2022

AP CollegeBoard

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free-Response Question 1

- ☑ **Scoring Guidelines**
- ☑ **Student Samples**
- ☑ **Scoring Commentary**

Applying the Scoring Criteria

Apply the question scoring criteria 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.*

Question 1: Methods and Control Structures — 9 points

Canonical solution

(a) — **4 points**

```
public int getScore()
{
   int score = 0;

   if (levelOne.goalReached())
   {
      score = levelOne.getPoints();

      if (levelTwo.goalReached())
      {
         score += levelTwo.getPoints();

         if (levelThree.goalReached())
         {
            score += levelThree.getPoints();
         }
      }
   }

   if (isBonus())
   {
      score *= 3;
   }

   return score;
}
```

(b) — **5 points**

```
public int playManyTimes(int num)
{
   int max = 0;

   for (int i = 0; i < num; i++)
   {
      play();
      int score = getScore();
      if (score > max)
      {
         max = score;
      }
   }

   return max;
}
```

(a) `getScore`

	Scoring Criteria	Decision Rules	
1	Calls `getPoints`, `goalReached`, and `isBonus`	Responses **will not** earn the point if they • fail to call `getPoints` or `goalReached` on a `Level` object • call `isBonus` on an object other than `this` (use of `this` is optional) • include parameters	**1 point**
2	Determines if points are earned based on `goalReached` return values	Responses **can** still earn the point even if they • calculate the score total incorrectly • call `goalReached` incorrectly • fail to distinguish all cases correctly Responses **will not** earn the point if they • fail to use a nested `if` statement or equivalent	**1 point**
3	Guards update of score for bonus game based on `isBonus` return value	Responses **can** still earn the point even if they • triple the calculated score incorrectly • update the score with something other than tripling • call `isBonus` incorrectly Responses **will not** earn the point if they • use the `isBonus` return value incorrectly	**1 point**
4	Initializes and accumulates appropriate score (*algorithm*)	Responses **can** still earn the point even if they • call methods incorrectly, as long as method calls are attempted • fail to return the score (*return is not assessed*) Responses **will not** earn the point if they • calculate the score total incorrectly • triple the calculated score incorrectly	**1 point**
		Total for part (a)	**4 points**

(b) `playManyTimes`

	Scoring Criteria	Decision Rules	
5	Loops `num` times	Responses **can** still earn the point even if they • return early	**1 point**
6	Calls `play` and `getScore`	Responses **will not** earn the point if they • call either method on an object other than `this` (use of `this` is optional) • include parameters	**1 point**
7	Compares a score to an identified max or to another score	Responses **can** still earn the point even if they • make the comparison outside the loop • call `getScore` incorrectly • fail to call `play` between calls to `getScore`	**1 point**
8	Identifies the maximum score (*algorithm*)	Responses **will not** earn the point if they • fail to initialize the result variable • compare a score to an identified max or to another score outside the loop • fail to call `play` exactly once each time through the loop	**1 point**
9	Returns identified maximum score	Responses **can** still earn the point even if they • calculate the maximum score incorrectly Responses **will not** earn the point if they • assign a value to the identified maximum score without any loop or logic to find the maximum	**1 point**
		Total for part (b)	**5 points**

Question-specific penalties
None

Total for question 1 **9 points**

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ● Question 2 ○ Question 3 ○ Question 4 ○

Begin your response to each question at the top of a new page.

part a)

```
public int getScore()
{
 int sum = 0;
 if (levelOne.goalReached())
 {
   sum += levelOne.getPoints();
   if (levelTwo.goalReached())
   {
     sum += levelTwo.getPoints();
     if (levelThree.goalReached())
     {
       sum += levelThree.getPoints();
     }
   }
 }
 if (isBonus())
 {
   sum *= 3;
 }
 return sum;
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ● Question 2 ○ Question 3 ○ Question 4 ○

Begin your response to each question at the top of a new page.

part b)

```
public int playManyTimes(int num)
{
  int score = 0;
  int highScore = 0;
  for(int n=0; n<num; n++)
  {
    Game.play();
    score = Game.getScore();
    if(highScore < score)
    {
      highScore = score;
    }
  }
  return highScore;
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1	Question 2	Question 3	Question 4
●	○	○	○

Begin your response to each question at the top of a new page.

a)

```
public int getScore (){
    int score = 0;
    if (levelOne.goalReached()==true){
       score += levelOne.getPoints();
     }
    if (levelTwo.goalReached() == true){
      score += levelTwo.getPoints();
     }
   if (levelThree.goalReached() == true){
      score += levelThree.getPoints();
     }
   if (game.isBonus() == true){
      score = score * 3;
     }
  return score;
}
```

b)

```
public int playManyTimes (int num){
    int highScore = 0;
    int currentScore = 0;
    for (int i = 0; i <= num; i++){
       game.play();
       currentScore = game.getScore();
       if (currentScore > highScore){
          highScore = currentScore;
        }
       }
    return highScore;
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1	Question 2	Question 3	Question 4
●	○	○	○

Begin your response to each question at the top of a new page.

11a

```
public int getScore() {
    int sum = 0;
boolean l1 = (levelOne.goalReached());
boolean l2 = (levelTwo.goalReached());
boolean l3 = (levelThree.goalReached());
    if (l1 == true) {
      sum += levelOne.getPoints();
      }
    if (l2 == true) {
    sum += levelTwo.getPoints();
    }
    if (l3 == true) {
     sum += levelThree.getPoints();
      }
    if (game.isBonus() == true) {
       sum x = 3;
      }
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ● Question 2 ○ Question 3 ○ Question 4 ○

Begin your response to each question at the top of a new page.

```
//b
public int playManyTimes(int num){
for(int i=0; i<num; i++){
Game x = new Game();
x.play();
x.getScore();
}
```


Use a pencil only. Do NOT write your name. Do NOT write outside the box.

0049146

Question 1

Note: Student samples are quoted verbatim and may contain spelling and grammatical errors.

Overview

This question tested the student's ability to:

- Write program code to call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

More specifically, this question assessed the ability to use `Level` objects, call methods within and outside the current class, use nested `if` logic to calculate a correct score for each level depending on whether that level and all previous levels' goals were met, iterate a specific number of times to identify a maximum score, and use method return values in conditional expressions.

In part (a) students were asked to declare and initialize a numeric score variable and then call the `getPoints` and `goalReached` methods from the `Level` class on the instance variables `levelOne`, `levelTwo`, and `levelThree` in order to calculate a correct score for each level, depending on whether that level and all previous levels' goals were met. Students then had to evaluate the returned value from `isBonus` to determine if the score for the game is tripled before being returned.

In part (b) students were asked to declare and initialize a maximum value variable, iterate `num` times to call the `play` method, and compare the value returned from `getScore` to the identified maximum, replacing the maximum as needed in the loop after a correct comparison. The students then had to return the identified maximum score.

Sample: 1A
Score: 8

In part (a) point 1 was earned by using the correct syntax to call each method, without parameters, at least once. The methods of the `Level` class, `getPoints` and `goalReached`, are correctly called on the instance variables `levelOne`, `levelTwo`, and `levelThree`. The `isBonus` method is also called correctly. Point 2 was earned by using `goalReached` return values to determine if points should be included in the score. Responses can earn point 2 even if they do not correctly determine which points should be added to the score or if they do not call `goalReached` correctly. Point 3 was earned by correctly using the result of `isBonus` to determine if a bonus should be added to the score. Responses can still earn point 3 even if they do not calculate the bonus correctly or if they call `isBonus` incorrectly. (The `isBonus` call is assessed in point 1.) Point 4 was earned because the variable `sum` is initialized and used to calculate the correct score. Responses can still earn point 4 even if the `return` statement is omitted. Calculation of a correct score for levels 2 and 3 depends on whether or not the goal was reached on previous levels. When the condition used to check if a previous level's goal has been reached is missing or incorrect, or if the bonus is incorrectly computed, point 4 is not earned.

Question 1 (continued)

In part (b) point 5 was earned because the response correctly uses a `for` loop to iterate `num` times. Point 6 was not earned because the `play` and `getScore` method calls are incorrect. The response uses the class name, `Game`, to access the methods, which is incorrect since neither `play` nor `getScore` is a static method. The methods are members of the `Game` class, as is the method being written. Therefore, the correct method calls are `play()` and `getScore()`. Responses may include the `this` keyword in calls to these methods. Point 7 was earned because the response compares an identified maximum variable, `highScore`, with another score. This point can be earned even if the `getScore` method is called incorrectly or if the values being compared are incorrect. However, the values compared must be identified scores or maximum values. Point 8 was earned because the response implements the correct algorithm to identify the maximum score. The `play` method must be called exactly once each time through the loop, and the maximum score must be appropriately updated in order for this point to be earned. The response correctly initializes the maximum value variable, `highScore`, and updates that variable inside the loop as needed. Point 9 was earned by returning the identified maximum score stored in `highScore`. To earn this point, there must be logic to find the maximum or there must be a loop and calculations on the variable that is returned. The value returned does not have to be the correct maximum score, nor even a maximum, to earn this point.

Sample: 1B
Score: 5

In part (a) point 1 was not earned because the `isBonus` method is called incorrectly. The method call `Game.isBonus()` is incorrect since `isBonus` is not a static method and is a member of the same class as the method being written. Responses may include the `this` keyword in the method call (`this.isBonus()`). The `getPoints` and `goalReached` methods are called correctly, but all three method calls must be correct in order to earn this point. Point 2 was not earned. Even though the `goalReached` method is called correctly, there is no attempt to add points only if the goals of the current level and all previous levels are reached. In order to earn this point, there must be code that includes level 2's points in the score only if the goals of levels 1 and 2 are reached or code that includes level 3's points in the score only if the goals of levels 1, 2, and 3 are all reached. This point could have been earned if either the level 2 points or the level 3 points were correctly handled. Point 3 was earned because the `isBonus` return value is used to determine whether the bonus should be added. Point 4 was not earned because points from levels 2 and 3 are added to the score if the goal of the current level is reached, regardless of whether the previous levels' goals were reached. To earn this point, both the level 2 points and level 3 points must be correctly handled.

In part (b) point 5 was earned because the `for` loop starts at `1` and loops while `i <= num`, so the body of the loop is executed exactly `num` times. Point 6 was not earned because the response incorrectly uses the class name, `Game`, to access the `play` and `getScore` methods. Point 7 was earned because the `if` statement compares a score with the identified maximum variable, `highScore`. This point was earned, even though the response calls `getScore` incorrectly, because the incorrect method call is assessed in point 6. Point 8 was earned because the response correctly identifies the maximum score by declaring a maximum variable, `highScore`; initializing it

Question 1 (continued)

to zero; and updating it as needed in the loop after a correct comparison. Point 9 was earned by returning the identified maximum, `highScore`.

Sample: 1C
Score: 2

In part (a) point 1 was not earned because the `isBonus` method is called incorrectly. All method calls must be correct in order to earn this point. Point 2 was not earned. Even though the `goalReached` method is called correctly, there is no attempt to add points only if the goals of the current level and all previous levels are reached. Point 3 was earned because the `isBonus` return value is used to determine whether the bonus should be applied. The incorrect call to `isBonus` does not affect this point as the correctness of the method call was assessed in point 1. Point 4 was not earned because the sequential `if` statements in the response do not accumulate the appropriate score. Note that no points were deducted for the missing `return` statement in part (a).

In part (b) point 5 was earned because the `for` loop iterates exactly `num` times. Point 6 was not earned because the response calls the `play` and `getScore` methods on an object other than `this`. Point 7 was not earned because there is no comparison of a score with another score or with an identified maximum value. Point 8 was not earned because the response does not correctly identify a maximum score. There are no comparisons in the loop, and the loop creates a new `Game` object in each iteration, which eliminates the possibility of a maximum. Point 9 was not earned because there is no `return` statement and no variable to hold a score or calculated value that could be returned.