AP® Computer Science A 2015 Scoring Guidelines

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 1: Diverse Array

Part (a)	`arraySum`	**2 points**

Intent: *Compute and return sum of elements in 1D array* `arr`*, passed as parameter*

+1 Accesses all elements of `arr`, (*no bounds errors on* `arr`)

+1 Initializes, computes, and returns sum of elements

Part (b)	`rowSums`	**4 points**

Intent: *Compute and return 1D array containing sums of each row in the 2D array* `arr2D`*, passed as parameter*

+1 Constructs correctly-sized 1D array of ints

+1 Accesses all words in `arr2D` (*no bounds errors on* `arr2D`)

+1 Computes sum of row in `arr2D` using `arraySum` and assigns to element in 1D array

+1 Returns 1D array where kth element is computed sum of corresponding row in 2D array for all rows

Part (c)	`isDiverse`	**3 points**

Intent: *Determine whether* `arr2D`*, passed as parameter, is diverse*

+1 Computes and uses array of row sums from `arr2D` using `rowSums`

+1 Compare all and only pairs of row sums for equality (*No bounds errors on row sums array; point not awarded if no adjustment when compares any row sum with itself*)

+1 Returns `true` if all compared row sums are different and `false` otherwise (*point not awarded for immediate return*)

Question-Specific Penalties

-1 (g) Uses `getLength/getSize` for array size

-1 (y) Destruction of persistent data (`arr` *or* `arr2D`)

Question 1: Diverse Array

Part (a):

```
public static int arraySum(int[] arr){
       int sum=0;
       for (int elem : arr){
            sum += elem;
       }
       return sum;
}
```

Part (b):

```
public static int[] rowSums(int[][] arr2D){
        int [] sums=new int[arr2D.length];
        int rowNum=0;
        for(int[] row : arr2D){
            sums[rowNum]=arraySum(row);
            rowNum++;
        }
        return sums;
    }
```

Part (c):

```
public static boolean isDiverse(int[][] arr2D){
        int [] sums=rowSums(arr2D);
        for (int i=0; i < sums.length; i++){
            for (int j=i+1; j < sums.length; j++){
                if (sums[i]==sums[j]){
                    return false;
                }
            }
        }
        return true;
    }
```

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;
    }

}
```

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.

AP® COMPUTER SCIENCE A
2015 SCORING GUIDELINES

Question 3: Sparse Array

Part (a)	`getValueAt`	**3 points**

Intent: *Return the value at row index* `row` *and column index* `col` *in sparse array*

+1 Accesses all necessary elements of `entries` (*No bounds errors*)

+1 Identifies element of `entries` at row index `row` and column index `col`, if exists

+1 Returns identified value or returns 0 if no entry exists in `entries` with row index `row` and column index `col`

Part (b)	`removeColumn`	**6 points**

Intent: *Remove column* `col` *from sparse array*

+1 Decrements `numCols` exactly once

+1 Accesses all elements of `entries` (*No bounds errors*)

+1 Identifies and removes entry with column index `col`

+2 Process entries with column index > `col` within loop

- **+1** Creates new `SparseArrayEntry` with current row index, column index -1, current value

- **+1** Identifies and replaces entry with column index > `col` with created entry

+1 On exit: All and only entries with column index `col` have been removed and all and only entries with column index > `col` have been changed to have column index -1. All other entries are unchanged. (*Minor loop errors ok*)

Question-Specific Penalties

-2 (t) Consistently uses incorrect name instead of `entries`

-1 (u) Directly accesses private instance variables in `SparseArrayEntry` object

Question 3: Sparse Array

Part (a):

```
public int getValueAt(int row, int col){
       for (SparseArrayEntry e : entries){
          if (e.getRow() == row && e.getCol() == col){
             return e.getValue();
          }
       }
       return 0;
}
```

Part (b):

```
public void removeColumn(int col){
       int i=0;
       while (i < entries.size()){
            SparseArrayEntry e = entries.get(i);
            if (e.getCol() == col){
                entries.remove(i);
            } else if (e.getCol() > col){
                entries.set(i, new SparseArrayEntry(e.getRow(),
                                                    e.getCol()-1,
                                                    e.getValue()));
                i++;
            } else {
                i++;
            }
      }
      numCols--;
}
```

AP® COMPUTER SCIENCE A
2015 SCORING GUIDELINES

Question 4: Number Group

Part (a)	Interface: `NumberGroup`	2 points

Intent: *Define interface to represent a number group*

+1 `interface NumberGroup` (*point lost if visibility private*)

+1 `boolean contains(int` *`num`*`);`
(*point lost if visibility not public or extraneous code present*)

Part (b)	Class: `Range`	5 points

Intent: *Define implementation of* `NumberGroup` *representing a range of numbers*

+1 `class Range implements NumberGroup` (*point lost if visibility private*)

+1 Declares appropriate `private` instance variable(s)

+1 Uses correct constructor header

+1 Initializes instance variables within constructor using parameters
(*point lost if bounds errors occur in container use*)

+1 Computes and returns correct value from `contains`
(*point lost for incorrect method header*)

Part (c)	`contains`	2 points

Intent: *Determine whether integer is part of any of the member number groups*

+1 Calls `contains` on elements of `groupList` in context of loop *(no bounds errors)*

+1 Computes and returns correct value

Question-Specific Penalties

-1 (s) Inappropriate use of `static`

Question 4: Number Group

Part (a):

```
public interface NumberGroup
{
    boolean contains(int num);
}
```

Part (b):

```
public class Range implements NumberGroup
{
    private int min;
    private int max;

    public Range(int min, int max)
    {
        this.min=min;
        this.max=max;
    }

    public boolean contains(int num){
        return num >= min && num <= max;
    }
}
```

Part (c):

```
public boolean contains(int num){
       for (NumberGroup group : groupList){
           if (group.contains(num)){
               return true;
           }
       }
       return false;
}
```