AP® COMPUTER SCIENCE A
2015 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

(v) Array/collection access confusion (`[] get`)

(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
- `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 `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- 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
2015 SCORING GUIDELINES

Question 2: Guessing Game

Class:	`HiddenWord`	**9 points**

Intent: *Define implementation of class to represent hidden word in guessing game*

+1 Uses correct class, constructor, and method headers

+1 Declares appropriate `private` instance variable

+1 Initializes instance variable within constructor using parameter

+6 Implement `getHint`

- **+1** Accesses all letters in both guess and hidden word in loop (*no bounds errors in either*)
- **+4** Process letters within loop
  - **+1** Extracts and compares corresponding single letters from guess and hidden word
  - **+1** Tests whether guess letter occurs in same position in both guess and hidden word
  - **+1** Tests whether guess letter occurs in hidden word but not in same position as in guess
  - **+1** Adds correct character exactly once to the hint string based on the test result
- **+1** Declares, initializes, and returns constructed hint string

Question-Specific Penalties

-1 (t) Uses `get` to access letters from strings

-2 (u) Consistently uses incorrect name instead of instance variable name for hidden word

AP® COMPUTER SCIENCE A
2015 CANONICAL SOLUTIONS

Question 2: Guessing Game

```
public class HiddenWord
{
    private String word;

    public HiddenWord(String hWord)
    {
        word = hWord;
    }

    public String getHint(String guess){
        String hint = "";
        for (int i = 0; i < guess.length(); i++){
          if (guess.substring(i,i+1).equals(word.substring(i,i+1))){
                hint += guess.substring(i,i+1);
          } else if (word.indexOf(guess.substring(i,i+1))!= -1){
                hint += "+";
          } else {
                hint += "*";
          }
        }
        return hint;
    }

}
```

2A

```
Public class HiddenWord {
  String word;
  Public HiddenWord(String w){
      word = w;
  }
  Public String getHint(String guess){
    String s = "";
  for(int i=0; i<guess.length(); i+1){
    if(word.substring(i,i+1).equals(guess.substring(i,i+1))){
       s+= guess.substring(i,i+1);
    } else{
       if(word.indexOf(guess.substring(i,i+1)) > -1){
            s+= "+";
       } else{
            s+= "*";
       }
    }
  }
  return s;
}
```

2. Consider a guessing game in which a player tries to guess a hidden word. The hidden word contains only capital letters and has a length known to the player. A guess contains only capital letters and has the same length as the hidden word.

After a guess is made, the player is given a hint that is based on a comparison between the hidden word and the guess. Each position in the hint contains a character that corresponds to the letter in the same position in the guess. The following rules determine the characters that appear in the hint.

If the letter in the guess is …	**the corresponding character in the hint is**
also in the same position in the hidden word,	the matching letter
also in the hidden word, but in a different position,	`"+"`
not in the hidden word,	`"*"`

The `HiddenWord` class will be used to represent the hidden word in the game. The hidden word is passed to the constructor. The class contains a method, `getHint`, that takes a guess and produces a hint.

For example, suppose the variable `puzzle` is declared as follows.

```
HiddenWord puzzle = new HiddenWord("HARPS");
```

The following table shows several guesses and the hints that would be produced.

Call to `getHint`	**String returned**
`puzzle.getHint("AAAAA")`	`"+A+++"`
`puzzle.getHint("HELLO")`	`"H****"`
`puzzle.getHint("HEART")`	`"H*++*"`
`puzzle.getHint("HARMS")`	`"HAR*S"`
`puzzle.getHint("HARPS")`	`"HARPS"`

Write the complete `HiddenWord` class, including any necessary instance variables, its constructor, and the method, `getHint`, described above. You may assume that the length of the guess is the same as the length of the hidden word.

```
public class HiddenWord
{
   String word;
   public HiddenWord(String w)
   {
      word = w;
   }
```


GO ON TO THE NEXT PAGE.

ADDITIONAL WORK SPACE

```
public String getHint( String guess)
{  String answer = "";
   for(int x=0; x < guess.length; x++)
   {
      if(guess.Substring(x,x).equals(word.Substring(x,x));
      {
         answer = answer + guess.Substring(x,x);
      }
     else if(guess.Substring(x,x).equals(word.Substring(0,Word.length);
      {  answer = answer + "+";
      }
      else
      {
         answer = answer + "*";
      }
   }
   return answer;
}
}
```

ADDITIONAL WORK SPACE

```
public class HiddenWord {
    private string wordHidden;

public void getHint (string hint){
    string Returned = hint;
for (int a=0 ; a<hint.length ; a++){
    if (hint.substring(a)  = wordHidden.substring(a)
        Returned.substring(a)  = hint.substring(a);

if( (hint.substring(a)  != wordHidden.substring(a))
&& wordHidden.indexOf(hint.substring(a))!= -1)

    Returned.substring(a) = "+";
if (wordHidden.indexOf(hint.substring(a) = -1)
    Returned.substring(a) = "*";

}
return Returned;
}
```

GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2015 SCORING COMMENTARY

Question 2

Overview

This question focused on class design and the `String` class. Students had to demonstrate an understanding of class, constructor, and method header syntax. Students were required to understand the importance of declaring instance variables `private` and initializing instance variables in a constructor. Students were also required to demonstrate an understanding of `String` methods.

Students were asked to write a class that represents a hidden word and includes a method to return a hint for a given guess. To create the hint string, students had to traverse the hidden word and the guess by comparing each letter, one by one, and adding a character to the resulting hint string. The character was the letter if the guess letter and the hidden word letter matched and were in the same position. The character was a `"+"` if the guess letter and the hidden word letter matched and were in different positions. The character was a `"*"` if the guess letter and the hidden word letter did not match. The resulting hint string was then returned.

Sample: 2A
Score: 8

The student correctly creates the class and constructor headers and the `getHint` method header, which earned the first point. The instance variable was initialized using the constructor's parameter, but it was not declared `private`, so the third point was earned, but not the second point.

The student writes a correct `getHint` method, initializing a hint string and accessing all the letters in both the guess and the hidden word in a correctly bounded loop. Within the loop, single letters from the guess and the hidden word are extracted and compared correctly. A single letter is appended to the hint string if the characters from the guess and the hidden word match and are in the same position. A single `"+"` is appended to the hint string if the characters from the guess and the hidden word match but are not in the same position. A single `"*"` is appended to the hint string if the letter from the guess does not match any letter from the hidden word. The correctly constructed hint string is returned, and the student earned all 6 points for the `getHint` method.

Sample: 2B
Score: 6

The student correctly creates the class and constructor headers and the `getHint` method header, which earned the first point. The instance variable is initialized using the constructor's parameter, but it is not declared `private`, so the third point was earned, but not the second point.

For the `getHint` method, the student writes a good loop but uses "`substring(x, x)`" instead of "`substring(x, x+1)`," so the "access all letters" point is not earned. Within the loop, single letters from the guess and the hidden word are extracted and compared correctly and the "corresponding letters" point is earned. A single letter is appended to the hint string if the characters from the guess and the hidden word match and are in the same position, so the "test in same position" point is earned. The test for matching single letters in different positions compares a single letter from the guess to the entire hidden word string, so the "test not in same position" point is not earned. A single `"*"` is appended to the hint string if the letter from the guess does not match any letter from the hidden word. The hint string was initialized, constructed, and returned. The student earned 4 points for the `getHint` method.

AP® COMPUTER SCIENCE A
2015 SCORING COMMENTARY

Question 2 (continued)

Sample: 2C
Score: 2

The student correctly creates the class header but there is no constructor and there is an incorrect return type on the `getHint` method, so the first point is not earned. The instance variable is declared `private`, but it is not initialized so the second point is earned, but not the third point.

For the `getHint` method, the student writes a good loop header so the "access all letters" point was earned. The point for "corresponding letters" was not earned due to the use of "`substring(a)`" instead of "`substring(a, a+1)`." The student used the primitive test "`==`" rather than the object test "`.equals`", so the "test in same position" point was not earned. The test for matching letters in different positions was incorrect, and the attempt to add single characters to the hint string used assignments made to substring calls, so the "test not in same position" and "add correct character" points were not earned. The hint string was not properly initialized, so the final point is not earned. The student earned 1 point for the `getHint` method.