NSCT Programming Cheat Sheet (C++, Java, Python)

By the NSCT Exam Editorial Team · Updated 2026-11-24 · Facts checked against official sources on 2026-11-24

Programming (C++/Java/Python) carries 10% of the NSCT according to HEC's Areas of Competencies, and 2026 candidates averaged 47.8% accuracy in it, per the HEC NSCT report and APP. Print this page and keep it next to your practice: most questions are "what does this print?", and the answers hide in the language differences below.

Key definitions

Term	One line
Compiler	Translates the whole program before it runs
Interpreter	Executes the program as it translates it
Linker / loader	Joins object files / puts the executable in memory
Parameter vs argument	Name in the definition vs value in the call
Pass-by-value	The callee gets a copy (Java always; for objects the reference is copied)
Encapsulation	Hiding state behind methods
Polymorphism	One call, behaviour chosen by the object's actual type
Abstract class	Cannot be instantiated; can hold state and code
Dangling pointer	Points to memory that has been freed or gone out of scope
Race condition	Result depends on the timing of unsynchronised threads
Closure	Function that captures variables (not values) from its scope

C++ vs Java vs Python

Feature	C++	Java	Python
Runs as	Native code	Bytecode on the JVM (JIT)	CPython bytecode, interpreted
`int` size	Minimum only	Fixed 32-bit, wraps on overflow	Unlimited precision
`-7 / 2`	-3	-3	-3.5 (`/`); `-7 // 2` = -4
`-7 % 2`	-1	-1	1
Strings	`std::string` mutable	`String` immutable	`str` immutable
Inheritance	Multiple	Single class, many interfaces	Multiple
Dynamic dispatch	Only `virtual` functions	All instance methods	All methods
Memory freed by	`delete`, RAII, smart pointers	Tracing garbage collector	Reference counting + cycle detector
`finally`	None (use RAII)	Yes	Yes
Checked exceptions	No	Yes	No
`switch` fall-through	Yes, without `break`	Yes, without `break`	No (`match`)

Rules and formulas

Rule / formula	Notes
Precedence	postfix > unary > `* / %` > `+ -` > shifts > relational > equality > `&` > `^` > `|` > `&&` > `||` > ternary > assignment
`~x = -x - 1`	`~5` = -6
`x << n = x × 2^n`	For non-negative values without overflow
Row-major address = base + (i × C + j) × size	C = number of columns

Rule / formula	Notes
Java `int` range	-2,147,483,648 to 2,147,483,647
C string length	Characters + 1 for `'\0'` (`"hello"` needs 6 bytes)
Amdahl speed-up = 1 / ((1 − p) + p / N)	p = parallel fraction, N = cores; limit 1 / (1 − p)
n `fork()` calls in sequence	2^n processes

Worked outputs to check yourself

Code	Result
Java `Integer.MAX_VALUE + 1`	-2147483648
Java `(byte) 130`	-126
Java `(int) 9.99` , Python `int(-3.99)`	9, -3 (truncate toward zero)
`double avg = 7 / 2;`	3.0
Python `2 ** 3 ** 2` , `-2 ** 2`	512, -4
Java `1 + 2 + "3" + 4`	`"334"`
Java `'a' + 1`	98
`int a[3][5]` at 1000, 4-byte int, `a[2][3]`	1000 + (2 × 5 + 3) × 4 = 1052
p = 0.9, N = 10 (Amdahl)	≈ 5.26; limit 10
Python `[lambda: i for i in range(3)]` called	All return 2

Collections at a glance

Structure	Order	Average lookup
`HashMap` , `unordered_map` , `HashSet` , Python `set`	None guaranteed	O(1)
`TreeMap` , `std::map` , `TreeSet`	Sorted	O(log n)
`LinkedHashMap` , Python `dict` (3.7+)	Insertion	O(1)
Dynamic array (`ArrayList` , `vector` , `list`)	Index	O(1) index, O(1) amortised append, O(n) insert at front

OOP, memory and exceptions

- **Four pillars:** encapsulation, inheritance, polymorphism, abstraction.
- **Java access:** `private` < default (package) < `protected` (package + subclasses) < `public` .
- `Animal a = new Dog(); a.sound();` runs `Dog` 's method; fields and static methods are not polymorphic.
- **Smart pointers:** `unique_ptr` one owner; `shared_ptr` use count; `weak_ptr` breaks cycles.
- `new` pairs with `delete` ; `new[]` with `delete[]` .
- **Stack:** frames and locals, freed on return. **Heap:** dynamic objects, freed by `delete` or GC.
- **Exceptions:** rest of `try` skipped; first matching handler runs; `finally` always runs. Subclass handlers go first.
- **Java checked** = `Exception` minus `RuntimeException` ; `NullPointerException` is unchecked.
- **Threads:** share code, heap and globals; each has its own stack, PC and registers. Java `start()` makes a thread; `run()` does not.

X vs Y quick distinctions

Pair	Difference
`x++` vs `++x`	Yields the old value vs the new value

Pair	Difference
`==` vs `equals()` (Java)	Same object vs same contents
Overloading vs overriding	Compile time, different parameters vs run time, same signature
Abstract class vs interface	Shared state and code vs shared capability
Pointer vs reference (C++)	Re-pointable, nullable vs fixed alias, never null
Checked vs unchecked	Must catch or declare vs no compiler rule
`break` vs `continue`	Leaves the innermost loop vs skips to the next pass
`while` vs `do-while`	May run zero times vs at least once
Concurrency vs parallelism	In progress together vs at the same instant
Static vs dynamic library	Copied in at link time vs loaded at run time
`git revert` vs `git reset`	New undo commit (safe when shared) vs moves the branch back
`volatile` vs atomic	Visibility only vs indivisible update

Classic traps

1. Java `i = i++;` leaves `i` unchanged; C++ `a++ + ++a` is undefined behaviour.
2. In C++/Java `==` binds tighter than `&`: `x & 1 == 0` means `x & (1 == 0)`.
3. `float f = 2.5;` does not compile in Java; write `2.5f`.
4. A stray `;` after `if (...)` or `for (...)` empties the body.
5. Python default `bag=[]` is created once and shared between calls.
6. Python `sort()` returns `None`; `name.upper()` without assignment changes nothing.
7. `[[0] * 3] * 2` repeats one row object, so editing one row edits both.
8. Java `list.remove(1)` on a `List<Integer>` removes index 1, not the value 1.
9. `try { return x; } finally { x = 20; }` returns the saved value; a `return` in `finally` overrides.
10. `nextInt()` or `cin >> n` leaves a newline, so the next line-read returns empty.
11. `range(5, 0)` is empty; a countdown needs `range(5, 0, -1)`.
12. Removing from a Java list inside for-each throws `ConcurrentModificationException`.

Keep going

- Full explanations: Programming revision notes
- All 16 topics and quizzes: Programming MCQs hub
- Deep pages: OOP code questions, pointers and memory, operators and expressions
- Timed practice: subject-wise mock test

Next step

Cover the "Result" column and predict every output, then take a 20-question Programming set in the subject-wise mock test. Code-tracing questions from every subject are collected in code tracing practice, and the full NSCT mock test puts it all together.