AP® COMPUTER SCIENCE A
GENERAL SCORING GUIDELINES

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b , c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times, or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

(w) Extraneous code that causes side effect (e.g., printing to output, incorrect precondition check)
(x) Local variables used but none declared
(y) Destruction of persistent data (e.g., changing value referenced by parameter)

Mr Lee's 1-Point Penalty:

- Inefficient, "long winded" or "messy" difficult to understand code which takes longer to write than standard more efficient solutions.
    - In an exam you need to save time by writing quickly hand writable efficient code which is easy for AP readers to understand.

No Penalty

- Extraneous code with no side effect (e.g., precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- Keyword used as an identifier
- Common mathematical symbols used for operators (x • ÷ ≤ ≥< > ≠)
- `=` instead of `==` and vice versa
- Missing `{}` where indentation clearly conveys intent
- Missing `()` around `if` conditions

** Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be unambiguously inferred from context; for example, "total" instead of "totl". As a counterexample, that if the code declares "int G=99 , g=O; ", then uses "while (G < 10) " instead of "while (g < 10) ", the context does not allow for the reader to assume the use of the lower-case variable.*

if – APLine FRQ

An `APLine` is a line be defined by the equation *ax+ by + c = 0* . where *a* is not equal to zero, *b* is not equal to zero, and *a*, *b*, and *c* are all `int`s. The slope of an `APLine` is defined to be the `double` value `-a/b`. A point (represented by integers `x` and `y`) is on an `APLine` if the equation of an `APLine` is satisfied when those *x* and *y* values are substituted into the equation. That is, a point represented by *x* and *y* is on the line if *ax + by + c* is equal to *0*. Examples of `APLine` equations are shown in the following table.

Equation	**Slope (*-a / b*)**	**Is point (5, -2) on the line?**
5x + 4y - 17 = 0	-5/ 4 = -1.25	Yes, because 5(5) + 4(-2) + (-17) = 0
-25x + 40y + 30 = 0	25/ 40 = 0.625	No, because -25(5) + 40(-2) + 30 != 0

Write code segments that will print:
(a) The slope of an APLine defined by the equation *ax+ by + c = 0*.

(b) The `boolean` value `true` if the point (`x`, `y`) is on the same APLine, as in part *(a)* above, defined by the equation *ax+ by + c = 0*, the `boolean` value `false` otherwise.