2021

AP CollegeBoard

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free Response Question 4

- ☑ **Scoring Guideline**
- ☑ **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 4: 2D Array — 9 points

Canonical solution

(a) — **3 points**

```
public static boolean isNonZeroRow(int[][] array2D, int r)
{
   for (int col = 0; col < array2D[0].length; col++)
   {
      if (array2D[r][col] == 0)
      {
         return false;
      }
   }
   return true;
}
```

(b) — **6 points**

```
public static int[][] resize(int[][] array2D)
{
   int numRows = array2D.length;
   int numCols = array2D[0].length;

   int[][] result = new int[numNonZeroRows(array2D)][numCols];
   int newRowIndex = 0;

   for (int r = 0; r < numRows; r++)
   {
      if (isNonZeroRow(array2D, r))
      {
         for (int c = 0; c < numCols; c++)
         {
            result[newRowIndex][c] = array2D[r][c];
         }
         newRowIndex++;
      }
   }
   return result;
}
```

(a) `isNonZero`

	Scoring Criteria	Decision Rules	
1	Compares an item from `array2D` with `0`	Responses **will not** earn the point if they fail to attempt the comparison, even if they access an item from `array2D`	**1 point**
2	Accesses every item from row `r` of 2D array (*no bounds errors*)	Responses **can** still earn the point even if they return early from an otherwise correctly-bounded loop	**1 point**
3	Returns `true` if and only if row contains no zeros	Responses **can** still earn the point even if they process a column of the 2D array rather than a row Responses **will not** earn the point if they fail to return a value in some cases	**1 point**
		Total for part (a)	**3 points**

(b) `resize`

	Scoring Criteria	Decision Rules	
4	Calls `numNonZeroRows` and `isNonZeroRow`	Responses **can** still earn the point even if they fail to use or store the return value Responses **will not** earn the point if they • include incorrect number or type of parameters • call methods on an object or class other than `ArrayResizer`	**1 point**
5	Identifies rows with no zeros (*in the context of an* `if`)	Responses **can** still earn the point even if they call `isNonZeroRow` incorrectly, if the row being tested is clearly identified (index or reference)	**1 point**
6	Declares and creates a new 2D array of the correct size	Response **will not** earn the point if they transpose the dimensions of the created array	**1 point**
7	Maintains an index in the new array	Responses **will not** earn the point if they • fail to declare, initialize, and update a different index • maintain the index in a way that overwrites, skips, or duplicates rows	**1 point**
8	Traverses all necessary elements of `array2D` (*no bounds errors*)	Responses **can** still earn the point even if they • cause a bounds error by declaring and creating a new 2D array of an incorrect size • fail to maintain an index in the new array correctly, resulting in a bounds error • fail to access individual elements in a nested loop, if they access each row as an entire row Responses **will not** earn the point if they transpose coordinates, leading to a bounds error and/or copying columns	**1 point**
9	Copies all and only rows identified as having no zero elements into the new array	Responses **can** still earn the point even if they • copy a reference • identify rows incorrectly, if the logical sense can be determined and is correct • copy columns instead of rows, consistent with the dimensions of the created 2D array	**1 point**

		Responses **will not** earn the point if they • remove or overwrite data from `array2D` (instead of or in addition to copying it to the new array) • reverse the logical sense of which rows to copy	
		Total for part (b)	**6 points**
	Question-specific penalties		
	-1 (u) Use `array2D[].length` to refer to the number of columns in a row of the 2D array		
		Total for question 4	**9 points**

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

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

a.
```
public static boolean isNonZeroRow(int[][] array2D, int r){
    for(int n : array2D[r]){
        if(n == 0){
            return false;
        }
    }
    return true;
}
```

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

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

b.

```
public static int [][] resize (int [][] array2D) {

    int [][] result = new int [numNonZeroRows (array2D)][array2D[0].length];
    int index = 0;
    for (int r = 0; r < array2D.length; r++) {
        if (isNonZeroRow (array2D, r)) {
            for (int c = 0; c < array2D[r].length; c++) {
                result [index] [c] = array2D [r][c];
            }
            index++;
        }
    }
    return result;
}
```

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

```
public static boolean isNonZeroRow(int[][] array2D, int r)
{
   int countZ = 0;
   for(int row[]: array2D)
      {
        for(int cell : row)
           {
             if(cell == 0)
                {
                  countZ++;
                }
           }
        }
      return countZ == 0;
}
```

Q4 Sample B 2 of 2

Question 1 | Question 2 | Question 3 | Question 4

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

```
public static int[][] resize(int[][] array2D)
{
   int nonZeroRows = numNonZeroRows(array2D);
   int[][] zeros = new int[nonZeroRows][array2D[0].length];
   for (int r = 0; r < array2D.length; r++)
      {
         if (isNonZeroRow(array2D, r) == false)
            {
              zeros[r] = array2D[r];
            }
      }
   return zeros;
}
```

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

```
public static boolean isNonZeroRow(int [][] array2D, int r)
{ int count = 0;
  for (int i : array2D.get(r))
  {
    if (array2D[r][i] != 0)
    {
      count = 0;
    }
  }
  if (count == array2D[r].size())
  {
    return true;
  }
  else
  {
    return false;
  }
```

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

```
public static int[][] resize(int[][] array2D)
{  int[][] output = {};
   for (i=0, i < array2D.size( ), i++)
   {
     if(array2D.isNonZeroRow(array2D, i) == true)
     {
        output += array.get(i) ;
     }
   }
   return output;
}
```

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.
- Write program code to create, traverse, and manipulate elements in 2D array objects.

This question involved the manipulation of a two-dimensional array of `int` values. A static class that included three methods, one written in part (a), one written in part (b), and a helper method, was provided.

In part (a) students were asked to write a `boolean` method, `isNonZeroRow`, which returned `true` if and only if all elements in row `r` of a two-dimensional array, `array2D`, are not equal to zero. Students were expected to be able to use the parameters `r` and `array2D` to traverse the given row in the two-dimensional array and determine if there were any zeros in that row.

In part (b) students were asked to write a method called `resize`, which returned a new two-dimensional array containing only rows from `array2D` with all nonzero values. The elements in the new array should appear in the same order as the order in which they appeared in the original array. Students were expected to create a new two-dimensional array with the correct dimensions. Students were expected to use the method in part (a), `isNonZeroRow`, and the helper method, `numNonZeroRows`, provided in the class framework. Students were then expected to identify the rows that were nonzero rows and copy them to the new two-dimensional array.

Sample: 4A
Score: 9

In part (a) point 1 was earned because the `if` statement inside the loop compares an integer `n` from a cell in the specified row of `array2D` with zero. To earn this point, the compared element must be from `array2D`, the given parameter of the method, and there must be a comparison with zero. Point 2 was earned because an enhanced `for` loop is used to iterate through all the elements of row `r` without a bounds error. This point focuses on iterating over a 2D array, accessing all elements of the row specified by the parameter, without going out of bounds. The point can still be earned even with an early `return`. Point 3 was earned because `true` is returned only when there are no zeros in the given row. This point focuses on the logic of determining whether a condition exists and returning the correct result at the correct place in the method.

In part (b) point 4 was earned because the calls to the `numNonZeroRows` and `isNonZeroRow` methods are correct. This point focuses on correctly calling methods with the proper parameters. This point can still be earned even if the returned values from the methods are not used. Point 5 was earned because the `isNonZeroRow` method is called in the context of an `if` statement to check rows for zeros. This point focuses on using an `if` statement to check whether a condition is satisfied before processing a row of the 2D array. This point can still be earned even if the method call has errors, but the row must be clearly indicated by either index or reference. Point 6 was earned because a 2D array is correctly declared and instantiated with the proper dimensions to handle only the rows identified with no zeros from the original 2D array. This point focuses on creation of a 2D array with the correct size. The number of rows changes based on the number of nonzero rows returned by the `numNonZeroRows` method. The number of columns remains the same as the number of columns in the original 2D array (`array2D[0].length`). Point 7 was earned because an index

Question 4 (continued)

variable is declared, initialized to zero, and updated after a nonzero row is put in the new 2D array. This point focuses on correctly maintaining a counter and using it as the row index of the new 2D array, making sure that the copy does not overwrite, skip, or duplicate rows in the new 2D array. Point 8 was earned because `array2D` is traversed to determine whether each row is a nonzero row. This point focuses on traversing the 2D array. The row and column bounds used are `array2D.length` and `array2D[r].length`. (Note that `array2D[0].length` would also be a correct bound.) Point 9 was earned because only rows identified as nonzero rows are stored in the new 2D array. This point focuses on copying elements from one array to another array with correct logic used in selecting rows to copy. The nonzero rows must be identified and copied to the new 2D array without removing or overwriting data from `array2D`.

Sample: 4B
Score: 6

In part (a) point 1 was earned because an element of `array2D` is compared to zero. This response uses enhanced `for` loops to traverse the elements. Point 2 was earned because every element in the given row is accessed. Point 3 was not earned because the response counts zeros in `array2D` instead of counting zeros in only the given row. The value `true` is only returned when all values in the 2D array are zero, instead of returning `true` if the specified row has no zeros.

In part (b) point 4 was earned because calls to the `numNonZeroRows` and `isNonZeroRow` methods are correct. Point 5 was earned because the value returned by `isNonZeroRow` is checked in the context of an `if` statement and the row to check is clearly identified. Point 6 was earned because a 2D array is correctly declared and instantiated with the proper dimensions to handle only the rows identified with no zeros from the original 2D array. Point 7 was not earned because no index variable is declared to maintain the row of the new 2D array for the copy process. Point 8 was earned because a loop is used to traverse the elements of `array2D` with proper bounds. This solution only looks at each row. Point 9 was not earned because the condition does not correctly identify nonzero rows and copies rows with zeros to the new 2D array.

Sample: 4C
Score: 2

Point 1 was earned because an element of `array2D` is correctly compared to zero. Point 2 was not earned because the loop does not access every element in row `r` in the 2D array. The enhanced `for` loop attempts to access a row and iterates through the elements of the array, but the condition incorrectly uses `i` as an index to access the 2D array; `i` instead contains an element of the array. Using an element as an index could cause the access to go out of bounds. The discussion of the penalty point deducted for this response explains how the incorrect syntax `array.get(r)` is assessed. Point 3 was not earned because the check to determine if rows contain zeros returns `false` every time and the `count` variable used to count nonzero rows is not incremented.

In part (b) point 4 was not earned because there is no call to the `numNonZeroRows` method and the `isNonZeroRow` method is called on `array2D`. Point 5 was earned because the value returned by `isNonZeroRow` is checked in the context of an `if` statement and the row to check is clearly identified. This point can still be earned even if the method call is incorrect. Point 6 was not earned because the new 2D array is declared but is not instantiated with the correct dimensions. Point 7 was not earned because no index variable is declared or used to specify the row of the new 2D array in the copy process. Point 8 was earned because a loop is used to traverse the rows of the `array2D` with proper bounds. The discussion of the penalty point deducted for this response explains how the incorrect syntax `array.get(i)` is assessed.

Question 4 (continued)

Point 9 was not earned because the attempt to copy the identified row to the new 2D array does not use correct indexing to store the row in the new 2D array.

Penalty point: 1 point was deducted (**–1v**) to address the array/collection confusion (`[] get`) in both part (a) and part (b). A complete list of 1-point penalty errors is found on page 1 of the Scoring Guidelines. Any such penalty may be assessed only once for a question and only if the response otherwise earns points via the question rubric. Here, although point 2 was not earned for other reasons, point 8 can still be earned because the penalty has been assessed. If *both* points would *not* be earned for reasons other than array/collection access confusion, the penalty point would not be assessed.