AP® COMPUTER SCIENCE A
2014 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.

1-Point Penalty

(w) Extraneous code that causes side effect (*e.g., writing to output, failure to compile*)

(x) Local variables used but none declared

(y) Destruction of persistent data *(e.g., changing value referenced by parameter)*

(z) Void method or constructor that returns a value

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
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ ≤ ≥ <> ≠)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- Array/collection access confusion (`[] get`)
- `length/size` confusion for array, `String`, `List`, or `ArrayList`, with or without `( )`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, *e.g.,* `int[size] nums = new int[size];`
- Missing `;` provided majority are present and indentation clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent and `{ }` are used elsewhere
- Missing `( )` on parameter-less method or constructor invocations
- 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, "`ArayList`" instead of "`ArrayList`". As a counterexample, note that if the code declares "`Bug bug;`", then uses "`Bug.move()`" instead of "`bug.move()`", the context does **not** allow for the reader to assume the object instead of the class.*

AP® COMPUTER SCIENCE A
2014 SCORING GUIDELINES

Question 1: Word Scramble

Part (a)	`scrambleWord`	**5 points**

Intent: *Scramble a word by swapping all letter pairs that begin with A*

+1 Accesses all letters in `word`, left to right (*no bounds errors*)

+1 Identifies at least one letter pair consisting of "A" followed by non-"A"

+1 Reverses identified pair in constructing result string

+1 Constructs correct result string (*Point lost if any letters swapped more than once, minor loop bounds errors ok*)

+1 Returns constructed string

Part (b)	`scrambleOrRemove`	**4 points**

Intent: *Modify list by replacing each word with scrambled version and removing any word unchanged by scrambling*

+1 Accesses all words in `wordList` (*no bounds errors*)

+1 Calls `scrambleWord` with a word from the list as parameter

+1 Identifies words unchanged by scrambling

+1 On exit: List includes all and only words that have been changed by scrambling once, in their original relative order (*minor loop bounds errors ok*)

AP® COMPUTER SCIENCE A
2014 CANONICAL SOLUTIONS

Question 1: Word Scramble

Part (a):

```
public static String scrambleWord(String word){
   int current = 0;
   String result="";
   while (current < word.length()-1){
      if (word.substring(current,current+1).equals("A") &&
            !word.substring(current+1,current+2).equals("A")){
         result += word.substring(current+1,current+2);
         result += "A";
         current += 2;
      }
      else {
         result += word.substring(current,current+1);
         current++;
      }
   }
   if (current < word.length()){
      result += word.substring(current);
   }
   return result;
}
```

Part (b):

```
public static void scrambleOrRemove(List<String> wordList){
   int index = 0;
   while (index < wordList.size()){
      String word=wordList.get(index);
      String scrambled=scrambleWord(word);
      if (word.equals(scrambled)){
         wordList.remove(index);
      }
      else {
         wordList.set(index,scrambled);
         index++;
      }
   }
}
```

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.