The Simple AP® Computer Science A Unit 2 Cheat Sheet

2.1–2.2 Algorithms & Boolean Expressions

Algorithms

Algorithms use **sequencing** (steps in order), **selection** (if) & **repetition** (loops).

Selection picks which code runs; **iteration** repeats code while a condition stays true.

Relational Operators

Operator	Meaning	Ex (x = 5)
==	equal to	x == 5 → true
!=	not equal	x != 5 → false
< , >	less / greater	x > 3 → true
<= , >=	or equal	x <= 4 → false

Boolean expression = true or false; you can store it: boolean pass = score >= 70;

Comparing Objects

Strings: use s1.equals(s2) to compare characters; == only checks if they're the same object.

Null check: obj == null is fine, but calling a method on a null object throws an exception.

Doubles & Assignment

Avoid == on doubles: 0.1 + 0.2 == 0.3 is false; check if Math.abs(a − b) is tiny instead.

= vs. ==: = stores a value in a variable; == compares 2 values & gives true or false.

⚠ s1 == s2 can be false even when both Strings say "hi"; always use .equals for text.

✓ Plug in values & write T or F above each part of a Boolean expression as you go.

2.3–2.4 if, else & else if

if & if-else

if (condition) runs its block only when the condition is true; otherwise it's skipped.

if-else: exactly 1 of the 2 blocks runs. **else if**: checks in order; the 1st true one runs.

score	Chain result
95	A (1st true)
85	B
65	else → F

Order Matters

Chain: if (s >= 90) A; else if (s >= 80) B; else F. A score of 95 gets only A, never B.

Put the strictest test 1st: if (s >= 80) B; else if (s >= 90) A → a 95 wrongly gets B.

Separate ifs are each checked, so several can run; an else-if chain runs at most 1 branch.

Nested if & Braces

Nested if: the inner test runs only when the outer test was true; trace outer, then inner.

Dangling else: an else pairs w/ the NEAREST if, no matter how the code is indented.

No braces: only the 1st statement after if belongs to it; Java ignores your indentation.

Ex: if (x > 0) count++; total++; → total++ ALWAYS runs; only count++ depends on x.

⚠ A stray semicolon, if (x > 0); { ... }, ends the if, so the block below always runs.

✓ In a chain, stop at the 1st true test & cross out every branch below it, even else.

2.5–2.6 Compound Booleans & De Morgan

Logical Operators

A	B	A && B	A || B	!A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

Order: ! first, then &&, then ||. Use parentheses to make your meaning clear to readers.

Short-Circuit Evaluation

&& skips the right side if the left is false; **||** skips the right side if the left is true (short-circuit).

Guard: if (x != 0 && 10 / x > 2) is safe; when x = 0, the division never runs, so nothing crashes.

De Morgan's Laws

!(a && b) = !a || !b. **!(a || b)** = !a && !b. Flip each part & swap && ↔ || to simplify.

Ex: !(x > 5 && y == 3) = x <= 5 || y != 3; the opposite of > is <= (it keeps the 5).

Equivalent Expressions

Test values: try 2–3 cases, including boundary values, to see if 2 expressions match.

Range check: x >= 1 && x <= 10 means 1 to 10; 1 <= x <= 10 won't even compile.

Odd test: use n % 2 != 0 (works for negatives).

⚠ The opposite of x > 5 is x <= 5, not x < 5.

✓ Build a mini truth table for "which is equivalent" questions, testing values like 5 & 6.

2.7 while Loops

Structure

while (condition) checks BEFORE each pass; if false at the start, the body runs 0 times.

3 parts: set the variable before, test it in the condition & update it inside the loop.

Ex: int i = 1; while (i <= 5) { sum += i; i++; } → sum = 15, & i ends at 6, not 5, after the last test.

Common Patterns

Sentinel loop: keep going until a stop value (like −1) appears in the input you read.

Digits: while (n > 0) { n /= 10; count++; } counts digits: 472 → 47 → 4 → 0, so 3 passes.

Errors

Infinite loop: the update never makes the condition false (you forgot i++).

Off-by-one: < vs. <= changes the number of passes by exactly 1; trace the 1st & last pass.

Loop	Passes
i = 0; i < n	n
i = 1; i <= n	n
i = 0; i <= n	n + 1
i = 1; i < n	n − 1

Final value: the 1st one that fails the test.

⚠ A variable declared inside the loop can't be used after the loop; declare it before.

✓ Trace w/ a table: 1 column per variable & 1 row per pass, plus the loop's test result.

2.8 for Loops

Structure

for (init; condition; update): init runs once; test before each pass; update after each pass.

Ex: for (int i = 0; i < 5; i++) prints 0 1 2 3 4; i doesn't exist after the loop ends.

Same as while: any for loop can be rewritten as a while loop w/ the same 3 parts.

Counting Passes

Formula: i = a; i < b → b − a passes. i = a; i <= b → b − a + 1 passes. Check 1 case.

Step size: for (i = 0; i < 20; i += 5) → 0, 5, 10, 15 → 4 passes; list the values out to check.

Count down: for (int i = 10; i > 0; i -= 2) → 10, 8, 6, 4, 2 → 5 passes; 0 fails the test.

Choosing a Loop

Use	When
for	Known # of passes
while	Stop on a condition
for over a String	Visit each index

Common Errors

Semicolon trap: for (...); → an empty loop, so the block after it runs only 1 time, not 5.

Changing i inside a for body too (an extra i++) skips values; MCQs love this trick.

⚠ i <= str.length() goes 1 past the last index & crashes w/ an out-of-bounds error.

✓ Use the formula, then check the 1st & last values of i by tracing them on paper.

2.9–2.10 Standard Algorithms

Number Algorithms

Sum & average: total += value each pass; average = (double) total / count to avoid int division.

Min / max: start w/ the 1st value, then update whenever a smaller or larger value appears.

Count matches: if (n % 2 == 0) count++; counts the even numbers as the loop runs.

Digits: n % 10 gives the last digit; n / 10 drops it.

Divisible? a % b == 0 means b divides a evenly.

String Algorithms

Task	Key code
Each char	s.substring(i, i + 1)
Reverse	rev = ch + rev
Count a letter	if (ch.equals("a"))
Find a word	s.indexOf("cat") != −1

Traverse: for (int i = 0; i < s.length(); i++) visits each index; the last one is length − 1.

Ex reverse: "abc" → "a" → "ba" → "cba" by putting each new char in front of the result.

Ex count: count "a" in "banana" → check each 1-letter substring → matches at 1, 3, 5 → 3.

Early exit: return true as soon as you find a match, but return false only AFTER the loop ends.

Min start: use the 1st value, not 0.

⚠ 7 / 2 = 3 in Java (int division); cast to double for an average: (double) sum / n.

✓ Count, sum, min/max & traverse patterns appear on the FRQs every year.

2.11–2.12 Nested Loops & Run Time

Nested Iteration

Inner loop runs ALL its passes for EACH outer pass; total = outer × inner passes.

Ex: outer 3 passes × inner 4 passes → the inner body runs 12 times in total; count before tracing.

Dependent inner: j < i w/ i = 0...3 → 0 + 1 + 2 + 3 = 6 passes in total, not 4 × 4 = 16.

Triangle ex: i = 1...4, j = i...4 → 4 + 3 + 2 + 1 = 10 runs of the inner body, not 4 × 4.

Patterns

Code	Output
i 1..3, j 1..i print *	*, **, ***
i 0..2, j 0..2 print i+j	0 1 2 / 1 2 3 / 2 3 4

Rows & columns: the outer loop picks the row; the inner loop fills that row's columns.

print vs. println: print stays on 1 line; println after the inner loop starts a new row.

Run-Time Analysis

Statement count: how many times a line runs; multiply the passes of every loop around it.

Ex: for (k = 0; k < n; k += 2) runs about n/2 times, half as many as the usual k++ loop.

Compare: n × n nested loops → n^2 runs; halving n each pass → only about $\log_2 n$ runs.

⚠ If the inner loop depends on the outer variable, add up each row's count instead.

✓ List the inner count for each outer value (0, 1, 2, ...), then add them all up at the end.

Tracing & Exam Strategy

Exam Format

Section I: 42 MCQs in 90 min (55%). **Section II**: 4 FRQs in 90 min (45% of the score).

Unit 2 = 25–35% of the exam, & its loops & ifs show up inside every FRQ you write.

Tracing Method

Trace table: 1 column per variable + the condition; update after every line of code.

Shortcut: for long loops, find the pattern after 3 passes, then jump to the last pass.

MCQ pacing: ≈ 2 min per question; skip long traces & come back to them at the end.

Top Unit 2 Traps

Trap	Fix
< vs. <=	Count passes
== on Strings	Use .equals
Missing braces	Only 1st line is in
else-if order	Strictest test 1st
int / int	Cast to double

FRQ Tips

Return, don't print, unless the method says to print; match the return type exactly.

Edge cases: test your code w/ an empty String, a single element & the very last index.

⚠ Don't return false inside the loop on the 1st mismatch; return it after the loop.

✓ Write the loop header 1st & check its bounds, then add the if statement inside it.