2017

AP® CollegeBoard

AP Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

☑ Free Response Question 2

☑ Scoring Guideline

☑ Student Samples

☑ Scoring Commentary

© 2017 The College Board. College Board, Advanced Placement Program, AP, AP Central, and the acorn logo are registered trademarks of the College Board. Visit the College Board on the Web: www.collegeboard.org.

AP Central is the official online home for the AP Program: apcentral.collegeboard.org

AP® COMPUTER SCIENCE A
2017 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., 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)

z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid 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* "`int G=99, g=0;`", *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.*

© 2017 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

Question 2: MultPractice

Class:	`MultPractice`	**9 points**

Intent: *Define implementation of class to produce multiplication practice problems*

+1 Declares header: `public class MultPractice implements StudyPractice`

+1 Declares all necessary `private` instance variables

+2 Constructor

- **+1** Declares header: `public MultPractice(int __, int __)`
- **+1** Initializes all instance variables using parameters

+3 `getProblem` method

- **+1** Declares header: `public String getProblem()`
- **+1** Builds string with current values of instance variables
- **+1** Returns constructed string

+2 `nextProblem` method

- **+1** Declares header: `public void nextProblem()`
- **+1** Updates instance variable(s) to reflect incremented second number

© 2017 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

Question 2: Scoring Notes

Class `MultPractice`			**9 points**
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Declares header: `public class MultPractice implements StudyPractice`	• omit keyword `public`	• declare class `private`
+1	Declares all necessary `private` instance variables	• declare the unchanging instance variable as `final`	• declare variables as `static` • omit keyword `private`
+2	Constructor		
+1	Declares header: `public MultPractice (int ___, int ___)`	• omit keyword `public`	
+1	Initializes all instance variables using parameters		• fail to declare nonlocal variables • initialize local variables instead of instance variables • assign variables to parameters
+3	`getProblem` method		
+1	Declares header: `public String getProblem()`		• fail to declare method `public`
+1	Builds string with current values of instance variables	• write appropriate code in a method other than `getProblem` • make capitalization or spacing errors	• fail to declare nonlocal variables • fail to use instance variables • miscast `(String) intVar` • call `intVar.toString()`
+1	Returns constructed string		• return a literal string
+2	`nextProblem` method		
+1	Declares header: `public void nextProblem()`		• fail to declare method `public`
+1	Updates instance variable(s) to reflect incremented second number		• fail to declare non-local variables

© 2017 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

Question 2: MultPractice

```
public class MultPractice implements StudyPractice
{
     private int first;
     private int second;

     public MultPractice(int num1, int num2)
     {
          first = num1;
          second = num2;
     }

     public String getProblem()
     {
          return first + " TIMES " + second;
     }

     public void nextProblem()
     {
          second++;
     }
}
```

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.

© 2017 The College Board.

2A

Write the complete `MultPractice` class. Your implementation must be consistent with the specifications and the given examples.

```
Public class MultPractice implements StudyPractice {

   Private int first;
   Private int second;

 Public MultPractice(int first, int second){
this.first= first;
this.second= second;
  }

  Public String getProblem() {
      String Problem = "";
      Problem += first + "TIMES " + second;
      return Problem;
   }

   Public void nextProblem() {
      second + 1;

   }

   }
```

Unauthorized copying or reuse of any part of this page is illegal.

GO ON TO THE NEXT PAGE.

© 2017 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

2B

Write the complete MultPractice class. Your implementation must be consistent with the specifications and the given examples.

```
public class MultPractice implements StudyPractice
 {
  public MultPractice (int a, int b)
   {
    a=a;
    b=b;
   }
  public String getProblem()
   {
    System.out.println(a + "TIMES" + b);
   }
  public void nextProblem()
   {
    b=b+1;
   }
```

Unauthorized copying or reuse of any part of this page is illegal.

GO ON TO THE NEXT PAGE.

© 2017 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

2C

Write the complete MultPractice class. Your implementation must be consistent with the specifications and the given examples.

```
Public Class MultPractice extends - StudyPractice
{
   int first=0;
   int second=0;
  Public MultPractice( x, y)
  {
     x=first;
     y=second;
  }
 Public String getProblem()
 {return x + "TIMES" + y;
 }
  Public void nextProblem()
 {
   y++;
 }
}
```

Unauthorized copying or reuse of any part of this page is illegal.

© 2017 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

Question 2

Overview

This question focused on the mechanics of creating a class to implement a specific interface. Students were asked to design the class `MultPractice` so that it implements two methods in the given `StudyPractice` interface. Students had to decide on an internal representation of the object that would allow them to implement two required methods so they are consistent with the given examples. In addition to demonstrating an understanding of class, constructor, and method header syntax, students had to correctly declare, initialize, access, and update instance variables. Students were expected to properly encapsulate their data members by declaring them as `private` and to properly expose the interface methods by declaring them to be `public`. To obtain a correct implementation, students had to demonstrate an understanding of the difference between returning a value and producing a side effect and how to convert from an integer to a string.

Sample: 2A
Score: 8

The response provides a correct header specifying that the class `implements` the interface, which earned point 1. Two instance variables are properly declared, with `private` access, for the operands of the multiplication operation, which earned point 2. The provided constructor has a correct header and correctly initializes the instance variables to the parameters, which earned points 3 and 4. For `getProblem` the response provides a correct header with the required `public` access specifier and earned point 5. The response implements the method properly by returning the result of creating the problem string from the instance variables, which earned points 6 and 7. For `nextProblem` the header is correct and point 8 was earned, but because the response fails to assign the incremented value to the instance variable, point 9 was not earned. The response earned 8 points.

Sample: 2B
Score: 4

The response provides a correct class header and a properly formed constructor header, which earned points 1 and 3. The response provides correct headers for the two interface methods, which earned points 5 and 8. The response fails to declare any instance variables. This was a commonly seen error and resulted in 4 points that were not earned: point 2 for the missing declarations, point 4 because the assignments in the constructor are to the local parameter variables, point 6 because the response incorrectly uses undeclared variables to build the problem string in `getProblem`, and point 9 because the response increments an undeclared variable in `nextProblem`. In `getProblem` the response prints the result rather than using `return`. This was another common confusion. Failing to `return` meant point 7 was not earned, but no additional usage point was deducted for the print, even though it produces a side effect and the method fails to return a value. The response earned 4 points.

Sample: 2C
Score: 3

The response improperly uses `extends` in the class header, so point 1 was not earned. The instance variables are not `private`, so point 2 was not earned. The types of the parameters in the constructor header are missing, so point 3 was not earned. The assignment statements inside the constructor overwrite the parameter variables with zeros, thus failing to initialize the instance variables, so point 4 was not earned. The response earned points 5 and 8 for the properly formed method headers. The expression to build the problem string in `getProblem` references undeclared variables, as does the increment in `nextProblem`, so points 6 and 9 were not earned. The return in `getProblem` earned point 7 because the constructed

Question 2 (continued)

string is built by concatenating values corresponding to the operands of the multiplication with the `"TIMES"` string. It was common for responses to earn point 7 for the return even if the expression did not use instance variables. The response earned 3 points.

© 2017 The College Board.