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.*

This question involves the `StringManip` class, which is used to perform manipulation on strings.

(a) Write a code segment, which takes the `String` variable `str` and prints a new string with spaces removed. For example, if `str` points to `"hi how are you"`, the code segment should print `"hihowareyou"`.

Complete the code segment below.

```
/** Takes a string str and prints a new string
 *  with all spaces removed.
 */
String str = "hi how are you";
```

(b) A proceeding code segment in the `StringManip` class, which takes the `String` produced by part *(a)* *(`str` with spaces removed)* and prints a new string with the characters in reverse order. For example, if `str` with spaces removed is `"ABCDE"` the code segment should print `"EDCBA"`.

Complete the code segment below by assigning the reversed string to `reverseString`.

```
/** Takes a string str and returns a new string
 *  with the characters reversed.
 */
String reverseString = "";
System.out.println(reverseString);
```

(c) For this question, let a palindrome be defined as a string that, when spaces are removed, reads the same forward and backward. For example, `"race car"` and `"taco cat"` are palindromes. You will write a code segment, which determines whether the `String str` is a palindrome and prints a message indicating the result. Examples of the intended behavior of the method are shown in the following table.

`String str`	Printed Message
`"taco cat"`	`taco cat is a palindrome`
`"laid on no dial"`	`laid on no dial is a palindrome`
`"level up"`	`level up is not a palindrome`

Write code segment below. Assume that previous code segments in parts *(a)* & *(b)* works as specified, regardless of what you wrote in them. You must use results of the previous code segments in parts *(a)* & *(b)* appropriately to receive full credit. Your implementation must conform to the examples in the table.

```
/** Determines whether str is a palindrome and prints a
 *  message indicating the result, as described in part (c).
 *  Precondition: str contains only lowercase letters and
 *  spaces.
*/
```