the core idea of a time-space trade-off.
1007. **B** — With a good hash function keys spread evenly, so a lookup checks a constant number of slots on average. If many keys collide into one bucket, a lookup can scan all n of them. Balanced-tree maps give O(log n) in every case instead.
1008. **D** — A dynamic array sometimes copies all its items into a larger block, which costs O(n). Because capacity grows geometrically, these copies are rare, and the total cost of n appends is O(n). That is O(1) per append amortised.
1009. **D** — log_a n = log_b n / log_b a, so changing the base multiplies by a constant. Big-O drops constants, so every base gives the same class. That is why $O(\log_2 n)$ and $O(\log_{10} n)$ are the same.
1010. **C** — A comparison sort must tell apart all n! orderings of its input, and a decision tree with n! leaves has height at least $\log_2(n!)$, which is Θ(n log n). Counting sort and radix sort beat this only because they do not compare elements.
1011. **C** — On sorted input each new element is compared once with its left neighbour and stays put, so n - 1 comparisons are made: O(n). Reverse-sorted input is the worst case, $O(n^2)$.
1012. **D** — The merge step copies elements into a temporary array, which needs O(n) extra memory. The recursion stack adds O(log n), which is smaller. Merge sort's time is O(n log n) in every case, but it is not in-place.

1013. **B** — The inner loop runs n times for each of the n outer passes, so the body runs n × n = 100 times. That is why two full nested loops over n are $O(n^2)$.
1014. **D** — For each i the inner loop runs i times: 0 + 1 + ... + 9 = n(n - 1)/2 = 45. Even though it is roughly half of n^2, the growth is still $O(n^2)$.
1015. **C** — i takes the values 1, 2, 4, 8, 16, 32 inside the loop; after doubling to 64 the condition i < 64 fails. That is 6 iterations, which is $\log_2 64$, so the loop is O(log n).
1016. **C** — The inner while loop halves j from 16 down to 1 in 4 steps ($\log_2 16$). It runs for each of the 16 outer passes, giving 16 × 4 = 64 steps. A linear loop around a halving loop is O(n log n).
1017. **C** — For each i the inner loop runs about n/i times, so the total is n(1 + 1/2 + 1/3 + ... + 1/n). That harmonic sum is about ln n, giving O(n log n). For n = 12: 12 + 6 + 4 + 3 + 2 + 2 + 1 + 1 + 1 + 1 + 1 + 1 = 35.
1018. **D** — Sequential loops add rather than multiply: n + n + n = 3n, and dropping the constant gives O(n). Only nesting multiplies the counts.
1019. **A** — Doubling n multiplies n^2 by $2^2 = 4$, so the time grows from about 2 s to about 8 s. For an O(n) algorithm it would double to 4 s.
1020. **D** — A is faster when $100n < n^2$, which means n > 100. At n = 100 they tie at 10,000 steps, and below that B wins. A lower growth rate wins only past the crossover point.
1021. **D** — Calls satisfy C(n) = C(n-1) + C(n-2) + 1 with C(0) = C(1) = 1, so C(2) = 3, C(3) = 5, C(4) = 9 and C(5) = 15. The count grows exponentially, which is why naive recursive Fibonacci is slow.
1022. **C** — The inner loop runs i times while i halves: 16 + 8 + 4 + 2 + 1 = 31. This geometric series is less than 2n, so the total is O(n) even though the loops are nested.
1023. **C** — range(1, 20, 3) gives 1, 4, 7, 10, 13, 16 and 19, which is 7 values. In general the count is ceil((stop - start) / step) = ceil(19 / 3) = 7.
1024. **A** — j starts at i + 1, so each unordered pair is compared once: n(n - 1)/2 = 100 × 99 / 2 = 4,950. 10,000 counts ordered pairs including self-pairs, and 5,050 is 1 + 2 + ... + 100.
1025. **B** — Building a set costs about 10^6 steps and each query is then O(1) on average, about 10^6 in total. Linear search costs about $1{,}000 \times 10^6 = 10^9$ comparisons. Sorting once and binary searching (about 2×10^7) is good too, but sorting for every query is far worse.
1026. **C** — A list membership test scans items one by one, so it is O(n). A set is a hash table, so it hashes x and checks a bucket, which is O(1) on average. Switching a list to a set is a common quick win.
1027. **A** — The inner loop runs a fixed 5 times regardless of n, so the body runs 5n times. Dropping the constant gives O(n). Count what each loop depends on instead of counting how many loops there are.
1028. **A** — The loop continues while $i^2 \le n$, so it stops once i passes √n. For n = 100 it runs 10 times. This is the bound used in trial-division prime checks.
1029. **D** — The recursion tree has $\log_2 n$ levels and each level does n work in total, so T(n) = Θ(n log n). The Master Theorem gives the same result (a = 2, b = 2, f(n) = n, case 2). Merge sort follows this recurrence.
1030. **D** — Each call with k > 0 makes two calls with k - 1, so the number of calls is $2^{n+1} - 1$. For n = 4 that is 31. The call tree doubles at each level, so the time is exponential, $O(2^n)$.
1031. **B** — With memoisation each value fib(0) to fib(n) is computed once and later calls are O(1) dictionary lookups. That is n + 1 computations, so the time is O(n), with O(n) extra space for the memo. For n = 20, naive recursion makes 21,891 calls and the memoised version computes 21 values.
1032. **A** — $n^2 = 10^{12}$ operations, and $10^{12} / 10^8 = 10^4$ seconds, about 2.8 hours. An O(n log n) method needs about 2×10^7 operations, roughly 0.2 seconds. Back-of-envelope estimates like this tell you which algorithm is feasible.
1033. **A** — list.insert(0, x) shifts every existing item one place right, which is O(k) for a list of k items. Summed over n inserts that is $O(n^2)$. Appending is amortised O(1), and deque.appendleft is O(1), so either fix brings the total to O(n).
1034. **C** — If the target is at position k it takes k comparisons, so the average is (1 + 2 + ... + 9) / 9 = (n + 1)/2 = 5. The average case is still O(n), because n/2 differs from n only by a constant.
1035. **C** — At n = 500, X takes 1,000 × 500 = 500,000 steps and Y takes $500^2 = 250{,}000$, so Y wins. X only becomes faster once n > 1,000. Big-O predicts behaviour for large n; constant factors still matter at small sizes.
1036. **A** — Each recursive call stays on the stack until the base case returns, so n frames are active at once: O(n) stack space. A loop needs only an accumulator, O(1). (If each call also slices the list, as a[1:], the copies add even more work, but the stack depth alone is O(n).)

1037. **A** — Constant time does not grow at all, a logarithm grows very slowly, linear grows in step with n, and n log n grows a little faster than linear because each of the n units carries an extra log n factor.

1038. **B** — Keep the fastest-growing term and drop constant factors. 7n outgrows 3 log n and the constant 12, so the running time is O(n). Writing O(7n) is not wrong in value but is never the standard form.

1039. **C** — A stack only touches its top element, so push and pop take constant time no matter how many items are stored. With a dynamic array the occasional resize makes push amortised O(1).

1040. **D** — Space complexity describes the memory an algorithm needs (usually the extra or auxiliary memory) as a function of the input size n, written with the same O, Ω and Θ notation as time complexity.

1041. **A** — Merge sort always splits the array in half and merges in linear time, so it is O(n log n) in every case. Bubble, selection and insertion sort are all $O(n^2)$ in the worst case.

1042. **B** — Big-O classifies functions by growth rate. Doubling n doubles both 5n and n, so they belong to the same class; constant factors depend on hardware and coding details, which Big-O deliberately ignores.

1043. **C** — Once n is larger than 1000, n^3 exceeds $1000n^2$, and the gap keeps widening. The dominant term decides the class, so $n^3 + 1000n^2$ is $O(n^3)$.

1044. **D** — Complexity is written as a function of input size: the number of elements in an array, characters in a string, or vertices in a graph. It is not a time or a hardware measure.

1045. **A** — Linear search checks items one by one, so it does the most work, n comparisons, when the target is last or missing. Sortedness and duplicates do not change that for a plain linear search.

1046. **B** — The head is available directly. A linked list has no index arithmetic, so reaching the k-th node means following k links one at a time, which is O(k), unlike an array's O(1) indexing.

1047. **C** — You create the node, point it at the current head and make it the new head. Only two pointer updates happen regardless of n, so the insertion is O(1). Arrays need O(n) for the same job because elements shift.

1048. **D** — $\log_2 10^6$ is about 20, so $n \log_2 n \approx 2 \times 10^7$, while $n^2 = 10^{12}$. That gap of roughly 50,000 times is why an O(n log n) sort finishes in moments where an $O(n^2)$ sort may take hours.

1049. **A** — If the very first comparison matches, the search stops after one step regardless of n. A missing target is the worst case, and sorting does not help a plain linear scan.

1050. **B** — BFS enqueues each vertex once and scans each adjacency list once, so it does work proportional to V plus E. $O(V^2)$ is the cost with an adjacency matrix, and O(E log V) belongs to Dijkstra with a binary heap.

1051. **C** — On sorted input the first element is the smallest, so each partition leaves $n - 1$ items on one side. The depths add up to $n + (n - 1) + \ldots + 1$, which is $O(n^2)$. Random or median-of-three pivots avoid this case.

1052. **D** — O gives an upper bound, so anything bounded by n is also bounded by n^2. It is true but loose. When an exam asks for the tight bound, Θ(n) or the tightest O is expected.

1053. **A** — If $T(n + 1) = 2T(n)$, then T(n) is proportional to 2^n, which is exponential growth. O(2n) is just O(n), which grows by a constant amount per extra element rather than doubling.

1054. **B** — Both operations restore the heap property by moving one element along a root-to-leaf path, and a complete binary tree has height about $\log_2 n$. Reading the minimum without removing it is O(1).

1055. **C** — An in-place algorithm rearranges the input inside its own array using a constant amount of extra memory (some texts allow O(log n) for recursion). Keeping equal elements in order is stability, a separate property.

1056. **D** — There are n choices for the first position, $n - 1$ for the second and so on, giving n! orderings. n! grows even faster than 2^n; 2^n counts subsets, not orderings.

1057. **A** — $\log(n!) = \log 1 + \log 2 + \ldots + \log n$. It is at most n log n, and the upper half of the terms alone are each at least $\log(n/2)$, giving at least $(n/2)\log(n/2)$. Both bounds are n log n, so $\log(n!) = \Theta(n \log n)$. This is the source of the comparison-sort lower bound.

1058. **B** — Each step does a constant amount of work and continues in one half, so $T(n) = T(n/2) + c$. Halving reaches size 1 after about $\log_2 n$ steps, so $T(n) = O(\log n)$.

1059. **C** — $2^{n+1} = 2 \cdot 2^n$, just a constant factor of 2, so it is $O(2^n)$. But $2^{2n} = 4^n = 2^n \cdot 2^n$, and the ratio 2^n is not bounded by any constant, so 4^n is not $O(2^n)$. A constant in the exponent's base or multiplier matters.

1060. **D** — $\log(n^2) = 2 \log n$, a constant multiple of log n, so both are Θ(log n). In the other pairs the ratio grows without bound: n, $(3/2)^n$ and log n respectively.

1061. **A** — range(0, 20, 2) gives 0, 2, 4, …, 18, which is 10 values. Stepping by 2 halves the count but the loop is still O(n).

1062. **B** — Linear time grows in proportion to n. Multiplying n by 4 multiplies the time by 4, so 3 × 4 = 12 seconds. 48 seconds would be the quadratic estimate ($4^2 = 16$ times).

1063. **C** — Each probe halves the remaining range. $\lfloor \log_2 1000 \rfloor + 1 = 9 + 1 = 10$, so at most 10 probes are needed. 500 is a linear search's average, not binary search.

1064. **D** — Three nested loops, each running n times independently, give $n \times n \times n = 5^3 = 125$ executions, so the code is $O(n^3)$.

1065. **A** — The two loops run one after the other, not nested, so their costs add: n iterations plus m iterations. Because n and m are independent sizes, keep both in the answer.

1066. **B** — $\log_2 1024 = 10$ and $\log_2 2^{20} = 20$. Squaring the input only doubles a logarithmic running time, which is why logarithmic algorithms scale so well.

1067. **C** — For each i, j runs from i to n − 1, giving n − i iterations: 10 + 9 + … + 1 = 55, which is $n(n + 1)/2$ and $O(n^2)$.

1068. **D** — i goes 81 → 27 → 9 → 3 → 1, so the body runs 4 times and then i > 1 fails. Dividing by 3 each time gives $\log_3 n$ iterations, which is still O(log n).

1069. **A** — i takes the values 1, 2, 4, 8, 16, so the outer loop runs $\log_2 32 = 5$ times, and each pass runs the inner loop 32 times: 5 × 32 = 160. In general that is n log n.

1070. **B** — The grid holds 1,000 × 1,000 = 10^6 integers, and $10^6 \times 4$ bytes = 4,000,000 bytes, about 4 MB. Memory grows with n^2, so doubling n quadruples it.

1071. **C** — Sorting costs O(n log n) once. Each of the k binary searches then costs O(log n), adding k log n. The two phases are sequential, so the costs add rather than multiply.

1072. **D** — With a hash set each lookup and insert is O(1) on average, so one pass is O(n). Nested pairs cost $O(n^2)$, sort-plus-binary-search costs O(n log n), and subsets cost $O(2^n)$.

1073. **A** — Building prefix[i] = a[0] + … + a[i − 1] takes one pass, O(n). Each query is then prefix[r + 1] − prefix[l], O(1), so q queries add O(q). Summing each range directly would cost O(n) per query.

1074. **B** — f(8) calls f(4) twice, each f(4) calls f(2) twice, and each f(2) calls f(1) twice: 1 + 2 + 4 + 8 = 15 calls. This recurrence, T(n) = 2T(n/2) + O(1), is O(n).

1075. **C** — Each + creates a new string and copies all existing characters, so the copies total 1 + 2 + … + n, which is $O(n^2)$. A StringBuilder appends into a growing buffer in amortised O(1), making the whole build O(n).

1076. **D** — The first window costs k additions, then each of the remaining n − k slides costs two operations. Since $k \le n$, the total is O(n). Recomputing each window from scratch would cost O(n × k).

1077. **A** — i goes 2 → 4 → 16 → 256 → 65,536, so the body runs 4 times. Squaring doubles the exponent each time (2^1, 2^2, 2^4, 2^8, 2^{16}), so the number of steps is $\log_2 \log_2 n = \log_2 16 = 4$.

1078. **B** — Unrolling gives n + (n − 1) + … + 1 = n(n + 1)/2, which is $O(n^2)$. Only one recursive call is made, so there is no exponential blow-up, and the input shrinks by one each time, not by half.

1079. **C** — For each i, j doubles from 1 until it reaches i, giving $\lceil \log_2 i \rceil$ steps: 0, 0, 1, 2, 2, 3, 3, 3 for i = 0…7, which sums to 14. Summing log i over n values is O(n log n).

1080. **D** — $\log_2 10^6 \approx 20$, so $n \log_2 n \approx 2 \times 10^7$ operations, and $2 \times 10^7 / 10^8 = 0.2$ seconds. An $O(n^2)$ algorithm on the same input would need 10^{12} operations, close to 3 hours.

1081. **A** — Naive matrix multiplication uses three nested loops, so it is $O(n^3)$. Tripling n multiplies the time by $3^3 = 27$. Nine seconds would be the quadratic estimate.

1082. **B** — The pass costs O(n) and Python's sorted() (Timsort) costs O(n log n). Sequential steps add, and n + n log n is dominated by n log n.

1083. **C** — The inner loop runs only on the first outer iteration, adding n steps once. The outer loop contributes n steps, so the total is about 2n, which is O(n). Nesting in the code does not always mean nested cost.

1084. **D** — in on a Python list is a linear scan, O(n). Doing it once for each of the n queries gives $O(n^2)$. Converting lst to a set first makes each test O(1) on average and the whole function O(n).

1085. **A** — Binary search needs O(1) access to the middle element. In a linked list you walk n/2 links, then n/4, and so on, which adds up to about n. So the total is O(n), no better than a linear scan.

1086. **B** — A heap keeps the minimum at the root and restores order in O(log n) after each insert or removal. Re-sorting each time costs O(n log n), and a sorted array needs O(n) shifting to insert. A stack returns the most recent item, not the smallest.

1087. **C** — With memoisation the cost is (number of distinct states) × (work per state). There are n × n pairs, each doing O(1) work, so the total is $O(n^2)$. This is how 2-D dynamic programming tables are costed.

1088. **D** — Heap sort meets both limits: it sorts in place with constant extra memory and is O(n log n) in every case. Standard merge sort needs an O(n) buffer, quick sort can degrade to $O(n^2)$, and insertion sort is $O(n^2)$.

1089. **A** — x not in result scans result, which can grow to n items, so all-distinct input costs 1 + 2 + … + n = $O(n^2)$. Keeping a set of seen values makes each check O(1) on average, giving O(n) while the list still preserves order.

1090. **B** — An early exit improves the best case, but the worst case, all-distinct items, still runs every pair: n(n − 1)/2 comparisons, which is $O(n^2)$.

1091. **C** — 100 → 50 → 25 → 24 → 12 → 6 → 3 → 2 → 1 is 8 steps. An odd value is always followed by a halving step, so there are at most about 2 $\log_2$ n iterations, which is O(log n).

1092. **D** — The inner loop runs ⌊√i⌋ times, so the count is the sum of ⌊√i⌋ for i = 1…16: 3×1 + 5×2 + 7×3 + 1×4 = 38. Summing √i over n values grows like n^1.5, which is O(n√n).

1093. **A** — Level k of the recursion tree has 2^k calls doing constant work each: 1 + 2 + 4 + … + n ≈ 2n. So T(n) = O(n). O(n log n) arises only when each level does linear work, as in merge sort's 2T(n/2) + n.

1094. **B** — The work is n + n/2 + n/4 + … which is a geometric series less than 2n. So T(n) = O(n). Quickselect's average case follows this pattern.

1095. **C** — Each recursive call adds a stack frame, and the depth equals the number of halvings, about $\log_2$ n, so the recursive version uses O(log n) space. The loop version keeps only lo, hi and mid, which is O(1).

1096. **D** — On sorted input each element needs one comparison, so insertion sort is Θ(n) in that case. On reverse-sorted input every element shifts past all earlier ones, giving n(n − 1)/2 shifts. One input family cannot establish the worst-case bound.

1097. **A** — A trace table has one column per variable (and often one for output) and one row per step. Filling it in line by line shows exactly how values change, which is the most reliable way to predict output or spot a fault.

1098. **D** — % is the modulo (remainder) operator, so 17 % 5 is 2. // is floor division (17 // 5 is 3), / is true division and ** is exponentiation. Modulo is used for even/odd tests, digit extraction and wrap-around indexes.

1099. **B** — A while loop repeats as long as a condition holds, so it suits cases where the number of repetitions is not known in advance. A for loop over range(10) caps the attempts at a fixed count, which is a different requirement.

1100. **A** — An accumulator collects a running result such as a sum, count or product. It is set once before the loop and updated on each pass. Resetting it inside the loop throws away the earlier values.

1101. **D** — and short-circuits: if a is falsy the result is already known to be falsy, so b is skipped. This is why a guard such as len(s) > 0 and s[0] == 'x' is safe on an empty string. or does the reverse and skips b when a is truthy.

1102. **D** — Assigning to a name inside a function creates a local variable by default, so the global x is untouched. To change the global you need a global x statement, though returning a value is usually cleaner.

1103. **C** — Negative indexes count from the end, so a[-1] is the last element and a[-2] the one before it. a[len(a)] is one past the end and raises IndexError, which is a classic off-by-one.

1104. **A** — return ends the function and gives a value to the code that called it, which can store or reuse it. print writes text to the screen and the function still returns None. Output questions often hinge on a function that prints instead of returning.

1105. **B** — A flag is a Boolean that records whether a condition has happened. After the loop you test the flag to decide what to report. An accumulator, by contrast, builds a running value such as a sum.

1106. **D** — The base case stops the recursion, and each recursive call must make the problem smaller so that the base case is eventually reached. Missing either part leads to infinite recursion and, in Python, a RecursionError.

1107. **C** — Python's // is floor division: it rounds towards negative infinity, so -3.5 becomes -4. C, C++ and Java integer division truncates towards zero and gives -3. This difference is a common trap when porting code.

1108. **A** — ** is right-associative, so the expression is 2 ** (3 ** 2) = 2 ** 9 = 512. Reading it left to right as (2 ** 3) ** 2 gives 64. Most other binary operators in Python are left-associative.

1109. **A** — Python passes a reference to the object. append mutates that object, so the caller sees the change. Rebinding the parameter (lst = [...]) would not affect the caller, because it only changes the local name.

1110. **A** — Python strings cannot be changed in place, so assigning to s[0] raises TypeError. Build a new string instead, for example "H" + s[1:].

1111. **C** — In a dry run you act as the computer: you follow each line with a chosen input and record the values, usually in a trace table. It is how you answer output-prediction questions without a computer.

1112. **B** — Stepwise refinement (top-down design) starts from a one-line statement of the task and splits it into smaller steps, then splits those again until each is a few lines of code. Each step often becomes a function.

1113. **C** — Only even values of i are added: 2 + 4 = 6. range(1, 6) gives 1 to 5, and the odd values fail the i % 2 == 0 test.

1114. **A** — A slice with step -1 walks the string from the end to the start, so [::-1] reverses it.

1115. **B** — Each pass adds the last digit (n % 10) and then removes it (n // 10): 6 + 9 + 0 + 4 = 19. The loop runs once per digit, 4 times here.

1116. **D** — Each pass shifts rev left by one digit and adds the last digit of n. Trace: rev = 3, 30, 302, 3021 while n goes 120, 12, 1, 0. The middle zero is kept because it is added as a digit, not dropped.

1117. **D** — x % 2 is 1 (truthy) for odd x, so only 1 and 3 pass the filter. Their squares are 1 and 9.

1118. **B** — a = 1 is positional, b keeps its default 2 and c is set to 4 by keyword. So the result is 1 + 2 * 4 = 9, since * binds tighter than +.

1119. **B** — For i = 0 the inner loop adds "0"; for i = 1 it adds "01"; for i = 2 it adds "012". Joining them gives 001012. The inner range is range(i + 1), so it grows by one each pass.

1120. **B** — g computes Fibonacci numbers with g(0) = 0 and g(1) = 1: 0, 1, 1, 2, 3, 5, 8, 13. g(7) is the eighth term, 13. Writing the sequence from the base cases up is quicker than expanding the call tree.

1121. **D** — The print comes after the recursive call, so p(3) first calls p(2), which calls p(1), which calls p(0) and returns. The prints then run as the calls unwind: 1, 2, 3. Placing print before the call would give 3 2 1.

1122. **A** — b = a copies the reference, not the array, so a and b name the same array. Pushing through b changes the array that a sees, and its length becomes 4. Use [...a] or a.slice() to make a copy.

1123. **D** — The loop counts each word with a dictionary: get returns 0 for a new word, then adds 1. "a" appears at positions 1, 3 and 6, so counts["a"] is 3.

1124. **D** — Without a compare function, Array.prototype.sort converts elements to strings and sorts them in UTF-16 code-unit order. "10" comes before "2" because '1' < '2'. Use sort((x, y) => x - y) for numeric order.

1125. **C** — x = x + [4] builds a new list and rebinds x to it, so y still refers to the original [1, 2, 3]. By contrast, x += [4] extends the list in place, and y would then show [1, 2, 3, 4].

1126. **C** — A for loop's else block runs when the loop finishes without hitting break. i never equals 5 in range(3), so break never runs and "done" prints. This is often used for search loops: else means "not found".

1127. **C** — The task needs a count of values above the mean, so add 1 for each x that passes a strict > test. For a = [2, 4, 6] the mean is 4 and only 6 is greater, so the answer is 1. (Recomputing the mean inside the generator is $O(n^2)$; storing it in a variable first is better practice.)

1128. **C** — For n = 1 the range(2, 1) is empty, so the loop never runs and the function returns True, yet 1 is not prime. 2 and 13 correctly return True and 9 correctly returns False. The fix is to return False for n < 2 first.

1129. **C** — range excludes its stop value, so to include √n itself you need int(n ** 0.5) + 1. Without the + 1, perfect squares of primes such as 9, 25 and 49 are reported as prime. Checking up to √n is enough, because any factor pair has one member ≤ √n.

1130. **B** — i and j start at 0 and 6 and move inwards: (0,6), (1,5), (2,4) all match. Then i = j = 3 and the loop stops, because the middle character needs no comparison. So there are 3 comparisons, about n/2 for a string of length n.

1131. **B** — Because a = b runs first, b = a + b adds the new a to itself, so b doubles each time. Trace (a, b): (1, 2), (2, 4), (4, 8), (8, 16), (16, 32), (32, 64), so a is 32. With a, b = b, a + b both right-hand values are taken before assignment and the code would print 8.

1132. **A** — The sequence is 6, 3, 10, 5, 16, 8, 4, 2, 1. That is 9 numbers but 8 steps, because the count goes up once per transition. Mixing up values and steps is a common off-by-one in tracing questions.

1133. **B** — i = 0 and 1 are added, i = 2 is skipped by continue, i = 3 is added, and at i = 4 break exits before adding. So s = 0 + 1 + 3 = 4.

1134. **A** — Each lambda refers to the variable i, not its value at creation time. The functions are called after the comprehension finishes, when i is 2, so all three return 2. Writing lambda i=i: i captures the value at each pass and gives [0, 1, 2].

1135. **D** — str.count counts non-overlapping occurrences, scanning left to right. It finds "ana" at index 1, then continues from index 4, where only "na" remains. The overlapping match at index 3 is not counted, so the result is 1.

1136. **C** — Each call adds the last digit and recurses on the rest: 5 + 7 + 8 + 9 + h(0), and h(0) = 0, giving 29. This is the recursive version of the digit-sum loop.

1137. **A** — Problem solving starts with understanding: what data comes in, what result must come out, and the conditions linking them. Language choice and naming come later, once the algorithm is clear.

1138. **C** — A sentinel is a special input value that signals the end of the data and is not processed as data itself. A flag is a true/false marker, an accumulator collects a running total, and a counter counts events.

1139. **D** — Reading the marks is input, calculating the average from them is processing, and displaying the result is output. The IPO model helps split a problem into these three parts before coding.

1140. **A** — Writing a = b first would overwrite a's value. The standard swap is temp = a; a = b; b = temp. Python's a, b = b, a does the same thing in one line.

1141. **B** — int() parses a string of digits into an integer. str() goes the other way, float() gives 42.0, and len() returns 2, the number of characters.

1142. **C** — Edge cases sit at the limits: zero items, one item, the minimum or maximum value. Bugs often hide there because loops and conditions behave differently at boundaries.

1143. **D** — break leaves the nearest enclosing loop and continues with the code after it. continue is the statement that skips the rest of the current iteration. Outer loops keep running.

1144. **A** — A missing colon breaks Python's grammar, so the code will not run at all. The other mistakes produce valid code that runs but gives wrong answers, which makes them logic errors.

1145. **B** — input() returns the typed text as a str, even when the user types digits. To do arithmetic you must convert it, for example int(input()).

1146. **C** — x = 5 stores 5 in x, while x == 5 evaluates to True or False. Python rejects = inside an if condition, which prevents a classic C bug.

1147. **D** — Good tests cover different behaviour: a single item, all negatives (catches a max initialised to 0) and duplicates of the maximum. The other sets all repeat the same easy case of positive ascending values.

1148. **A** — In C, dividing two ints truncates towards zero, giving 3. In Python 3, / is always true division and returns 3.5; Python's floor division operator // would give 3.

1149. **B** — Empty containers, 0, 0.0, None and False are falsy. The string "0" and the string of one space are non-empty, and [0] is a list with one element, so all three are truthy.

1150. **C** — Every Python function returns something. If no return runs, it returns None. That is why result = print("hi") leaves result set to None.

1151. **D** — A precondition must hold before a function is called for it to work correctly. A postcondition is what the function guarantees afterwards, and an invariant holds at every iteration of a loop.

1152. **A** — A loop invariant is a condition that stays true at the start and end of each iteration. When the loop ends, the invariant plus the exit condition proves the result is correct.

1153. **B** — Most decimal fractions have no exact binary representation, so 0.1 + 0.2 gives 0.30000000000000004. Compare floats with a tolerance, for example math.isclose(a, b).

1154. **C** — Python has function scope, not block scope, so the loop variable stays defined with its last value, 2. In Java or C++ a variable declared in the for header disappears after the loop.

1155. **D** — Stability matters when sorting records by one field after another. Python's sorted() is stable, so sorting students by name and then by grade keeps names in order within each grade.

1156. **A** — Java's int is a 32-bit two's-complement type, and overflow is defined to wrap: 2147483647 + 1 becomes -2147483648 with no error. Use long or Math.addExact when overflow must be caught.

1157. **B** — Python 3 rounds halves to the nearest even integer (banker's rounding), so 2.5 → 2 and 3.5 → 4. This avoids a systematic upward bias when many values are rounded.

1158. **C** — For objects, == tests identity. Two separately created strings with equal text are different objects. s1.equals(s2) compares the contents, which is what you almost always want.

1159. **D** — The second stable sort keeps equal-grade students in the order the first sort left them, which is alphabetical. Sorting by the secondary key first, then the primary key, is a standard multi-key technique.

1160. **A** — range(2, 10, 3) starts at 2 and adds 3 while staying below 10: 2, 5, 8. The stop value 10 is excluded, so 11 never appears.
1161. **B** — m becomes 9 at index 1. At index 3 the value is also 9, but 9 > 9 is false, so idx stays 1. A strict > keeps the first occurrence of the maximum.
1162. **C** — 7 % 3 is the remainder, 1, and 7 // 3 is the floor quotient, 2. Python evaluates both before adding, so y = 1 + 2 = 3.
1163. **D** — The letters e, u, a, i and o are vowels; d, c, t and n are not. The loop adds one for each vowel, so it prints 5.
1164. **A** — With + and a string operand, JavaScript converts 3 to "3" and concatenates, giving "53". The - operator has no string meaning, so "5" is converted to a number and 5 - 3 = 2.
1165. **B** — Multiplying a string by an integer repeats it, so "ab" * 3 is "ababab" with no separators.
1166. **C** — The default list acc is created once, when the function is defined, and shared by every call that omits acc. The first call leaves [1] in it, so the second call appends 2 and returns [1, 2]. Use acc=None and create a new list inside.
1167. **D** — (48, 18) → (18, 12) → (12, 6) → (6, 0). When b reaches 0, a holds the greatest common divisor, 6.
1168. **A** — The remainders are 1, 0, 1, 1 (13 → 6 → 3 → 1 → 0). Each new bit goes at the front of the string, so the result reads 1101, the binary form of 13.
1169. **B** — inner looks up x when it runs, not when it is defined. By the time inner() is called, outer has set x to 5, so it returns 5 + 1 = 6.
1170. **C** — strip() removes the outer spaces, lower() gives "nsct 2026", and split() with no argument splits on whitespace into ['nsct', '2026'].
1171. **D** — a[1:4] takes indices 1, 2 and 3 (the stop is excluded), giving [2, 3, 4]. a[-2:] starts two from the end and runs to the end, giving [4, 5] in normal order.
1172. **A** — Compare and swap neighbours left to right: 5 and 1 swap, 5 and 4 swap, 5 and 2 swap, 5 and 8 stay. One pass moves the largest value to the end but does not finish sorting.
1173. **B** — The loop adds 1 to 10 except multiples of 3. The full sum is 55 and the skipped values 3 + 6 + 9 = 18, so s = 55 - 18 = 37.
1174. **C** — Any non-empty string is truthy, including "False", so bool("False") is True. The integer 0 is falsy, so bool(0) is False.
1175. **D** — 10 / 3 is 3.333..., and the format spec :.2f rounds to two decimal places, giving 3.33.
1176. **A** — [[0] * 3] * 3 makes a list holding three references to the same inner list. Changing grid[0][0] changes that one shared row, so every row shows 1. Build rows with [[0] * 3 for _ in range(3)] instead.
1177. **B** — var is function-scoped, so all three callbacks share one i. The callbacks run after the loop has finished, when i is 3. With let, each iteration gets its own i and the output is 0 1 2.
1178. **C** — Java evaluates left to right. x++ yields 10 and sets x to 11; ++x then sets x to 12 and yields 12. The sum 10 + 12 = 22 is assigned to x. (In C or C++ the same line is undefined behaviour.)
1179. **D** — return 1 is evaluated, but the finally block always runs before the function actually returns, printing F with a trailing space. Then print shows the returned 1 on the same line.
1180. **A** — r(1) = [] + [1] + [] = [1]. r(2) = [1] + [2] + [1] = [1, 2, 1]. r(3) = r(2) + [3] + r(2) = [1, 2, 1, 3, 1, 2, 1], which has $2^3 - 1 = 7$ items.
1181. **C** — range(1, n) stops before n, so it adds 1 to 4 and gives 10. range(1, n + 1) includes n and gives 15. This is a classic off-by-one error at the upper bound.
1182. **D** — m starts at 0, which is larger than every value in an all-negative list, so the function returns 0 instead of -1. Initialise with m = a[0] to fix it.
1183. **A** — split(" ") splits on every single space, so the double space creates an empty string: ['a', '', 'b'], length 3. split() with no argument treats any run of whitespace as one separator and ignores leading and trailing spaces, giving 2.
1184. **B** — A year is a leap year if it divides by 4, except centuries, which must also divide by 400. So 2024 and 2000 are leap years, while 1900 is not.
1185. **C** — On the first pass, a[0] is 4, not 7, so the code reaches return -1 inside the loop and exits. The return -1 belongs after the loop, so every item is checked first.

1186. **D** — The loop runs over the whole list, so each pair is swapped twice: once in the first half and back again in the second half. Looping only to n // 2 reverses it correctly.
1187. **A** — Ten additions of 0.1 give 0.9999999999999999, not 1.0, so the loop steps past 1.0 and runs forever. Loop on a counter, or use x < 1.0 or a tolerance check.
1188. **B** — Removing the first 2 shifts the second 2 into the current index, and the loop then moves on, skipping it. Build a new list instead: [x for x in a if x != 2].
1189. **C** — Each trailing zero needs a factor 5 paired with a 2. The loop adds 100 // 5 = 20 multiples of 5, then 20 // 5 = 4 multiples of 25 that give an extra 5, then 0. Total 24.
1190. **D** — (1, 11) = 12 is too big, so j moves left. (1, 8) = 9 is too small, so i moves right. (3, 8) = 11 is too big, then (3, 6) = 9 is too small, and finally (4, 6) = 10 matches.
1191. **A** — The base case returns 0, and every level multiplies by it: 5 × 4 × 3 × 2 × 1 × 0 = 0. The empty product is 1, so fact(0) must return 1.
1192. **B** — For 7, mid is 1, then 2, moving lo to 3. Now lo == hi, so the loop exits without checking a[3] and returns -1. With an inclusive hi the condition must be lo <= hi.
1193. **C** — x goes 1 → 1×1+1 = 2 → 2×2+1 = 5 → 5×3+1 = 16 → 16×4+1 = 65. Stopping before 4 would give 16, the usual mistake when TO is read as exclusive.
1194. **D** — a.copy() makes a new outer list holding the same inner lists. b[0].append(9) changes the inner list shared with a. b[1] = [0] only rebinds a slot of b's outer list, so a[1] is still [3]. copy.deepcopy would isolate a completely.
1195. **B** — The gap between two percentages is measured in percentage points: 50 - 40 = 10 points. Relative to the starting value it is 10 / 40 = 25%. Saying 'rose by 10%' would mean 40% became 44%.
1196. **A** — A line chart puts time on the x-axis and joins consecutive values, so the reader sees the trend, peaks and dips at a glance. Pie charts show shares of a whole, not change over time.
1197. **D** — Part-to-whole data with a few categories that sum to 100% is what a pie chart or a 100% stacked bar is designed for. Line charts show trends and scatter plots show relationships between two numeric variables.
1198. **D** — A scatter plot places each student as a point with study hours on one axis and marks on the other, so any linear or curved relationship, and any outliers, are visible.
1199. **C** — The median is the middle value of the sorted data, so one huge value only moves it by at most one position. The mean adds every value, so a single extreme value pulls it strongly.
1200. **A** — Correlation does not prove causation. Hot weather is a confounding variable: it drives both ice-cream sales and the number of people swimming, so the two series move together without one causing the other.
1201. **C** — A histogram groups a continuous or numeric variable into bins and draws bars touching each other, so you see the shape of the distribution. A bar chart compares distinct categories, with gaps between bars.
1202. **A** — Bar length is read as size. With the axis starting at 95, the bars are 1 and 3 units tall, so 98 looks three times 96 even though it is only about 2% higher. Bar charts should normally start at zero.
1203. **D** — IQR is the interquartile range, Q3 - Q1. Tukey's fences put boundaries 1.5 × IQR below Q1 and above Q3; points outside them are flagged as possible outliers.
1204. **D** — Members of a study group are likely to be more prepared than average, so the sample does not represent the whole population. This is selection (sampling) bias, and a larger biased sample does not fix it.
1205. **B** — The mode is the most frequent value. 7 appears three times, 4 appears twice and the others once, so the mode is 7.
1206. **C** — Standard deviation measures spread around the mean. Equal means with a larger standard deviation mean class Q has more very low and very high marks; it says nothing about class size or which class did better.
1207. **C** — In Simpson's paradox the group sizes differ so much that the combined figures tell the opposite story to every subgroup. It is a reason to break data down before drawing conclusions.
1208. **A** — r ranges from -1 to +1; -0.9 shows a strong negative linear association. It is not a slope, not a percentage of causation, and correlation alone cannot establish cause.
1209. **B** — Blanks may mean 'absent', 'not yet entered' or 'zero'. Treating them as 0 silently drags the average down, so you should understand the cause and choose a handling rule you can justify and report.
1210. **C** — A few very high values pull the mean up but barely move the median, so mean > median usually signals right (positive) skew. It is a rule of thumb rather than a theorem, but it holds for typical income data.

1211. **C** — Percentage change = (new - old) / old × 100 = (180 - 120) / 120 × 100 = 50%. Dividing by the new value gives 33.3%, a common base error; 150% is the ratio 180/120, not the growth.
1212. **C** — Sum = 62 + 75 + 81 + 75 + 90 = 383, and 383 / 5 = 76.6. 75 is both the median and the mode, which is why it is a tempting distractor.
1213. **B** — Change = 68 - 80 = -12 thousand. Divided by the original 80 gives -0.15, so -15%. Dividing by 68 (the new value) gives -17.6%, which uses the wrong base.
1214. **A** — Weight each average by its size: (30 × 60 + 20 × 70) / 50 = (1800 + 1400) / 50 = 64. The simple average of 60 and 70 (65) ignores that Section A is larger.
1215. **B** — Sorted: 95, 98, 105, 110, 120, 300. With an even count the median is the mean of the two middle values: (105 + 110) / 2 = 107.5 ms. The mean, 138 ms, is pulled up by the 300 ms outlier.
1216. **D** — A full circle is 360°, so the angle is 0.35 × 360 = 126°. Each 1% of the whole corresponds to 3.6°.
1217. **B** — 10,000 × 1.20 = 12,000, then 12,000 × 0.80 = 9,600. A 20% fall is taken on the larger March figure, so equal percentage rises and falls do not cancel out; the net change is -4%.
1218. **D** — CAGR = (end / start)^(1/years) - 1 = (1331 / 1000)^(1/3) - 1 = 1.1 - 1 = 10%. Dividing the total growth of 33.1% by 3 (11.03%) ignores compounding.
1219. **B** — Gains: CS 52 - 45 = 7, IT 47 - 38 = 9, SE 56 - 50 = 6, AI 49 - 41 = 8 percentage points. IT has the largest gain (it also has the largest relative gain, 23.7%). SE has the highest rate but the smallest gain.
1220. **A** — IQR = 60 - 40 = 20. Upper fence = Q3 + 1.5 × 20 = 90. Only 95 lies above 90; 85, 88 and 89 are inside the fence even though they are high.
1221. **D** — Rates: A 54/1200 = 4.5%, B 40/800 = 5.0%, C 60/1500 = 4.0%, D 27/600 = 4.5%. Page C has the most orders, but conversion is a ratio, so B wins.
1222. **D** — Range = maximum - minimum = 61 - 38 = 23. Using Monday's 40 as the minimum (giving 21) is a common misreading of the chart.
1223. **C** — A 3-day moving average on day 5 averages days 3, 4 and 5: (15 + 11 + 16) / 3 = 14. 12.8 is the average of all five days, not a moving average.
1224. **D** — A = 100/2000 = 5.0%, B = 140/2500 = 5.6%. The absolute gap is 0.6 percentage points; relative lift = 5.6 / 5.0 - 1 = 12%. Comparing raw sign-ups ignores that B had more visitors.
1225. **B** — Morning: X 81/90 = 90%, Y 19/20 = 95%. Evening: X 2/10 = 20%, Y 24/80 = 30%. Y wins both batches, but 80 of Y's 100 students were in the hard evening batch. This is Simpson's paradox: the batch mix, not the tutor, drives the overall figures.
1226. **A** — Motivated students may both visit the library and study harder, so the correlation need not be causal. Testing the policy on a randomly chosen group, compared with a control group, would show whether visits themselves help.
1227. **A** — 52 / 49 - 1 = 0.061, so B is about 6.1% higher. The bars appear as 4 and 1 units tall only because the axis starts at 48; on a zero baseline they would look almost equal.
1228. **B** — Conversions: P = 10, Q = 18, total 28 out of 1,000 leads = 2.8%. Averaging the two rates (6%) treats both branches as equal size, but Q handled nine times as many leads.
1229. **A** — Per 1,000 assignments: 20 copied, 19 flagged; 980 original, 49 wrongly flagged. Flagged total = 68, of which 19 are copied: 19 / 68 ≈ 27.9%. The low base rate means most flags are false alarms.
1230. **B** — Looking only at successes hides the much larger group of dropouts whose ventures failed. To judge the claim you need success rates for dropouts and graduates alike, not just the survivors.
1231. **C** — The four entered marks sum to 300. Dividing by 6 (treating blanks as zero) gives 50; dividing by the 4 students who sat the quiz gives 75. Missing is not the same as zero.
1232. **D** — Compare like with like. Nov 2025 vs Nov 2024: 500/480 - 1 = 4.2%. Dec 2025 vs Dec 2024: 420/360 - 1 = 16.7%. The month-on-month fall is seasonal, and December's year-on-year growth is much stronger than the pre-campaign trend.
1233. **C** — The owner's 400 pulls the mean to 650 / 6 ≈ 108.3, higher than five of the six salaries. The median, (50 + 55) / 2 = 52.5, sits in the middle of what most staff earn.
1234. **A** — The promise is about the 95th percentile (p95), the time within which 95% of requests finish. A's p95 is 260 ms, which breaks the promise; B's p95 is 190 ms, which meets it. The mean is the wrong statistic here.

1235. **D** — Separate named categories (programmes) compared at one point in time suit a bar chart. A line chart implies a trend over an ordered axis such as time, a scatter plot needs two numeric variables, and a histogram shows the distribution of one numeric variable.

1236. **C** — A pivot table aggregates raw rows: it groups by one or more fields (region, month) and applies a summary such as SUM, COUNT or AVERAGE. It is the spreadsheet equivalent of SQL GROUP BY.

1237. **D** — Data work starts from a well-defined question; it tells you which data to collect, which metric matters and when you are done. Collecting everything or building dashboards first produces noise without an answer.

1238. **B** — The box spans Q1 to Q3 (the middle 50% of the data) and the line inside it marks the median (Q2). Box plots do not show the mean by default, although some tools add a separate marker for it.

1239. **A** — A heatmap is a grid (here weekday by hour) in which each cell's value is encoded by colour intensity. It is good for spotting patterns such as slow periods across two categorical dimensions.

1240. **C** — A percentile is a position in the distribution, not a mark. The 90th percentile means about 90% of scores are at or below this one, so the student is roughly in the top 10%, whatever the raw mark was.

1241. **B** — r runs from -1 (perfect negative linear relationship) through 0 (no linear relationship) to +1 (perfect positive linear relationship). It is unit-free, so it is never a percentage.

1242. **A** — When people choose whether to respond, those with strong opinions reply more often, so the 12% are not representative of all students. That is voluntary response (non-response) bias; a random sample with follow-up reminders would reduce it.

1243. **C** — Lines of code was a rough signal of output, but once it became the target people optimised the number instead of the goal. That is Goodhart's law, and it is why teams pair metrics or measure outcomes rather than activity.

1244. **B** — A Pareto chart combines descending bars with a cumulative-percentage line. It shows where, for example, 80% of bug reports come from about 20% of modules, so effort can go to the few causes that matter most.

1245. **B** — Random assignment balances known and unknown factors (device, location, motivation) between the groups. Then the page is the only systematic difference, which lets you read the result causally. Equal group sizes are convenient but not the point.

1246. **B** — A log scale spaces values by their logarithm, so each equal step is the same multiplication factor. Steady exponential growth therefore appears as a straight line, which is why log axes are used for user growth or case counts.

1247. **A** — Standard deviation measures spread around the mean; it is 0 only when every value equals the mean, so all values are identical. The mean can be any value, and many identical readings also give 0.

1248. **A** — Raw counts depend on population size. If City A has 5 million residents and City B 1 million, A's rate is 0.1 per 1,000 against B's 0.3, the opposite conclusion. Normalising by a denominator is a core data-driven habit.

1249. **A** — Inconsistent formats make dates ambiguous (3 April or 4 March?), which corrupts any time-based analysis. Standardising to ISO 8601 based on where each entry came from removes the ambiguity; spreadsheet guessing is exactly what causes the error.

1250. **D** — The few very low marks pull the mean down towards the tail, while the median depends only on the middle position. So in a left (negative) skew the mean is usually below the median.

1251. **C** — A cohort is a group that shares a starting event, such as sign-up month. Comparing cohorts' retention over time shows whether newer users stay longer than older ones, which totals across all users would hide.

1252. **B** — A leading indicator moves before the outcome and can still be acted on; weekly practice completion predicts later results. The final pass rate is a lagging indicator: it only tells you the outcome after it is too late to change.

1253. **B** — A p-value is computed assuming the null hypothesis (no real difference) is true: it is the probability of data at least this extreme under that assumption. It is not the probability that either version is better, and it says nothing about the size of the effect.

1254. **C** — P(at least one false positive) = 1 - 0.95^20 = 1 - 0.358 = 0.64. Running many tests and reporting only the winner (the multiple-comparisons problem) makes a fluke very likely; the fix is to adjust the threshold or retest the winner.

1255. **C** — Quadrupling n multiplies sqrt(n) by 2, so 1/sqrt(n) and the margin of error are halved. This is why precision gets expensive: to halve the error you need four times the sample.

1256. **D** — Predicting within the observed range (interpolation) is fairly safe; predicting far beyond it assumes nothing changes, such as market saturation or competition. Long extrapolations of short trends are a classic source of wrong forecasts.

1257. **B** — The data describes countries, not people. A country-level correlation can hold even if, within each country, internet access makes little difference to individual earnings, so conclusions about individuals need individual-level data.

1258. **C** — Sort the values: 3, 4, 5, 8, 10. With five values the median is the 3rd one, 5. The mean would be 30 / 5 = 6, which is a common mix-up; 8 is the middle value of the unsorted list.

1259. **C** — Failed = 18% of 250 = 45, so passed = 250 - 45 = 205 (equivalently 82% of 250). 45 is the number who failed, not passed.

1260. **D** — Percentage change uses the starting value of that step: (100,000 - 80,000) / 100,000 = 20%. The rise was 25% because its base was 80,000, so the same Rs 20,000 is a different percentage each way.

1261. **A** — Total = 12 + 18 + 20 + 30 + 40 = 120. 25% of 120 is 30, which is Thursday. Friday is the largest day at 40 / 120 = 33.3%.

1262. **A** — Rate per 1,000 = 120 / 48,000 x 1,000 = 2.5. Rates like this let you compare towns of different sizes fairly.

1263. **B** — A full pie is 360 degrees, so the share is 72 / 360 = 0.20 = 20%. The angle itself is not a percentage.

1264. **A** — Weighted GPA = (3 x 4.0 + 3 x 3.0 + 2 x 2.0 + 1 x 3.0) / (3 + 3 + 2 + 1) = 28 / 9 = 3.11. The simple average of the four grade points, (4 + 3 + 2 + 3) / 4 = 3.00, ignores the credits.

1265. **B** — Average speed = total distance / total time = 120 / (60/30 + 60/60) = 120 / 3 = 40 km/h. Averaging the two speeds (45) is wrong because the van spends twice as long at the slower speed.

1266. **A** — Mean = 40 / 8 = 5. Squared deviations: 9, 1, 1, 1, 0, 0, 4, 16, total 32. Population variance = 32 / 8 = 4, so the standard deviation is sqrt(4) = 2. Dividing by n - 1 = 7 gives the sample value 2.14, and 4 is the variance, not the deviation.

1267. **B** — z = (x - mean) / sd = (76 - 60) / 8 = 2, so the mark is two standard deviations above the mean. 16 is the raw gap before dividing by the standard deviation.

1268. **C** — IQR = Q3 - Q1 = 70 - 35 = 35. It describes the spread of the middle half of the marks. 75 is the range (95 - 20), and 50 is the median.

1269. **D** — Rates: ads 20/200 = 10.0, email 6/40 = 15.0, social 12/100 = 12.0, search 7/50 = 14.0. Ordered by rate descending, email is first. Multiplying by 100.0 first forces real division; writing conversions / visits * 100 would use integer division and give 0 for every row.

1270. **A** — Step rates: 3,000/10,000 = 30%, 600/3,000 = 20%, 150/600 = 25%. Account-to-trial is the weakest step, so that is where to investigate first. Visitor-to-paid (1.5%) is the overall rate across all steps, not a single step.

1271. **B** — Percentage change = (129.8 - 118) / 118 = 11.8 / 118 = 10%. 11.8 is the change in index points, and 29.8% is the rise since the 2023 base, not since 2025.

1272. **D** — The discounted price is 80% of the original, so original = 6,400 / 0.8 = Rs 8,000. Adding 20% to 6,400 (giving 7,680) is wrong because the 20% was taken from the larger original price.

1273. **A** — Successive changes multiply: 1.10 x 1.20 = 1.32, a 32% total rise. Adding 10% and 20% ignores that the second rise applies to an already larger cost.

1274. **D** — Means: x = 3, y = 4. Deviation products sum to 6, sum of squared x-deviations = 10, sum of squared y-deviations = 6. r = 6 / sqrt(10 x 6) = 6 / 7.75 = 0.77, a fairly strong positive linear relationship. The sign is positive because y tends to rise with x.

1275. **A** — x-mean = 2.5, y-mean = 5.75. Sxy = 8.5 and Sxx = 5, so b = 1.7 and a = 5.75 - 1.7 x 2.5 = 1.5. Prediction for x = 6: 1.5 + 1.7 x 6 = 11.7. Adding the last step (+1 per month) gives 10, which ignores the fitted trend.

1276. **B** — sqrt(0.5 x 0.5 / 400) = sqrt(0.000625) = 0.025. Multiply by 1.96: 0.049, or about 4.9 percentage points. 2.5 is the standard error before multiplying by 1.96, and 9.8 is the full width of the interval.

1277. **D** — Sorted: 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 108, 110, 112, 115, 120, 130, 140, 180, 250. Rank = ceiling(0.9 x 20) = 18, and the 18th value is 140 ms. 180 is the 19th value and 250 is the maximum.

1278. **D** — Month 1: 1,000 x 0.6 = 600. Month 2: 600 x 0.8 = 480. Month 3: 480 x 0.8 = 384. Subtracting a flat 20% of the original 600 each month (giving 360) treats the decline as linear instead of compounding.

1279. **D** — A running total can only go up even if monthly downloads are falling. Plotting new downloads per period (or the month-on-month change) shows whether growth is really accelerating.

1280. **D** — An average from 8 reviews can swing a lot with one or two extra reviews; 2,300 reviews give a much more stable estimate. The 0.3-star gap does not show P is better.
1281. **C** — The question suggests the expected answer, which biases responses. A neutral wording such as 'How does the new timetable compare with the old one?' with a balanced scale gives a fairer measure.
1282. **D** — The first gap is 10 years and the rest are 1 to 4 years, yet they look identical, so the slopes are not comparable. A proportional time axis (or a bar chart with labelled gaps) fixes it.
1283. **B** — EV(B) = 0.6 x 100 + 0.4 x (-20) = 60 - 8 = 52 thousand, slightly above A's 50. Expected value is one input; a firm that cannot absorb a 20 thousand loss may still rationally choose A.
1284. **B** — Behaviour often changes temporarily when something is new or people know they are being watched. The fall back to baseline shows the lasting effect is near zero, which is why experiments should run long enough.
1285. **A** — People who opt in to betas are usually more engaged already, so the groups differ before the treatment. A randomised A/B test, where the app decides who sees the beta, would isolate the effect of the new version.
1286. **A** — Impossible values are data-quality errors, not real outliers. The right step is to trace them to the source, correct or exclude them, and document the decision so the analysis can be reproduced.
1287. **D** — Agents were optimising the single target by rushing calls. A balancing metric that captures quality (problems solved on the first call) keeps speed honest. Call length alone just creates a new number to game.
1288. **D** — Percentages from a tiny base can be huge while the real impact is small. Showing the absolute change (20 versus 2,000) alongside the relative change gives an honest picture.
1289. **A** — When the way data is collected changes, the series is no longer like-for-like. More reports can mean more crime, or that crimes previously unreported are now recorded. An independent measure is needed to separate the two.
1290. **C** — Dual-axis charts let the designer stretch either scale freely; moving one axis range would make the lines cross anywhere or never. Compare the series with separate panels or indexed values instead.
1291. **C** — Revenue = orders x average value: 1,000 x 2,000 = 2.0 million in March and 800 x 2,300 = 1.84 million in April, an 8% fall. A ratio metric can improve while the total it comes from gets worse.
1292. **C** — Choosing sites because of an unusually bad year means some of that year's excess was chance and would not repeat. A comparison group with similar 2025 counts but no cameras shows how much of the drop the cameras actually explain.
1293. **C** — Because everyone in the data scored 80 to 100, the test's full range is missing. Correlations computed on such a selected group are weakened, so the test may still separate good from poor candidates across all applicants.
1294. **B** — De Bono contrasted lateral thinking with 'vertical' step-by-step logic. Lateral thinking challenges assumptions and restructures the problem so that new solution paths appear.
1295. **B** — Divergent thinking spreads out to produce many options. Convergent thinking then narrows those options down to the single best one.
1296. **C** — Convergent thinking evaluates and narrows options to one answer. Generating the 30 ideas was the divergent phase.
1297. **D** — Alex Osborn's brainstorming rules ask the group to withhold criticism during generation, aim for quantity and build on others' ideas. Evaluation comes afterwards.
1298. **B** — SCAMPER stands for Substitute, Combine, Adapt, Modify, Put to another use, Eliminate and Reverse. Giving an existing object a new purpose is 'Put to another use'.
1299. **C** — Green is for creativity and alternatives. White is for facts, red for feelings, black for caution and risks, yellow for benefits and blue for managing the process.
1300. **C** — The black hat is critical judgement: risks, weaknesses and what could go wrong. The yellow hat would list benefits instead.
1301. **C** — Inversion flips the question, for example 'how could we make students hate this app?'. The answers expose hidden causes, and reversing each one gives a preventive action.
1302. **B** — Analogical thinking transfers a working mechanism from one field to another. Borrowing from nature is called biomimicry; the hook-and-loop fastener is the textbook example.
1303. **A** — Functional fixedness, studied in Duncker's candle problem, is seeing an object only in its usual role. Breaking it lets you see a coin as a flat metal edge.

1304. **C** — First-principles thinking strips away assumptions and analogies, keeps only the fundamental facts and reasons upward from them. It often reveals that an 'impossible' cost or limit is only a convention.

1305. **A** — The owner reframed the problem: the real pain was how the wait felt, not its length. A new problem statement opened a much cheaper solution.

1306. **B** — An MVP is the smallest version that lets you learn whether the idea works. Early feedback is cheap; discovering a wrong assumption after months of building is expensive.

1307. **B** — The d.school model starts by understanding users (Empathise), states the problem (Define), generates ideas (Ideate), builds (Prototype) and checks with users (Test). In practice the stages loop back.

1308. **D** — A mind map puts the topic in the centre and branches related ideas outward. It supports divergent thinking by making connections easy to see and extend.

1309. **B** — An invention is a new idea or device; innovation is turning a new or improved idea into something used that delivers value, such as a better process or service. Many innovations are not patented at all.

1310. **C** — There are 50 pairs, each summing to 101, so the total is 50 × 101 = 5050, which matches n(n + 1)/2. 10100 is 100 × 101, double-counting every pair.

1311. **D** — Instead of adding up rounds (32 + 16 + 8 + 4 + 2 + 1), notice that every team except the champion must lose exactly once. 64 - 1 = 63 matches.

1312. **A** — Worst case, the first three socks are one of each colour. The fourth must match one of them (pigeonhole principle): 3 colours + 1 = 4.

1313. **D** — The trick is to use a second property of the bulb, heat, as well as light. That gives three distinguishable states (on, off-warm, off-cold) from one visit.

1314. **D** — 1 + 2 + ... + 100 = 5050. The missing number is 5050 - 4977 = 73. This sum trick finds the gap in one pass with no sorting or searching.

1315. **C** — Writing states as (3 L jug, 5 L jug): fill the 5 L (0,5), pour into the 3 L (3,2), empty the 3 L (0,2), pour the 5 L into the 3 L (2,0), fill the 5 L (2,5), top up the 3 L from the 5 L (3,4). That is 6 steps, and a breadth-first search over all jug states confirms no shorter sequence exists.

1316. **D** — Each weighing has three outcomes (left heavy, right heavy, balanced), so k weighings separate up to 3^k cases. 3^1 = 3 < 8 but 3^2 = 9 ≥ 8. Weigh 3 vs 3: if balanced, weigh the remaining 2; otherwise weigh 1 vs 1 from the heavy group.

1317. **A** — Because burning is uneven, length tells you nothing about time. Lighting both ends halves the remaining time whatever the shape: A gives 30 minutes, and B's remaining 30 minutes becomes 15 when lit from both ends. 30 + 15 = 45.

1318. **D** — Colouring gives the insight that counting alone misses. Opposite corners share a colour, so 30 of one colour and 32 of the other remain. 31 dominoes would need 31 of each, so no tiling exists.

1319. **A** — A hash set gives average O(1) membership checks, so one pass of n insertions finds a duplicate in O(n) expected time. Sorting first (O(n log n)) is another valid alternative; nested-loop variants stay $O(n^2)$.

1320. **A** — P = [0, 3, 4, 8, 9, 14, 23, 25, 31]. Sum of A[2..5] = P[6] - P[2] = 23 - 4 = 19, which equals 4 + 1 + 5 + 9. Each query now costs O(1) instead of a loop.

1321. **A** — Each handshake is a pair of people, so count pairs: C(10, 2) = 10 × 9 / 2 = 45. 90 counts every handshake twice, once from each person.

1322. **D** — There are 12 months, so 12 people could all have different months. The 13th person must share a month with someone, by the pigeonhole principle.

1323. **B** — There are 12 × 2 = 24 possible answers. Each weighing has 3 outcomes, so k weighings distinguish at most 3^k cases: 3^2 = 9 < 24 ≤ 27 = 3^3. A known three-weighing scheme achieves it, so 3 is the minimum.

1324. **C** — The real problem is perceived waiting, not the 40-second wait itself. Uncertainty and boredom make waits feel longer, so information and distraction tackle the cause at low cost.

1325. **D** — Inversion is a means, not an end. Each way to fail points to a fix: slow sign-up becomes one-tap sign-up, English-only becomes Urdu support, and so on. Then the fixes are evaluated and ranked.

1326. **A** — Checks: 1 + 11 = 12 (move right), 1 + 8 = 9 (move left), 3 + 8 = 11 (move right), 3 + 6 = 9 (move left), 4 + 6 = 10. Found after 5 checks. The method is O(n) because each step discards one element.

1327. **A** — The puzzle states the other three rules. The extra assumption that lines must end on or inside the dot square is never stated; extending lines beyond it makes a four-line solution possible.

1328. **C** — Weighted scores: A = 0.5×4 + 0.3×9 + 0.2×3 = 5.3; B = 3.5 + 2.4 + 1.4 = 7.3; C = 4.5 + 1.5 + 1.4 = 7.4. C wins once cost-friendliness, the heaviest criterion, is weighted.
1329. **A** — Both are single-queue, multi-server systems: no server sits idle while work waits elsewhere, and one slow job does not trap everyone behind it. Analogy transfers the benefit directly.
1330. **B** — Send 1 and 2 (2), 1 returns (1), send 5 and 10 together (10), 2 returns (2), send 1 and 2 (2): 17. The insight is to pair the two slowest so the 10-minute crossing also covers the 5. A shortest-path search over all states confirms 17.
1331. **D** — Early criticism makes people self-censor, which kills divergent thinking. Splitting the session into a no-judgement generation phase and a later evaluation phase restores the flow of ideas.
1332. **B** — Early prototypes or MVPs test the riskiest assumption, here what students actually want, before months of work. A paper or clickable prototype in week one would have revealed the need.
1333. **C** — Reframing moves from a pre-chosen solution (a bigger server) to the underlying need (fast pages). That opens image compression, lazy loading and caching, which address the real cause.
1334. **C** — The white hat is neutral and objective: what data do we have, what do we need, and how do we get it? Red is feelings, yellow is benefits and blue manages the thinking process.
1335. **A** — The red hat allows feelings, hunches and intuition to be stated without justification. The black hat would need reasons (risks), white needs facts, and green generates new ideas.
1336. **B** — Eliminate asks what could be removed or simplified. Taking away the keyboard freed space for a larger screen. Combine would merge two things, Substitute swaps one part for another, and Put to another use finds a new purpose.
1337. **D** — Combine joins two products, features or ideas into one. The camera phone merges two previously separate devices; Adapt would borrow an idea from another context.
1338. **B** — 'How might we' signals that a solution is possible (how), that many answers exist (might) and that it is a shared effort (we). It opens divergent thinking without prescribing a fix.
1339. **A** — A random stimulus (for example the word 'umbrella') pulls thinking away from its usual paths, and forcing connections to the problem produces unexpected ideas. It is a generation tool, not a selection method.
1340. **C** — Writing in parallel lets everyone contribute at once and anonymously, so quieter members are heard and nobody waits for a turn. It still produces many ideas that are evaluated later.
1341. **C** — Adapt borrows an idea that works in another context and adjusts it to a new problem. Modify would change a feature's size or shape, and Reverse would flip the order or roles.
1342. **B** — Empathise gathers insight about users; Define synthesises that insight into a point-of-view statement such as 'Final-year students need X because Y'. Ideation then generates solutions to that statement.
1343. **A** — In Wallas's model (preparation, incubation, illumination, verification), stepping away lets the mind keep working in the background, and the insight that follows is illumination. Verification would be testing the fix.
1344. **D** — Morphological analysis builds a grid of attributes and options and explores combinations systematically, which can surface designs nobody would think of directly. A mind map is free association around a centre.
1345. **A** — The Einstellung effect is a fixation on a familiar method that blocks a better one, shown in Luchins's water-jar experiments. Functional fixedness is about seeing an object only in its usual role.
1346. **A** — Reverse/Rearrange asks what happens if order or roles are swapped. Here the customer takes over the cashier's task, which reduces queues and staff cost.
1347. **B** — A provocation such as 'Po: exams have no questions' is not meant to be true. Its job is to jolt thinking out of its normal pattern, then you 'move' from it towards a practical idea.
1348. **A** — Disruptive innovations enter at the low end or in new markets that incumbents ignore, then move upmarket. A steady improvement for existing customers is a sustaining innovation.
1349. **D** — Incremental innovation makes small improvements to an existing product or process. The other options created new categories or replaced the old technology, which is radical innovation.
1350. **C** — Groupthink is the pressure to agree that suppresses dissent and alternatives. A named devil's advocate, anonymous input and leaders speaking last all make it safer to disagree.
1351. **D** — Build-Measure-Learn loops produce evidence. When it shows the current hypothesis is wrong, a pivot changes one part of the plan while keeping what was learned; persevering keeps the plan.

1352. **A** — The Five Whys (from Toyota) moves from a symptom to an underlying cause, so a creative fix addresses the cause rather than patching the symptom. The number five is a guide, not a rule.

1353. **B** — Altshuller studied patents and found recurring solution patterns. TRIZ frames a problem as a contradiction (stronger but lighter) and points to principles such as segmentation that resolve it without a trade-off.

1354. **C** — Analogy is fast and often good, since it borrows what worked elsewhere, but it can inherit old assumptions. First-principles thinking strips a problem to fundamentals and recombines them, which is slower but can break those assumptions.

1355. **D** — Empathise-Define-Ideate-Prototype-Test is a guide, not a waterfall. Test results routinely reveal a wrong assumption, so teams loop back; that learning is the purpose of prototyping early.

1356. **B** — Waiting for a turn blocks idea production, and evaluation fear adds to it. That is why techniques such as brainwriting or individual ideation before group discussion often outperform open group brainstorming.

1357. **D** — The first diamond explores the problem widely (Discover) and then narrows it (Define); the second explores many solutions (Develop) and then narrows to one (Deliver). It alternates divergent and convergent thinking twice.

1358. **D** — 1 to 99 contains 50 odd numbers, so the sum is 50 squared = 2500. 4950 is the sum of every whole number from 1 to 99, odd and even, and 5000 doubles the right answer.

1359. **C** — 100 / 10 = 10 gaps, and a line with 10 gaps has 11 posts. The answer 10 confuses gaps with posts, a classic 'fencepost' error that also appears as off-by-one bugs in loops.

1360. **C** — Work backwards: if it doubles each day, the day before full coverage it was half. Splitting the time in half (day 15) wrongly treats doubling as linear growth.

1361. **C** — Each guess halves the remaining range, and 2^10 = 1,024 is at least 1,000, so 10 guesses always suffice. Guessing one by one could take 1,000.

1362. **A** — Good reframing targets the underlying need (quiet focus), which opens options like silent zones, booking pods or noise-cancelling areas. The others jump to one solution or a punishment.

1363. **D** — Put to another use finds a new purpose for something that already exists, here the order data, for kitchen planning. Removing is Eliminate, merging is Combine and enlarging is Magnify/Modify.

1364. **D** — Worst case: you take all 10 left gloves (5 black, 5 brown) and still have no pair. The 11th glove must be a right glove, which matches a left one of its colour. 3 works for socks, where left and right do not matter.

1365. **B** — The minute hand gains 330 degrees per hour on the hour hand, so it laps it every 12/11 hours: 12 / (12/11) = 11 overlaps in 12 hours. The overlap near 12:00 is shared, which is why it is not 12.

1366. **A** — sorted('listen') and sorted('silent') are both e, i, l, n, s, t; 'apple' and 'paple' both sort to a, e, l, p, p; 'rat' and 'car' differ. So it prints True True False.

1367. **C** — When Ali finishes, Bilal has run 90 m. Chand runs at 90% of Bilal's speed, so he has run 0.9 x 90 = 81 m. Ali wins by 19 m. Adding the two margins (20) ignores that the second is measured over Bilal's full 100 m.

1368. **A** — Work backwards: before a sale, eggs = 2 x (eggs after) + 1. From 0: 1, then 3, then 7. Check: 7 to 3 (sold 4), 3 to 1 (sold 2), 1 to 0 (sold 1).

1369. **D** — Total = 900. Numbers with no 7: first digit 8 choices (1-9 except 7), others 9 each, so 8 x 9 x 9 = 648. With at least one 7: 900 - 648 = 252. Counting directly risks double-counting numbers like 777.

1370. **B** — P(no six) = (5/6)^4 = 625/1296 = 0.482, so P(at least one) = 1 - 0.482 = 0.518, about 52%. Adding 4 x 1/6 = 67% double-counts rolls with more than one six; 48% is the chance of no six.

1371. **D** — Reverse all: [7,6,5,4,3,2,1]. Reverse the first 3: [5,6,7,...]. Reverse the rest: [1,2,3,4]. Result [5, 6, 7, 1, 2, 3, 4], a right rotation by 3 in O(n) time and O(1) extra space.

1372. **A** — x ^ x = 0 and x ^ 0 = x, and XOR is order-independent, so the pairs (1, 1) and (2, 2) cancel and the single 4 remains. It uses O(1) extra space.

1373. **B** — Windows of 3: 2+1+5 = 8, 1+5+1 = 7, 5+1+3 = 9, 1+3+2 = 6. The maximum is 9. Each step adds the new element and subtracts the one leaving, so it is O(n).

1374. **C** — calls(n) = calls(n-1) + calls(n-2) + 1 with calls(0) = calls(1) = 1, giving 177 for n = 10. Storing results (memoisation or a bottom-up loop) computes each value once, turning exponential time into O(n).

1375. **A** — Your first pick is right 1/3 of the time. In the other 2/3 the prize is behind one of the remaining doors, and the host removes the empty one, so switching wins 2/3 of the time. The '1/2' intuition ignores the host's knowledge.

1376. **C** — Number the bottles in binary (10 bits, since 2^10 = 1,024). Strip k gets a drop from every bottle whose bit k is 1. The pattern of positive strips spells out the poisoned bottle's number.
1377. **C** — Locker n is toggled once per divisor of n. Divisors come in pairs except for perfect squares, so only squares have an odd count and end open: 1, 4, 9, ..., 100, which is 10 lockers.
1378. **C** — With d drops the best plan covers d(d+1)/2 floors: drop from floor 14, then 27 (13 higher), 39, and so on, shrinking the step by one each time. 13 x 14 / 2 = 91 < 100 but 14 x 15 / 2 = 105 >= 100, so 14. Floors of 10 at a time give a worst case of 19.
1379. **A** — The team is anchored on its own industry. Analogies from unrelated fields give new starting points; more time or more competitor study keeps the same frame.
1380. **B** — A senior opinion stated first anchors the room and discourages alternatives. Gathering ideas independently first, and having leaders speak last, protects the range of ideas.
1381. **D** — The unstated assumption is that submission needs physical presence. Questioning it opens online forms or signed email. The other options accept the assumption and only tune the current process.
1382. **B** — The cause is one bottleneck point and a single peak. Pre-ordering moves payment off the counter and spreads collection across time. The others make waiting easier without shortening it.
1383. **C** — High-impact, low-effort ideas are 'quick wins' and usually go first because they deliver value fast. High-impact, high-effort ideas are major projects to plan next; low-impact, high-effort ones are usually dropped.
1384. **B** — Building without understanding users skips Empathise, so the team solved the wrong problem. They clearly prototyped and are now testing, which is how the gap was revealed.
1385. **A** — A good analogy shares the deep structure of the problem: a fast, coordinated, high-risk transfer with fixed roles. Pit-stop procedures (clear roles, checklists, rehearsal) map directly onto patient handover.
1386. **B** — Constraints push towards recombining existing free resources. A form plus a shared sheet gives instant, countable sign-ups with no cost or build time; the others break the budget or deadline, or create manual counting work.
1387. **D** — Rephrasing 'do two numbers differ by k?' as 'for each x, is x + k present?' turns pair-checking into set lookups, O(n) expected instead of O(n^2). (For k = 0 you would need to check for duplicates instead.)
1388. **D** — A tetrahedron has four equilateral triangular faces and exactly six edges, so building upwards solves it. The blocking assumption is the two-dimensional frame, a classic example of self-imposed limits.
1389. **D** — Floyd's tortoise-and-hare: in a cycle the fast pointer gains one node per step and must meet the slow one; without a cycle it reaches the end. Only two pointers are stored.
1390. **C** — In the magic square 2 7 6 / 9 5 1 / 4 3 8, the eight lines are exactly the eight triples from 1 to 9 that sum to 15. Changing the representation turns a hard arithmetic search into a familiar spatial game.
1391. **A** — At 7 min the second run of the 4 has 1 min left. Flipping the 7 then, it has run 1 minute when the 4 empties at 8; flipping it back lets exactly that 1 minute run out, ending at 9.
1392. **B** — The data says acquisition works but the product as built does not keep users, except one feature. A zoom-in pivot tests that signal cheaply. More features or ads ignore the retention evidence.
1393. **B** — Five heats (races 1-5) rank each group. Race 6 pits the five heat winners; call the order A1 > B1 > C1 > D1 > E1. A1 is the fastest overall. Places 2 and 3 can only be A2, A3, B1, B2 or C1, and race 7 between those five settles them.
1394. **C** — Tasks on the longest dependency chain have no spare time, so any delay to them delays the whole project. Its length is the earliest possible finish time.
1395. **A** — Slack = latest start - earliest start. Tasks on the critical path have zero slack; tasks off it can slip by their slack without moving the finish date.
1396. **B** — Two courses sharing a student cannot share a slot, exactly like adjacent vertices that cannot share a colour. The minimum number of slots is the graph's chromatic number.
1397. **C** — Earliest finish time leaves the most room for the remaining talks, and an exchange argument (CLRS, activity selection) proves it optimal. Shortest-first and earliest-start can both be beaten by counterexamples.
1398. **D** — Rooms are only needed for meetings that overlap. Sweeping through start and end times and tracking the peak number of meetings in progress gives the minimum room count.
1399. **B** — Output can never exceed the slowest stage's capacity. Improving any other stage adds no throughput until the bottleneck is improved.

1400. **D** — Below break-even the business makes a loss; above it, a profit. Break-even units = fixed cost / (price - variable cost per unit).
1401. **B** — The objective is what you optimise, here cost. A constraint is a condition every acceptable solution must meet, here the Friday deadline.
1402. **A** — A Gantt chart lists tasks down the side and time across the top. Each task is a bar, so overlaps, sequence and progress are visible at a glance.
1403. **A** — Finish-to-start (the most common type) means the successor can begin only once the predecessor has finished.
1404. **D** — The ratio-greedy rule is optimal for the fractional knapsack. For the 0/1 knapsack it can leave budget unused or choose a worse combination, so dynamic programming or search is needed.
1405. **A** — Urgent and important: do it now. Important but not urgent: schedule it. Urgent but not important: delegate it. Neither: drop it.
1406. **C** — Crashing trades money for time: overtime or extra staff shorten critical tasks. Overlapping tasks without extra resources is a different technique called fast-tracking.
1407. **B** — Newcomers must be trained by existing staff, and communication paths grow roughly as n(n - 1)/2. Brooks himself called the law an outrageous simplification, but it holds for most late projects.
1408. **A** — Little's law says L = λW for any stable system. It is used for cafés, call centres and servers alike: if you know two of the quantities you get the third.
1409. **C** — Project duration equals the length of the critical path. Shortening a non-critical task only adds slack to it; the longest chain, and so the finish date, does not change.
1410. **C** — Rates add: 1/6 + 1/3 = 1/2 of the job per hour, so together they take 2 hours. 4.5 hours is the average of the two times, which is not how joint work combines.
1411. **C** — They coincide every LCM(12, 18) = 36 minutes, so next at 8:36. 8:06 uses the difference of the intervals, which has no meaning here.
1412. **D** — Earliest finishes: A = 3, B = 5, C = 7, D = max(5, 7) + 1 = 8. The critical path is A → C → D. 10 days wrongly adds every task as if nothing ran in parallel.
1413. **B** — Earliest-finish greedy: 9-10, then 10-11, then a talk ending at 13 (11-13 or 12-13, a tie), then 13-15. 14-16 clashes with 13-15. That gives 4, and checking every subset confirms 4 is the maximum.
1414. **A** — At 10:00-10:30 the first three meetings all run at once, so at least 3 rooms are needed. Later, 11:00-12:00 reuses the room freed at 11:00 and 11:30-12:30 reuses the one freed at 11:30, so 3 is enough.
1415. **A** — Each report contributes 500 - 300 = 200 rupees towards fixed costs. 60,000 / 200 = 300 reports. 120 divides by the full price, ignoring the variable cost.
1416. **A** — L = λW = 30 per hour × (10/60) hour = 5. The units must match: 30 × 10 = 300 mixes hours and minutes.
1417. **B** — Greedy by profit with latest free slot: J1 (100) on day 2, J3 (27) on day 1, J4 and J2 need day 1 (taken), J5 (15) on day 3. Total 142; checking every feasible subset confirms it. 167 ignores that only two jobs can finish by day 2.
1418. **C** — Net rate = 1/4 - 1/6 = 1/12 of the tank per hour, so it takes 12 hours. 2.4 hours would be correct only if both pipes were filling.
1419. **D** — Feasible pairs: P2 + P3 costs 10 for 19; P1 + P2 costs 9 for 17; P3 + P4 costs 9 for 17. P1 + P3 costs 11, over budget. Ratio-greedy picks P1 then P2 for only 17, showing why 0/1 knapsack needs a full search.
1420. **A** — UTC+5 is five hours ahead of UTC, so subtract 5 hours: 14:00 - 5 = 09:00 UTC. Adding the offset (19:00) goes the wrong way.
1421. **D** — Total weight is 20 tonnes, so at least 2 vans. Two vans suffice: 6 + 4 = 10 and 5 + 3 + 2 = 10. Pairing 6 + 3 and 5 + 4 leaves no room for the 2, which is how careless packing ends up with 3 vans.
1422. **D** — Home → B (2) → A (1) = 3, A → C (5) = 8, C → Uni (3) = 11. Alternatives: Home → A → Uni = 14, Home → B → C → Uni = 13, Home → A → C → Uni = 12. Dijkstra's algorithm confirms 11.
1423. **C** — Throughput equals the bottleneck rate, stage 2 at 25 per hour. 25 × 8 = 200. 320 uses stage 1's rate, and 760 adds all three stages' rates together.
1424. **D** — Path A-B-E = 9 days; path A-C-D-E = 10 days, the critical path. E can start on day 7; B finishes on day 6, so B can slip 1 day. Every task on A-C-D-E has zero slack.

1425. **B** — Only the 10-day path A-B-D sets the duration, so you must cut a task on it. B is the cheapest critical task at 300. Afterwards both paths take 9 days. C is cheaper but sits on the shorter path, so crashing it saves nothing.
1426. **D** — The queue is caused by synchronised demand, not by too little capacity across the day. Staggering breaks spreads arrivals so existing capacity copes.
1427. **B** — Greedy takes 4, leaving 2, then 1 + 1: three coins. Dynamic programming finds 3 + 3: two coins. Greedy is only safe for coin systems with special structure, such as 1, 2, 5, 10.
1428. **D** — Ramp-up and the growth in communication paths (from n(n - 1)/2) often outweigh the extra hands in the short term. Reducing scope or adding help to well-separated tasks is usually safer.
1429. **C** — Shortest processing time first: completions 1, 3, 8, 16, average 28 / 4 = 7. Running short jobs first means fewer jobs wait behind long ones. Checking all 24 orders confirms 7 is the minimum.
1430. **C** — Take the goat over (1), return (2), take the wolf (3), bring the goat back (4), take the cabbage (5), return (6), take the goat (7). The key move is carrying the goat back. A breadth-first search over all states confirms 7.
1431. **B** — A, B and C all clash with each other (a triangle), so at least 3 slots are needed. 3 works: slot 1 = A and D, slot 2 = B and E, slot 3 = C. Checking every 2-colouring confirms 2 slots are impossible.
1432. **A** — Conflicting constraints are a decision for the stakeholder. Stating the trade-off clearly, with options such as launching on Monday with cash-on-arrival and adding payments later, lets the client choose with full information.
1433. **B** — Own server: 300 + 5m; cloud: 15m. They are equal when 300 = 10m, at month 30. At 36 months: own = 300 + 180 = 480; cloud = 15 × 36 = 540. A real decision would also weigh maintenance effort and risk.
1434. **C** — A feasible solution meets all constraints. Among the feasible solutions, the one with the best objective value is the optimal solution. An option that breaks a constraint is infeasible, however good it looks.
1435. **D** — Example: talks 9-12 and 12-15 do not overlap, but a short talk 11-13 overlaps both. Shortest-first picks 11-13 and then fits neither, giving 1 talk instead of 2. Only earliest-finish-first is proven optimal.
1436. **D** — The Pareto principle observes that roughly 80% of effects come from about 20% of causes. A Pareto chart sorts causes by frequency so a team can tackle the few that matter most.
1437. **C** — Total size divided by capacity, rounded up, is a lower bound for bin packing. Loads that cannot be split may still force extra vans, so the true minimum can be higher, never lower.
1438. **A** — A line's throughput equals its bottleneck's capacity (theory of constraints). Improving a non-bottleneck stage leaves throughput unchanged; only raising the bottleneck's capacity helps.
1439. **B** — A milestone has no duration; it marks a significant event such as 'design approved' or 'system live'. It helps track progress without adding work to the schedule.
1440. **C** — PERT's three-point (beta) estimate weights the most likely value four times: te = (O + 4M + P) / 6. (P - O) / 6 is the standard deviation, not the expected time.
1441. **C** — Total float = LS - ES = LF - EF. It is how long the task can slip without delaying the project. EF - ES is just the task's duration, and project duration minus duration ignores the task's position in the network.
1442. **B** — Free float protects the next tasks' earliest starts; total float protects only the project end date. Free float is therefore never larger than total float, and both are zero on the critical path.
1443. **C** — The triple constraint says scope, time and cost are linked: changing one, such as adding scope, affects at least one of the others. Quality is often drawn in the middle of the triangle.
1444. **D** — Resource levelling moves tasks, often using their float, so demand for a resource never exceeds what is available. If float runs out, the end date moves. Adding staff is crashing; overlapping tasks is fast-tracking.
1445. **A** — Fast-tracking overlaps activities, for example starting coding before design is fully signed off. It shortens the schedule without direct extra cost but raises the risk of rework. Adding resources at a cost is crashing.
1446. **A** — SMART goals are Specific, Measurable, Achievable, Relevant and Time-bound. 'Measurable' means progress can be checked against a number or clear criterion, such as 'reduce queue time to 5 minutes'.
1447. **A** — Visiting every location once and returning is the travelling salesman problem. No polynomial-time exact algorithm is known, so practical tools use heuristics such as nearest neighbour or 2-opt. Dijkstra finds one shortest route between two points, not a tour.
1448. **D** — Connecting every vertex at minimum total edge weight, with no need for a cycle, is a minimum spanning tree, solved by Kruskal's or Prim's algorithm. A shortest-path tree minimises distance from one source, which can use more total cable.

1449. **C** — Dijkstra's algorithm assumes non-negative edge weights; a negative edge can make an already-finalised distance wrong. Cycles and fractional weights are fine. Bellman-Ford handles negative weights.

1450. **B** — One-to-one matching of n agents to n tasks at minimum total cost is the assignment problem. The Hungarian algorithm solves it in polynomial time. Picking each person's cheapest duty in turn can fail.

1451. **A** — Opportunity cost is the value of the best alternative given up. The forgone internship pay is a real cost of choosing the course, even though no money changes hands.

1452. **D** — Money already spent cannot be recovered whatever is chosen next, so it is a sunk cost. Rational decisions compare only future costs and benefits. Continuing 'so it isn't wasted' is the sunk-cost fallacy.

1453. **D** — Earliest deadline first (Jackson's rule) minimises the maximum lateness on a single machine. Shortest processing time first minimises average completion time, which is a different objective.

1454. **A** — A weighted decision matrix rates options on several criteria, multiplies each score by the criterion's weight and sums the results. It makes trade-offs explicit when no option is best on everything.

1455. **D** — If arrivals come as fast as or faster than the server can serve ($\rho \geq 1$), the queue grows without limit. Stability needs $\rho = \lambda/\mu < 1$; even at $\lambda = \mu$ random variation makes the queue drift upward.

1456. **A** — Rates add: 1/4 + 1/6 + 1/12 = 3/12 + 2/12 + 1/12 = 6/12 = 1/2 job per hour, so the job takes 2 hours. Averaging the times (7.3 h) is the common mistake.

1457. **B** — Break-even units = fixed cost / (price - variable cost) = 48,000 / (200 - 120) = 48,000 / 80 = 600 meals. Dividing by the price (240) ignores the cost of each meal.

1458. **B** — Stage capacities are 2 × 15 = 30, 20 and 3 × 10 = 30 per hour, so stage 2 is the bottleneck at 20 per hour: 20 × 6 = 120. Using the slowest single machine (10 × 6 = 60) forgets parallel machines.

1459. **D** — te = (O + 4M + P) / 6 = (4 + 24 + 14) / 6 = 42 / 6 = 7 days. The simple average (4 + 6 + 14) / 3 = 8 gives the pessimistic value too much weight.

1460. **D** — Paths: A-B-D-E = 4 + 3 + 2 + 5 = 14 days; A-C-E = 4 + 6 + 5 = 15 days. The longest path, 15 days, is the critical path. Adding all durations (20) ignores that B-D and C run in parallel.

1461. **D** — Sweep the times: at 9:15 three meetings run (8:30-10:00, 9:00-9:30, 9:15-11:00), and at 10:15 three run again (9:15-11:00, 10:00-10:30, 10:15-11:15). Four never overlap, so 3 rooms suffice.

1462. **B** — Earliest finish first: 8-10 (or 9-10), then 10-12, 12-14, 14-16, giving 4. Five cannot fit: the talks lie within 8:00-16:00 (8 hours), only one talk is 1 hour long and the rest are 2 hours, so five talks need at least 1 + 4 × 2 = 9 hours.

1463. **D** — Little's law $L = \lambda W$ gives $W = L / \lambda = 12 / 4 = 3$ minutes. Multiplying (48) or adding (16) the figures does not follow the law.

1464. **A** — $L = \lambda \times W$ = 2 per minute × 15 minutes = 30 visitors. The units match (per minute × minutes), so no conversion is needed.

1465. **C** — B + C + D costs 6 + 5 + 4 = 15 for 11 + 9 + 8 = 28. Taking the biggest benefit first (A, then B) gives only 27, and A + B + E costs 18, over budget. Whole-item choices need a full comparison.

1466. **C** — Order 2, 3, 6, 9: the jobs wait 0, 2, 5 and 11 minutes, total 18, average 4.5. The average completion time would be (2 + 5 + 11 + 20) / 4 = 9.5, a different measure.

1467. **C** — PKT is 5 hours ahead of UTC, so add 5 hours: 23:59 + 5 h = 04:59, which rolls over to the next day, 11 March. Subtracting gives 18:59, the wrong direction.

1468. **A** — Combined rate = 40 + 60 = 100 pages per minute, so 1,500 / 100 = 15 minutes. Using one printer gives 25 minutes (Y alone) or 37.5 minutes (X alone).

1469. **A** — Weighted scores: P = 4.5 + 1.8 + 1.0 = 7.3; Q = 2.5 + 2.7 + 1.8 = 7.0; R = 4.0 + 2.1 + 1.4 = 7.5. R wins. Q has the highest unweighted total (23), but that ignores the weights.

1470. **B** — Total is 26 tonnes, so at least $\lceil 26/10 \rceil = 3$ vans. Three work: 7 + 3, 5 + 4 + 1, 4 + 2. Two vans hold only 20 tonnes, so 2 is impossible.

1471. **D** — The critical path A-C-D-E takes 11 days, so E must start by day 10. B (4 days) must finish by 10, so its latest start is day 6. Its earliest start is day 3, giving 3 days of float.

1472. **A** — Kruskal's algorithm takes the cheapest edges that do not form a cycle: B-C 2, A-C 3, D-E 3, then A-B 4 is skipped (cycle), then B-D 5 joins the two groups. Total 2 + 3 + 3 + 5 = 13. 30 is the sum of every edge.

1473. **C** — EDF order J4, J1, J2, J3 finishes at 1, 4, 6 and 10 against deadlines 3, 4, 6 and 9, so only J3 is late, by 1 day. Shortest-first (J4, J2, J1, J3) would finish J1 at day 6, 2 days late.

1474. **B** — 250 / 40 = 6.25, and a fraction of a session is impossible, so round up to 7. Six sessions seat only 240 students.

1475. **A** — Utilisation ρ = λ/μ = 8/10 = 0.8, so L = 0.8 / 0.2 = 4. The value 0.8 is utilisation, not the number in the system; L grows sharply as ρ approaches 1.

1476. **C** — Check all 6 assignments: W1-J2, W2-J1, W3-J3 costs 2 + 6 + 1 = 9, the minimum. Giving each worker in turn their cheapest free job (W1-J2, W2-J3, W3-J1) costs 10. The total 6 takes every worker's cheapest job, but W2 and W3 would both get J3, which is not allowed.

1477. **B** — The 9 a.m. deadline and the 300 copies are constraints, so P (late) and S (too few) are infeasible. Of the feasible shops, Q is cheaper than R, so Q is optimal.

1478. **B** — Earliest start picks 9-17, which blocks every other talk, giving 1. Picking 10-11, 11-12 and 12-13 gives 3. This is why earliest start is not a safe rule; earliest finish is.

1479. **B** — Nearest neighbour goes S-A (1), A-C (2), C-B (9), B-S (2) = 14, because it is forced into the long C-B leg. S-B-A-C-S = 2 + 4 + 2 + 3 = 11 is the best of the three distinct tours.

1480. **D** — The pattern points to damp paper from overnight storage. Moving the paper removes the cause. The other options manage the symptom and keep costing time or money.

1481. **C** — Path 1 drops to 12 days, but path 2 still takes 14, so it becomes critical and the project takes 14 days. Crashing beyond the gap to the next-longest path wastes money unless both paths are shortened.

1482. **D** — The measurement method changed at the same time as the numbers. Easier reporting alone can raise counts, so the change in data collection must be ruled out before blaming the hostels.

1483. **B** — Rooms needed = the peak number of meetings at one moment. Y and Z do not overlap, so the peak is 2 (X with Y, or X with Z). Y and Z can share a room.

1484. **D** — Output equals the slowest stage. After the upgrade the stages are 50, 45 and 35, so output rises only to 35. The next bottleneck limits the gain from any upgrade.

1485. **C** — Before: 90,000 / (400 - 250) = 600. After: 90,000 / (450 - 250) = 450. The margin per student rises from 150 to 200 (33%), so break-even drops by a quarter, more than the 12.5% fee rise.

1486. **A** — Only Venue 1 meets all three constraints. Venue 2 is not free on Friday, Venue 3 is too small and Venue 4 is over budget, however attractive their other features are.

1487. **A** — The 2 million is sunk under both choices, so ignore it. Finishing gives 1 - 1.5 = -0.5 million; the new system gives 1.8 - 1 = +0.8 million. Only future costs and benefits matter.

1488. **B** — London's 9:00-17:00 is 14:00-22:00 in Lahore. Overlap with Lahore's 9:00-17:00 is 14:00-17:00, which is 3 hours (9:00-12:00 London time).

1489. **A** — Expected cost = quote + probability × penalty. A: 100,000 + 0.2 × 50,000 = 110,000. B: 105,000 + 0.05 × 50,000 = 107,500. B is cheaper on average despite the higher quote.

1490. **C** — There are 100 faults. Paper jam (45) plus toner (25) is 70, exactly 70%. Paper jam alone is 45%. This is a Pareto analysis: fix the biggest categories first.

1491. **C** — A vague complaint must first be turned into measurable facts. Measuring shows whether the problem is one page, peak load or the network, so the fix targets the real cause.

1492. **D** — A functional requirement states something the system must do, a behaviour or service. Load time, availability and accessibility describe how well it does things, so they are non-functional (quality) requirements.

1493. **C** — Non-functional requirements set quality attributes such as performance, security, reliability and usability. The other three describe features the system must provide.

1494. **D** — User stories put the user, the goal and the reason in one sentence. 'Given/When/Then' is the format for acceptance criteria, and 'shall' statements are the traditional SRS style.

1495. **B** — Given/When/Then (used in behaviour-driven development, for example Gherkin) states a precondition, a trigger and an observable result, so each criterion can be turned straight into a test.

1496. **B** — Clear code shows what happens; comments add what the code cannot say, such as why a workaround exists or which rule a number comes from. Comments that repeat the code add noise and go stale.

1497. **B** — PEP 257 defines a docstring as the first statement string in a module, function, class or method. Python stores it in the object's __doc__ attribute, and tools like help() display it.

1498. **D** — A good requirement is verifiable: a tester must be able to say pass or fail. Rewriting it with numbers, such as '90% of screens load within 2 seconds on 4G', makes it testable.

1499. **D** — An SRS (described in standards such as IEEE 830 and its successor ISO/IEC/IEEE 29148) captures what is to be built. Designers build from it, testers test against it and clients accept against it.
1500. **A** — A precondition is what must be true before the call, and the caller is responsible for it. A postcondition is what the function promises afterwards. Together they form the function's contract.
1501. **C** — A README is the front door of a project. A new reader needs its purpose, how to set it up and run it, and a quick example; links to licence and contribution notes are also common.
1502. **B** — SemVer: MAJOR for incompatible API changes, MINOR for backward-compatible new features, PATCH for backward-compatible bug fixes.
1503. **A** — A developer must reproduce a bug before fixing it. Steps, expected versus actual behaviour, and environment details (browser, version) make that possible.
1504. **A** — A sequence diagram puts participants across the top, with lifelines down the page and messages as arrows in time order. Class diagrams show static structure, and use case diagrams show actors and goals.
1505. **C** — Traceability shows that every requirement is designed, built and tested, and it lets you see which tests and modules are affected when a requirement changes.
1506. **B** — An ADR is a short document that explains why a decision was made, for example 'use PostgreSQL rather than MongoDB', so future developers understand it instead of guessing.
1507. **B** — RFC 2119 defines MUST/SHALL as absolute requirements and SHOULD as recommended but open to justified exceptions. Mixing them carelessly leaves testers unsure what is required.
1508. **A** — 1.17 is a 'magic number'. A named constant explains its meaning, keeps the rate in one place for future changes, and a comment can say where the rate comes from. Restating the arithmetic adds nothing.
1509. **D** — Only the first version states a measurable task, a time limit and a pass threshold, so a usability test can pass or fail it. The others still depend on opinion.
1510. **B** — The list comprehension keeps only n > 0, so zeros and negatives are excluded; for example [0, 2, 4] gives 3, not 2. An empty result returns 0 rather than raising an error.
1511. **A** — A good commit summary says what changed and why in a short imperative line, so anyone scanning the history understands it. File names and vague words do not.
1512. **C** — The correct criterion fixes the starting state (2 failures), the trigger (the 3rd wrong password) and an observable result. The others are vague or untestable.
1513. **B** — range(5) gives five attempts, so the comment is wrong or the code is. A stale comment misleads every future reader, so check the intended behaviour and fix whichever side is wrong.
1514. **B** — A backward-compatible feature increases MINOR and resets PATCH to 0: 2.4.7 → 2.5.0. Keeping the patch digit (2.5.7) breaks the reset rule.
1515. **A** — The release is versioned by its most significant change. A new compatible feature means a MINOR bump, which resets PATCH: 3.3.0. Bug fixes in the same release do not add to the patch number.
1516. **D** — Good explanations match the audience: state the impact (new date), the cause in plain words, and the options. Jargon, vague updates and blame do not help the manager decide.
1517. **A** — The function scans left to right and returns the first matching index; -1 is returned only after the whole list has been checked. Putting RETURN -1 in an ELSE would stop after the first mismatch, and scanning backwards would return the last match.
1518. **D** — Callers need to know the error behaviour to handle bad input. Parameters, return value, errors and an example are the core of function-level API documentation.
1519. **A** — Binary search discards half the range based on a[mid] < x, which is only valid if a is ascending. On [7, 5, 3, 1] it fails to find 7 and returns -1. If x is missing, it correctly returns -1, so presence is not required.
1520. **C** — The first report gives environment, steps, expected and actual results and how often it happens, so the developer can reproduce it at once. The others force a long back-and-forth.
1521. **B** — For each i the inner loop runs over all later j, giving n(n - 1)/2 comparisons, 4,950 for n = 100. That is quadratic. A wrong complexity claim misleads anyone choosing this function for large inputs.
1522. **D** — 'Sorted by date' does not say ascending or descending, and 'last' could mean most recent or final in some list. Resolving that ambiguity changes what is built and tested; the other questions do not affect the requirement.

1523. **D** — Requirements must be consistent. Only stakeholders can decide, for example 'PDF by default, Excel for finance', and the SRS must be updated so design and testing follow the agreed version.
1524. **A** — For n = 1, range(2, 1) is empty, so the loop never runs and the function returns True, but 1 is not prime. The other three give correct answers: 2 and 13 are prime, 9 is not.
1525. **A** — Comments that restate code add clutter and drift out of date. Clear names carry the 'what'; comments should carry the 'why'.
1526. **C** — list.sort() sorts in place, so a is changed, and the function returns that same object. The fix is return sorted(items), which builds a new list, or a corrected docstring.
1527. **C** — Restating confirms you understood the task, a worked example lets the listener follow, and complexity plus edge cases show you have checked the limits. This mirrors how good written solution documents are organised.
1528. **C** — An untested requirement has no evidence it is met. For a security requirement like hashing, a silent failure could expose user passwords, so a test should be added.
1529. **C** — Release notes for users describe effects they notice, in plain language. Internal names like AuthService mean nothing to them, and 'various improvements' tells them nothing.
1530. **C** — A sequence diagram shows messages between participants (browser, server, database) in the order they happen. Class and ER diagrams show static structure, not the order of interactions.
1531. **D** — Stating assumptions and limits tells future users when the solution is safe to use and when it will fail, for example on a 10 GB file or duplicate IDs. Hidden assumptions are a common source of production bugs.
1532. **D** — A use case diagram shows actors (users or external systems) and the use cases they take part in, giving a high-level view of what the system offers. Classes belong in a class diagram, message order in a sequence diagram, and servers in a deployment diagram.
1533. **C** — A class diagram is UML's static structure diagram: classes, their attributes and operations, and relationships such as association, inheritance and composition.
1534. **C** — Flowcharts use a diamond for a decision with two or more exits, a rectangle for a process step, a parallelogram for input/output and an oval (terminator) for start and end.
1535. **C** — The parallelogram is the input/output symbol. A rectangle is for processing, a diamond for decisions and an oval for the start and end terminators.
1536. **B** — A pre-release version has lower precedence than the normal version it leads to, so 1.0.0-alpha and 1.0.0-beta come before 1.0.0. Pre-release labels compare by ASCII order, so alpha comes before beta.
1537. **A** — SemVer says major version zero is for initial development, so anything may change. Version 1.0.0 defines the first stable public API, after which the MAJOR/MINOR/PATCH rules apply in full.
1538. **B** — Conventional Commits maps 'feat:' (a new feature) to MINOR and 'fix:' (a bug fix) to PATCH. A breaking change, marked with '!' or a BREAKING CHANGE footer, maps to MAJOR. 'docs:' and 'chore:' do not normally trigger a release.
1539. **C** — The convention is to write the subject as a command, completing the sentence 'If applied, this commit will ...'. Git's own generated messages, such as 'Merge branch', use the same style.
1540. **D** — A changelog (for example in the 'Keep a Changelog' style) summarises notable changes per version for people, not machines. A raw commit log is too detailed and full of internal noise to serve that purpose.
1541. **C** — A loop invariant is a property that holds before the first iteration and is preserved by each iteration, such as 'total equals the sum of the first i items'. Stating it documents why the loop is correct.
1542. **C** — Per PEP 257, a docstring becomes the object's __doc__ attribute, so tools such as help(), IDEs and documentation generators can show it. Comments starting with # are removed when the code is parsed.
1543. **B** — Javadoc uses @throws (or its synonym @exception) to name an exception and the condition that causes it. @param describes a parameter, @return the return value and @see a cross-reference.
1544. **C** — Type hints (PEP 484) are annotations. They help readers, IDEs and static checkers such as mypy, but CPython does not check them at runtime, so grade('80') still runs unless the code itself checks.
1545. **D** — MoSCoW sorts requirements into Must have, Should have, Could have and Won't have this time. It tells stakeholders clearly what will be delivered first and what is deferred.
1546. **A** — INVEST stands for Independent, Negotiable, Valuable, Estimable, Small and Testable. A testable story has clear acceptance criteria, so the team knows when it is done.

1547. **A** — The Definition of Done (for example 'code reviewed, tests pass, docs updated') applies to every backlog item. Acceptance criteria state what this particular story must do. An item is complete only when it meets both.

1548. **C** — Diátaxis separates four kinds of docs: tutorials (learning by doing), how-to guides (solving a specific task), reference (accurate technical description) and explanation (background and reasons). Mixing them makes documents harder to use.

1549. **A** — Self-documenting code uses descriptive names such as overdue_fines and small, focused functions so the 'what' is obvious. Comments are then saved for the 'why'.

1550. **A** — A data dictionary gives one agreed definition for each data element (for example 'CNIC: 13 digits, required'). It prevents different teams interpreting the same field in different ways.

1551. **D** — An ER diagram models entities such as Student and Course, their attributes and relationships such as 'enrols in', with cardinalities. It is the standard way to document a database's logical design.

1552. **B** — A state machine diagram shows states (for example Placed, Paid, Shipped) and the event-triggered transitions between them. It suits objects whose behaviour depends on their current state.

1553. **A** — In Sommerville's terms, user requirements state in plain language (plus diagrams) what services the system provides and its constraints. System requirements expand them into detailed functions, services and constraints, often forming the contract.

1554. **B** — A minimal reproducible example removes everything unrelated while still triggering the bug. It lets others run it immediately, and often helps the author find the cause while trimming it down.

1555. **D** — Validation asks whether these are the right requirements: are they consistent, complete, realistic and verifiable, and do they reflect real needs? Reviews and prototypes are common techniques. Interviews are elicitation.

1556. **A** — A compatible bug fix bumps PATCH: 9 becomes 10, giving 4.1.10. Version parts are separate integers, so there is no carry into MINOR.

1557. **C** — SemVer says 1.0.0 defines the first stable public API. Moving to 0.10.0 would still signal initial development.

1558. **D** — Each part is compared as a number, so MINOR 10 beats 9 and 2. 1.10.0-rc.1 is a pre-release of 1.10.0, so it ranks below 1.10.0 itself. Comparing as text would wrongly put 1.9.9 first.

1559. **C** — Only the first gives a number, a measurement period and the exclusions, so a tester can say pass or fail. 'Highly' and 'as possible' cannot be measured.

1560. **A** — A user story names a real user role, a goal and the benefit. The database statement is a design decision, 'As a system' has no real user, and the last option has no role, goal or benefit.

1561. **A** — Boundary values are where mistakes such as > instead of >= appear, so 5,000 exactly is the key case. 10,000 is a normal case, and the other two have no checkable outcome for the rule.

1562. **B** — max(lo, min(x, hi)) caps x at hi and then raises it to at least lo, so it clamps x into [lo, hi]. It never raises an error, and if lo > hi the result is meaningless, so the precondition lo <= hi is worth stating.

1563. **A** — split(" ") splits on every single space, so a trailing space produces an empty final piece: last_word("hello world ") returns "" instead of "world". Using s.split() with no argument would fix it.

1564. **C** — The useful comment explains why the code waits and grows the delay: a rule of the external API. The others repeat what the code already says.

1565. **B** — A good subject is a short imperative summary of what the change does. 'changes' and 'Fixed stuff' say nothing, and listing file names repeats what the diff already shows.

1566. **A** — Severity measures technical impact: a typo breaks no function, so it is low. Priority measures how urgently it must be fixed: the demo makes it urgent, so it is high.

1567. **C** — n is halved on every pass, so the loop runs about log2(n) times: halvings(1024) returns 10. The loop is not O(1), because the count grows with n, and it is far below O(n).

1568. **B** — name.split() with no argument splits on any run of whitespace, so the double space is ignored and the words are 'ada' and 'lovelace'. Their first letters upper-cased and joined give 'AL'; doctest shows the string with quotes.

1569. **B** — 2.675 cannot be stored exactly in binary floating point; it is slightly less than 2.675, so round(2.675, 2) gives 2.67. Documenting 2.68 would make the doctest fail and mislead readers.

1570. **B** — A good incident update gives the impact and time, the cause in plain words, the current status and the prevention. The vague options give no facts, and the jargon-heavy one is aimed at the wrong reader.

1571. **B** — Release notes for end users describe the effect they will notice. Endpoint names, refactoring and pull request numbers mean nothing to students.

1572. **D** — The good version fixes the file size, the time limit, when the clock starts and the test condition. The others still rely on words such as 'quickly', 'efficient' and 'too long' that cannot be measured.

1573. **D** — R3 has no linked test, so it could be missing or broken without anyone noticing. T4 is linked to nothing, so it may test an undocumented feature or the matrix is incomplete. The matrix does not say T4 covers R3; that must be checked and recorded.

1574. **D** — An order's life is a set of states with event-driven transitions (payment received, item dispatched, order cancelled), which is exactly what a state machine diagram shows.

1575. **D** — Removing a public function breaks backward compatibility, so MAJOR goes up and MINOR and PATCH reset to 0: 3.0.0. Keeping 3.3.4 forgets the reset.

1576. **C** — The guard rejects amount <= 0 and amount > balance with ValueError, so valid calls need 0 < amount <= balance (withdrawing the full balance is allowed). The function never returns a negative balance or silently ignores a bad amount.

1577. **A** — The comment says orders over 1000 get the discount, which excludes exactly 1000, but the code uses >=, so final_price(1000) returns 900.0. At 1500, 999 and 500 the code and comment agree. Someone must confirm which rule is intended.

1578. **B** — sorted() returns ascending order, so top_scores([1, 2]) returns [1, 2] instead of [2, 1]. The bug only shows with at least two different values; fix it with sorted(scores, reverse=True).

1579. **D** — For an even count the median is the mean of the two middle values: 2.5 for [1, 2, 3, 4]. The function returns a[2] = 3. Odd-length inputs return the true middle value, so they hide the bug.

1580. **A** — Affected users need clear facts about what happened to their data and practical steps to protect themselves, in plain language. Hiding or delaying the facts damages trust and leaves students exposed.

1581. **C** — 'Gracefully' has no agreed meaning, so a tester cannot say pass or fail. It should say what happens, for example 'show an error message and keep the user's input'. The other three state checkable behaviour or numbers.

1582. **D** — A default value is evaluated once when the function is defined. Both calls append to that same list, so the second returns [1, 2] and the two results are the same object. The usual fix is items=None and creating a new list inside.

1583. **B** — Code that relied on getting None now crashes, so the change is backward-incompatible. SemVer requires a MAJOR bump for that, signalling users to read the notes before upgrading. Calling it a fix does not change its impact.

1584. **B** — Atomic commits let reviewers understand each change and let the team revert one, such as the dependency update, without undoing the others. A longer message on a mixed commit does not fix that.

1585. **C** — '90% done' with no change suggests an estimate, not a measure. Listing finished and remaining tasks with a date lets the manager see the real position and the risk to the deadline.

1586. **A** — "Aba" reversed is "abA", so the function returns False, but ignoring case it is a palindrome. The fix is to compare s.lower() with its reverse. All-lowercase inputs cannot reveal the missing case handling.

1587. **A** — A wrong docstring is worse than none: readers trust it and write handling code for an exception that never comes, so a None slips through and fails later. Docs must be updated with the code.

1588. **B** — An ER diagram shows entities, attributes and relationships with cardinality, for example that a Student has many Enrolments. That is the structure the new developers need to see.

1589. **D** — Good criteria name a concrete state (a registered user with a valid password), a specific action, and an observable result (the dashboard shows their name). 'It works' gives a tester nothing to check.

1590. **D** — Managers need the effect in business terms: speed that does not degrade at peak load. Big-O notation is right for a technical reviewer, and the last two options say nothing about the benefit.

1591. **D** — With pages numbered from 1, page 1 should return items 0-4, but the slice starts at 1 × 5, returning [5, 6, 7, 8, 9]. The code treats pages as starting at 0. The other calls return empty lists or use page 0, which the docstring does not define, so they do not show the mismatch clearly.

1592. **A** — For $an^2 + bn + c$ the second difference is always 2a, so 2a = 6 gives a = 3. For example $3n^2 + 5n - 2$ gives 6, 20, 40, 66, 98 with second differences 6, 6, 6.

1593. **D** — 11, 13, 17, 19, 23 and 29 are consecutive prime numbers (15, 21, 25 and 27 are skipped because they are composite). The next prime after 29 is 31.

1594. **B** — An interleaved (alternate) series is two series woven together. Positions 1, 3 and 5 give 2, 4, 6 (+2 each time) and positions 2, 4 and 6 give the second series 30, 27, 24 (−3 each time).

1595. **A** — For a cubic an^3 + ... the third differences are constant at 6a. Here a = 2, so they are 12. Check: 1, 14, 51, 124, 245 give first differences 13, 37, 73, 121, second differences 24, 36, 48 and third differences 12, 12.

1596. **D** — V is 22. Adding 5 gives 27, and 27 − 26 = 1, which is A. Counting on: W, X, Y, Z, A.

1597. **B** — The differences form a geometric series (×2), so the next difference is 24 × 2 = 48. The second differences are 3, 6, 12, which are not constant, so the series is not quadratic. The terms themselves (for example 1, 4, 10, 22, 46) do not have a constant ratio.

1598. **D** — The differences are 2, 5, 10, 17, which are $n^2 + 1$ for n = 1 to 4. The next difference is $5^2 + 1 = 26$, so the next term is 42 + 26 = 68.

1599. **B** — Each term is a cube plus 2: 1 + 2, 8 + 2, 27 + 2, 64 + 2, 125 + 2. The next is $6^3 + 2 = 216 + 2 = 218$.

1600. **A** — The operations alternate: +2, ×2, +2, ×2, +2, so the next step is ×2. 30 × 2 = 60.

1601. **B** — It is two interleaved series. Odd positions double (1, 2, 4, 8) and even positions fall by 5 (50, 45, 40). The blank is at an even position, so it is 40 − 5 = 35.

1602. **C** — The jumps are +3, +6, +9, so the next jump is +12. U is 21, and 21 + 12 = 33. Wrapping past Z: 33 − 26 = 7, which is G.

1603. **D** — The differences are 4, 9, 16, 25, the squares of 2 to 5. The next difference is $6^2 = 36$, so the next term is 56 + 36 = 92.

1604. **A** — From the third term on, each term is the sum of the two before it (3 + 1 = 4, 4 + 3 = 7, 7 + 4 = 11, 11 + 7 = 18). So the next is 18 + 11 = 29.

1605. **A** — Differences 9, 13, 17, 21 have a constant second difference of 4, so a = 4 / 2 = 2. Fitting $2n^2 + bn + c$ to the first terms gives b = 3 and c = 0, so $a_n = 2n^2 + 3n$. The 50th term is 2(2500) + 150 = 5150.

1606. **C** — The rule is ×2 − 1: 3 → 5 → 9 → 17 → 33 → 65 → 129. The sixth term should be 65, not 66. 129 = 2 × 65 − 1 confirms that 66 is the odd one out.

1607. **C** — Test a rule that fits both ends: ×3 − 3 gives 4 → 9 → 24 → 69 → 204. The missing term is 3 × 9 − 3 = 24, and 3 × 24 − 3 = 69 confirms it.

1608. **C** — n(n + 2) gives 3, 8, 15, 24, 35, matching every term (it equals $(n + 1)^2 − 1$). Always test a candidate formula on at least three terms, not just the first two.

1609. **D** — The differences are 3, 6, 10, 15, which are triangular numbers, not multiples of 3. The next triangular number is 21, so the next term is 35 + 21 = 56. These are the tetrahedral numbers.

1610. **B** — 5, 8, 13, 20, 29 has differences 3, 5, 7, 9 and constant second differences of 2, so it is quadratic ($n^2 + 4$). The doubling series and the cubes have changing second differences, and an arithmetic series has second differences of zero.

1611. **C** — Odd positions double: 2, 4, 8, 16, 32. Even positions fall by 3: 15, 12, 9, so the next should be 6. The 8th term, 7, is wrong.

1612. **C** — With 8 seats the person opposite is 8 / 2 = 4 seats away, so 4 − 1 = 3 people sit between them on each side. In general, for an even number n the count is n/2 − 1.

1613. **D** — Picture the person at the 6 o'clock seat facing the centre (north). Their left is west, the 9 o'clock side, and moving from 6 to 9 on a clock face is clockwise. If everyone faced outwards, left and right would swap.

1614. **A** — 'Some A are B' means at least one thing is both A and B, so that thing is also a B that is an A: 'Some B are A' always follows. Nothing tells you whether every engineer is a poet or whether any poet is not an engineer.

1615. **D** — 'All A are B' cannot be reversed: there may be graduates who are not managers. With non-empty groups, the managers are graduates, so 'Some managers are graduates' and 'Some graduates are managers' both follow, and 'No manager is a non-graduate' is just a restatement.

1616. **C** — A knight saying it would be lying, which knights never do. A knave saying it would be telling the truth, which knaves never do. So no islander can make the statement; brute-forcing both cases finds no consistent assignment.

1617. **A** — Fix one person's seat to remove rotations, then arrange the other 4 in 4! = 24 ways. 120 = 5! counts each rotation separately.

1618. **B** — Sara is counted in both positions, so the total is 9 + 14 − 1 = 22.

1619. **B** — M is in seat 3 and K in seat 2. If O were at the left end (seat 1), L would need seat 2, which K holds. So O is in seat 5, L in seat 4, and N fills seat 1: N, K, M, L, O.

1620. **D** — Number seats clockwise from G = 0; a person's left is the next seat clockwise. H is in seat 1, J in seat 3 and K in seat 4. I cannot take seat 2 (H's left), so I is in seat 5 and L in seat 2. Seat 5 is opposite seat 2, so L is opposite I.

1621. **A** — Chain the facts: T > V > R (V is between T and R), R > S and S > U. The full order is T, V, R, S, U, so R is third tallest.

1622. **D** — The pencils exist, are all erasers and are never pens, so those erasers are not pens: 'Some erasers are not pens' follows. Whether any pen is an eraser is left open, so neither 'No pens are erasers' nor 'Some pens are erasers' is certain.

1623. **C** — The three claims are mutually exclusive, so at most one is true and at most one speaker is a knight. If nobody were a knight, C's claim would be true, a contradiction. So exactly one is a knight, which makes A's statement true: A is the knight, B and C are knaves.

1624. **B** — Test each suspect. If Bilal did it, Ali's and Chand's statements are true and only Bilal lies: exactly one liar. If Ali did it, Ali and Chand lie. If Chand did it, Bilal and Chand lie. Only Bilal fits.

1625. **A** — B is 40 − 20 + 1 = 21st from the left. Between positions 12 and 21 there are 21 − 12 − 1 = 8 students.

1626. **C** — 'B immediately right of E' forces E into seat 1 (seat 5 has no right neighbour) and B into seat 2, so A must take seat 4 and D seat 5: E, B, C, A, D. Brute force over all 120 orders confirms only this clue leaves one arrangement.

1627. **B** — The phones that are tablets cannot be laptops, so at least some phones are not laptops. The other phones are unconstrained, so nothing more follows.

1628. **D** — If W took it: W lies, X lies, Y lies, and Z ('I did not') tells the truth, which is exactly one truth. Every other suspect makes two or three statements true.

1629. **A** — The middle term 'bloggers' is never covered in full (undistributed middle). A diagram where the coding bloggers sit outside the analyst circle satisfies both premises, so the conclusion is not certain, with or without the non-empty assumption.

1630. **C** — Fix P's seat. Q and S are then fixed. R can take either seat beside P (2 choices), and T and U fill the last two seats in 2 ways, so 2 × 2 = 4 arrangements.

1631. **B** — The possible orders (tallest first) are T R S U, T R U S and T U R S. T is first in all three. Each other statement fails in at least one order.

1632. **B** — The gap between the two rates is 40 − 32 = 8 percentage points. Relative to the starting rate, the increase is 8 / 32 × 100 = 25%. 20% would be 8 / 40, which divides by the new value instead of the old.

1633. **C** — A 100% increase adds 100% of the original value, so the new value is 2 times the old one. For example, 50 becomes 100. A tripling would be a 200% increase.

1634. **A** — An index expresses each value as a percentage of the base year. 80 means the 2024 value is 80% of the 2020 value, which is 20% lower. It says nothing about the units or the year-by-year path.

1635. **D** — Multiply the factors: 1.25 × 0.80 = 1.00, so the price returns exactly to where it started. The 20% fall is taken from a larger base, which cancels the 25% rise. Adding the percentages (+25 − 20 = +5) is the classic error.

1636. **B** — The overall rate is total passes divided by total students, which weights each section by its size. A simple mean treats a 10-student section the same as a 100-student one: 50% and 90% average to 70%, but the true overall rate is 59/110 = 53.6%.

1637. **D** — A full circle is 360°, so each category's angle is its share × 360°. One-eighth of 360° is 45°. 12.5 is the percentage share, not the angle.

1638. **B** — Percentage change = (new − old) / old × 100 = (242 − 160) / 160 × 100 = 82 / 160 × 100 = 51.25%. 82 is the absolute change in thousands, not a percentage.

1639. **C** — Add the Q3 column: 270 + 220 + 162 = 652 thousand. 611 is the Q2 total and 764 is the Q4 total.

1640. **A** — Alpha: 120 / 240 = 50%. Beta: 82 / 160 = 51.25%. Gamma: 72 / 90 = 80%. Gamma grew the most in percentage terms even though Alpha added the most downloads (120 thousand).

1641. **C** — The Q4 total is 360 + 242 + 162 = 764. Alpha's share is 360 / 764 × 100 = 47.1%. 41.4% is Alpha's share in Q3 (270 / 652).

1642. **C** — Each student is worth 360° / 720 = 0.5°, so Java's 90° is 90 / 360 × 720 = 180 students. 90 is the angle, not the count.

1643. **B** — Total marks = 45 × 64 + 35 × 72 = 2,880 + 2,520 = 5,400. Dividing by 80 students gives 67.5. The simple mean of 64 and 72 is 68, which ignores that section A is larger.

1644. **A** — CAGR = (end / start)^(1/years) − 1 = (72,000 / 50,000)^(1/2) − 1 = 1.44^0.5 − 1 = 1.2 − 1 = 20%. 44% is the total growth, and 22% is that total divided by 2, which ignores compounding.

1645. **A** — The new value is 85% of the old one, so old = 340 / 0.85 = 400. Adding 15% to 340 (giving 391) is wrong because the 15% was taken from the original, larger value.

1646. **D** — The rise of 15 index points is measured against 120, not 100: 15 / 120 × 100 = 12.5%. Index points equal percentages only when you compare with the base year.

1647. **D** — The overall rate is a weighted average, so it depends on the department sizes. With 100 CS and 20 IT students it is 63.3%; with 20 CS and 100 IT it is 76.7%. You can only say it lies between 60% and 80%, so no single value can be chosen.

1648. **B** — Gamma stayed at 162 in Q3 and Q4, but the total rose from 652 to 764, so its share fell from 24.8% to 21.2%. A flat value can still lose share when the total grows.

1649. **A** — Gamma went from 90 to 135, a rise of 45 / 90 = 50%. The next largest is Alpha from Q3 to Q4 at 90 / 270 = 33.3%, which is the largest absolute rise but not the largest percentage.

1650. **C** — X grew 60 / 200 = 30% and Y grew 200 / 1,000 = 20%. So X grew faster as a rate, while Y grew more in absolute units. A good answer names which measure the claim uses.

1651. **D** — February: 20 / 50 = 40%. April: 21 / 63 = 33.3%. May: 26 / 84 = 31.0%. March fell. May had the largest absolute rise (26 thousand), but February had the largest percentage rise because its starting value was smallest.