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` or `while` 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.*

Strings – ScrambledWord FRQ

This question involves reasoning about strings made up of uppercase letters.

Write a code segment which takes a given word and contructs a String that contains a scrambled version of the word according to the following rules.

- The scrambling process begins at the first letter of the word and continues from left to right.
- If two consecutive letters consist of an "A" followed by a letter that is not an "A", then the two letters are swapped in the resulting string.
- Once the letters in two adjacent positions have been swapped, neither of those two positions can be involved in a future swap.

The following table shows several examples of words and their scrambled versions.

word	scrambled version
"TAN"	*"TNA"*
"ABRACADABRA"	*"BARCADABARA"*
"WHOA"	*"WHOA"*
"AARDVARK"	*"ARADVRAK"*
"EGGS"	*"EGGS"*
"A"	*"A"*
""	*""*

Complete the code segment below.

```
/**
* Scrambles a given word.
* @param word the word to be scrambled
* @ prints the scrambled word (possibly equal to word)
* Precondition: word is either an empty string or contains only
* uppercase letters.
* Postcondition: the String printed is stored in a String variable and
* was created from word as follows:
*  - the word was scrambled, beginning at the first letter and
*    continuing from left to right
*  - two consecutive letters consisting of "A" followed by a letter
*    that was not "A" were swapped
*  - letters were swapped at most once
*/
String word;
```