Computer Science, Claremont McKenna College
CS51 - Introduction to Computer Science, Spring 2015

Problem Set 7 Due: 11:55 PM, Mar 29, 2015

General Instructions

Please carefully read and follow the directions exactly for each problem. Files and classes should be named exactly as directed in the problem (including capitalization!) as this will help with grading.

You should create your programs using your preferred text-editor, the Eclipse text editor, or jGrasp. Do not use a word processor such as Microsoft Word, WordPad, Google Docs, Apple's Pages, etc...

Your programs should be formatted in a way that's readable. In other words, indent appropriately, use informative names for variables, etc… Points will be deducted if your indenting style makes it difficult to follow your code. If you are uncertain about what is a readable style, look at the examples from class as a starting point for a reasonable coding style. .

Your programs should compile and run without errors. Please do not submit programs that do not compile! It's better to submit partial implementation that compiles as oppose to implementations that do not compile.

At the top of each file you submit, you should put your name and your email as comments.

This homework set should be submitted via Sakai using the Assignment Menu option (on the left pane). You should only submit the requested ".java" files. Do not submit ".class" files or Eclipse-specific project files. Finally, please do not submit files using Sakai's Dropbox Menu.

Turn in the following file(s):

```
Range.java
Mode.java
Unique.java
CollapsedArray.java
```

Directions

In this problem set, you will practice working with arrays. In each problem, you should supply a main method that calls the specified methods with several test cases and print some informative messages indicating the return value of the method. When a method is specified, be sure to write a method that returns the desired return value. Do not use a Scanner to prompt the user for input.

Problem 1

In a class called `Range`, write a method call `getRange` that accepts an array of double numbers as a parameter and returns the range of numbers in the array where the range is defined as the largest value in the array minus the smallest value. For example, if the array `a1` stores `[12.3, 22.2, 14.4, 33.1]` and the array `a2` stores `[1.4, 4.3, -7.0, 19.5, -24.2 46.2]`, the calls `getRange(list1)` and `getRange(list2)` should return `20.8` and `70.4` respectively. You may assume that the array has at least 1 element. You may not use Arrayy.sort or any other routine to sort the array. (Note: in your loop to find the min and max, be careful that you initialize your "current" min and max variable to a value that's actually in the array.)

Problem 2

In a class called `Mode`, write a method call `getMode` that accepts an array of integer numbers as a parameter and returns the mode of the array where the mode is defined as the most frequently occurring value in the array. In statistics the mode given a set of data is the value that appears most often in the set. You may assume that values of the array are limited to the range 0 to 100 (inclusive). Break ties by choosing the lower value. For example, for the array `[27, 15, 7, 15, 27, 99]`, the method would return 15. (Hint: one possible solution is to create a separate "tally" array to count the occurrences of each integer in the input array).

Problem 3

In a class called `Unique`, write a method call `isUnique` that accepts an array of integer numbers as a parameter and returns `true` if all the integers are unique (no duplicates) and `false` otherwise. For example, if the array `a1` stores `[3, 8, 12, -48, 46, -3]`, the call `isUnique(a1)` would return `true` and if the array `a2` stores `[4, 7, 3, 9, 12, -47, 3]` the call `isUnique(a2)` would return false since the value 3 appears twice. (Hint: since the array of integers is not constrained to a small range, using a tally array is not feasiable. Instead, consider using nested for loops.)

Problem 4

In a class called `CollapsedArray`, write a method call `getCollapsedArray` that accepts an array of integers as a parameter and returns a new array containing the result of replacing each pair of integers with the sum of that pair. For example, if the array `a1` stores `[3, 8, 2, -8, 6, -3, 1, 9]`, the call `getCollapsedArray(a1)` would return the array `[11, -6, 3, 10]`. If the array length is odd, the last element is not collapsed and is simply copied to the new array. For example, if the array `a2` stores `[3, 8, 2, -8, 6, -3, 1, 9, 6]`, the call `getCollapsedArray(a2)` would return the array `[11, -6, 3, 10, 6]`. You should not modify the input array. When testing, print out both the input array and the new array.