2019

AP® CollegeBoard

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free Response Question 4

☑ Scoring Guideline

☑ Student Samples

☑ Scoring Commentary

AP® COMPUTER SCIENCE A
2019 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.*

AP® COMPUTER SCIENCE A
2019 SCORING GUIDELINES

Question 4: Light Board

Part (a)	`LightBoard`	**4 points**

Intent: *Define implementation of a constructor that initializes a 2D array of lights*

+1 Creates a `new boolean[numRows][numCols]` and assigns to instance variable `lights`

+1 Accesses all elements in the created 2D array (*no bounds errors*)

+1 Computes the 40% probability

+1 Sets all values of 2D array based on computed probability

Part (b)	`evaluateLight`	**5 points**

Intent: *Evaluate the status of a light in a 2D array of lights*

+1 Accesses an element of `lights` as a `boolean` value in an expression

+1 Traverses specified `col` of a 2D array (*no bounds errors*)

+1 Counts the number of `true` values in the traversal

+1 Performs an even calculation and a multiple of three calculation

+1 Returns `true` or `false` according to all three rules

Question-Specific Penalties

-1 (z) Constructor returns a value

-1 (y) Destruction of persistent data

AP® COMPUTER SCIENCE A
2019 SCORING GUIDELINES

Question 4: Scoring Notes

Part (a) `LightBoard`			**4 points**
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Creates a `new boolean[numRows][numCols]` and assigns to instance variable `lights`		• initialize a local variable that is never assigned to `lights` • omit the keyword `new` • use a type other than `boolean`
+1	Accesses all elements in the created 2D array (*no bounds errors*)	• fail to create `lights` but assume `lights[numRows][numCols]`	
+1	Computes the 40% probability	• use `Math.random() <= .4`	• incorrectly cast to `int`
+1	Sets all values of 2D array based on computed probability	• only assign `true` values	• compute a single probability but use it multiple times • reverse the sense of the comparison when assigning
Part (b) `evaluateLight`			**5 points**
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Accesses an element of `lights` as a `boolean` value in an expression		• access `lights` as a type other than `boolean`
+1	Traverses specified `col` of a 2D array (*no bounds errors*)		
+1	Counts the number of `true` values in the traversal	• access too many or too few items in a single column • access a single row instead of a single column	• count an item more than once
+1	Performs an even calculation and a multiple of three calculation		• use `/` instead of `%`
+1	Returns `true` or `false` according to all three rules	• have an incorrect column count but use the correct logic	• fail to return a value in some case • implement counting loop more than once with one loop that is incorrect

Question 4: Light Board

Part (a)

```
public LightBoard(int numRows, int numCols)
{
   lights = new boolean[numRows][numCols];

   for (int r = 0; r < numRows; r++)
   {
      for (int c = 0; c < numCols; c++)
      {
         double rnd = Math.random();
         lights[r][c] = rnd < 0.4;
      }
   }
}
```

Part (b)

```
public boolean evaluateLight(int row, int col)
{
   int numOn = 0;

   for (int r = 0; r < lights.length; r++)
   {
      if (lights[r][col])
      {
         numOn++;
      }
   }

   if (lights[row][col] && numOn % 2 == 0)
   {
      return false;
   }
   if (!lights[row][col] && numOn % 3 == 0)
   {
      return true;
   }
   return lights[row][col];
}
```

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.

4Aa

(a) Write the constructor for the `LightBoard` class, which initializes `lights` so that each light is set to on with a 40% probability. The notation `lights[r][c]` represents the array element at row `r` and column `c`.

Complete the `LightBoard` constructor below.

```
/** Constructs a LightBoard object having numRows rows and numCols columns.
 *  Precondition: numRows > 0, numCols > 0
 *  Postcondition: each light has a 40% probability of being set to on.
 */
public LightBoard(int numRows, int numCols)
{
  lights = new boolean[numRows][numCols];
  for(int r=0; r<numRows; r++)
   {
    for(int c=0; c<numCols; c++)
     {
     double chance = Math.random();
      if(chance < .4)
       lights[r][c]=true;
     }
   }
}
```

Part (b) begins on page 18.

GO ON TO THE NEXT PAGE.

4Ab

Complete the `evaluateLight` method below.

```
/** Evaluates a light in row index row and column index col and returns a status
 *  as described in part (b).
 *  Precondition: row and col are valid indexes in lights.
 */
public boolean evaluateLight(int row, int col)
{
 if(lights[row][col] == true) //light on
   { int onCount=0;
    for(int r=0; r<lights.length; r++)
      {
        if (lights[r][col] == true)
           onCount++;
      }
    if (onCount % 2 == 0)
        return false;
   }
 if(lights[row][col] == false) // light off
   { int numOn=0;
    for(int rowOn=0; rowOn < lights.length; rowOn++)
     {
      if(lights[rowOn][col] == true)
          numOn++;
     }
    if (numOn % 3 == 0)
       return true;
   }
  boolean toReturn = lights[row][col];
  return toReturn;
}
```

GO ON TO THE NEXT PAGE.

4Ba

(a) Write the constructor for the `LightBoard` class, which initializes `lights` so that each light is set to on with a 40% probability. The notation `lights[r][c]` represents the array element at row `r` and column `c`.

Complete the `LightBoard` constructor below.

```
/** Constructs a LightBoard object having numRows rows and numCols columns.
 *  Precondition: numRows > 0, numCols > 0
 *  Postcondition: each light has a 40% probability of being set to on.
 */
public LightBoard(int numRows, int numCols){

    boolean [] [] Display = new int [numRows] [numCols];

      for ( int i = 0; i < numRows; i++) {
        for(int j = i; j < numCols; j++) {
            Display[i] [j] = (Math. random() > 0.6);
        }

      }

  }
```

Part (b) begins on page 18.

GO ON TO THE NEXT PAGE.

4Bb

Complete the `evaluateLight` method below.

```
/** Evaluates a light in row index row and column index col and returns a status
 *  as described in part (b).
 *  Precondition: row and col are valid indexes in lights.
 */
public boolean evaluateLight(int row, int col)
```

```
int count = 0;
int [] countOn = new int [lights[0].length];
for(int i = 0; i < lights.length; i++){
    Count = 0;
    for(int j = 0; j < lights[i].length; j++){
            if(lights[i][j] == true){
            count ++;
        }
            countOn [i] = count);

    }

}

if(lights[row][col]      == true){
    if( countOn[row]%2 == 0){
        return false;
    }
    return true;
}
ie(lights[row][col] == false){
    if(countOn[row]%3 == 0]{
        return true;
    }
    return false;
}
```

GO ON TO THE NEXT PAGE.

4Ca

(a) Write the constructor for the `LightBoard` class, which initializes `lights` so that each light is set to on with a 40% probability. The notation `lights[r][c]` represents the array element at row `r` and column `c`.

Complete the `LightBoard` constructor below.

```
/** Constructs a LightBoard object having numRows rows and numCols columns.
 *  Precondition: numRows > 0, numCols > 0
 *  Postcondition: each light has a 40% probability of being set to on.
 */
public LightBoard(int numRows, int numCols)
{ int total= numRows * numCols;
   int prob= total * (.4);
 lights = [numRows][numCols];
  for(int r=0; r< lights.length; r++)
  {
    for(int c=0; c< lights[0].length; c++)
    {
       if (prob >0)
       {
        lights[r][c] = true;
       }
       else
        lights[r][c] = false;
       }
    }
  }
}
```

Part (b) begins on page 18.

GO ON TO THE NEXT PAGE.

4Cb

Complete the evaluateLight method below.

```
/** Evaluates a light in row index row and column index col and returns a status
 *  as described in part (b).
 *  Precondition: row and col are valid indexes in lights.
 */
public boolean evaluateLight(int row, int col)
{
 int lightcount;


  for(int i=0; i< col ; i++)
  {
    if ( lights [i] [col] = true)
    { lightcount ++;
    }
  }

  if ( lights[row][col] = on)
  {
    if( lightcount %2 = 0)
    { return false;
    }
    else
    { return lights [row] [col] ;
    }
 }
 else if( lights[row][col] = off)
 {
  if ( lightcount %3 = 0)
   {
     return true;
   }
   else
   {
     return lights[row][col];
   }
 }

}
```

GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A 2019 SCORING COMMENTARY

Question 4

Overview

This question tested the student's ability to:

- Write program code to create objects of a class and call methods;
- Write program code to satisfy methods using expressions, conditional statements, and iterative statements; and
- Write program code to create, traverse, and manipulate elements in 2D array objects.

This question involved the creation and manipulation of a two-dimensional array of `boolean` values. Students were expected to implement a constructor and a method of the enclosing `LightBoard` class.

In part (a) students were asked to construct a two-dimensional array and initialize the values in the two-dimensional array based on a computed probability. Students were expected to be able to use the constructor's parameters to construct a two-dimensional `boolean` array with the correct number of rows and columns. Once the two-dimensional array was constructed, students were expected to write nested loops to access each item. For each item, the students were expected to use `Math.random()` to compute a probability for the purpose of choosing which `boolean` value should be assigned to the item.

In part (b) students were given parameters representing a `row` and `col` and were asked to evaluate the status of the light in the two-dimensional array at `lights[row][col]`. Students were expected to write a loop to access each item in the given column and use an accumulator to count the number of lights that are set to `true` in that column. Students were then expected to return a `boolean` value based on three rules: (1) If the light is on, return `false` if the number of lights in its column that are on is even, including the current light. (2) If the light is off, return `true` if the number of lights in its column that are on is divisible by three. (3) Otherwise, return the light's current status. To implement the rules, students were expected to perform an even calculation and a multiple of three calculation.

Sample: 4A
Score: 9

In part (a) the response correctly constructs the `boolean` 2D array and assigns it to the instance variable `lights`. Therefore point 1 was earned. All elements in the created 2D array are accessed, so point 2 was earned. The response earned point 3 by testing if `Math.random()` returns a value less than `0.4`, which occurs 40% of the time. All values in the 2D array are set to the correct `boolean` values based on the computed probability. The response takes advantage of the fact that, by default, values in a `boolean` array are set to `false` when the array is first created. As a result, the response earned point 4. Part (a) earned 4 points.

In part (b) the response earned point 5 by accessing an element of the `lights` array as a `boolean` value in an expression. Point 6 was earned because the response traverses the specified `col` of the 2D array without a bounds error. The number of `true` values is counted during the traversal. Therefore point 7 was earned. The response performs correct even and multiple of three calculations. For this reason, point 8 was earned. Because the response returns `true` or `false` correctly according to all three rules, the response earned point 9. Part (b) earned 5 points.

Question 4 (continued)

Sample: 4B
Score: 6

In part (a) the response uses an invalid data type when creating the 2D array and does not assign the 2D array to the instance variable `lights`. Therefore point 1 was not earned. Because the inner loop starts at index `i`, not all elements in the created 2D array are accessed. Therefore point 2 was not earned. The response earned point 3 by testing if the value returned by `Math.random()` is greater than `0.6`, which is true 40% of the time. All accessed elements in the 2D array are set to the correct `boolean` values, and the response earned point 4. Part (a) earned 2 points.

In part (b) the response earned point 5 by accessing an element of the `lights` array as a `boolean` value in an expression. Point 6 was not earned because the response does not use the variable `col` when traversing the 2D array. The number of `true` values is counted during the traversal, even though the response traverses each row instead of a single column. Therefore point 7 was earned. The response performs correct even and multiple of three calculations. For this reason, point 8 was earned. Because the response returns `true` or `false` correctly according to all three rules, the response earned point 9. Part (b) earned 4 points.

Sample: 4C
Score: 3

In part (a) the response does not specify the data type or use the keyword `new` when creating the 2D array. Therefore point 1 was not earned. All elements in the created 2D array are accessed, so point 2 was earned. Point 3 was not earned because the response does not use a pseudorandom value to compute probability. Because the response only attempts to compute a single probability and use it multiple times to set all values of the 2D array, point 4 was not earned. Part (a) earned 1 point.

In part (b) the response earned point 5 by accessing an element of `lights` as a `boolean` value in an expression. Point 6 was not earned because the response uses `col` instead of the number of rows in `lights` to traverse the 2D array. The response omits the initialization of the accumulator to 0. Therefore point 7 was not earned. The response performs correct even and multiple of three calculations. For this reason, point 8 was earned. The response did not earn point 9 because it does not compare `lights[row][col]` to a `boolean` value, failing to correctly implement all three rules. Part (b) earned 2 points.