CS64 Labs 02 – 04 Practice with Arrays and Loops

The objective of these exercises is to get experience working with arrays and loops. You will use the file server `ellafitz.mills.edu` to access packages of code that I supply for various exercises and to submit your work electronically to the appropriate dropbox on `ellafitz.mills.edu`. You will also practice using an application program without a GUI to write and test static methods. The code you have been provided generates 100 random `int` values in the range 0–49 and stores them in an array.

1. Drag the folder `forLabs02Through04` from the directory `Users/cs64/Public/"Lab Code"` on `ellaFitz` onto the desktop to make a copy that you can use.
2. Build an Eclipse project that uses the package of Java code you just copied. Remember to click the radio button under `Project Layout` to indicate that you want to use the project folder for your Java source code and the class files that the compiler will create.
3. In order to display your name in the output, use the editor to add a line of code, substituting your name for mine and the appropriate lab number, positioned as indicated below.

```
public static void main (String args [] ){
   System.out.println("CS 64 Lab 02 Barbara Li Santi");
   //allocate array to hold ints
```

4. Run the application. You should see the line of output displaying your name.

For Labs 02 – 04 you will write and test methods that perform the tasks given below. You may invoke any method you have written in any subsequent method. You have also been provided methods named `displayMyInts` and `bubbleSortMyInts` that you can use to display the values in the array 10 per line and to sort the values in the array `myInts` if you find that to be convenient.

Note the following 10 specifications about your lab submissions:

1. Please submit hard copies of your code and the output. Your name, the due date, and the lab number should be clearly indicated in the Java code **and** in the output.
2. Please submit a copy of your Eclipse project folder (whose name clearly indicates your name and the lab number) to the appropriate `Dropbox` within `cs64/Public` on `ellaFitz`.
3. **Within your code, please write a comment before each method giving the task letter as indicated below and write comments in the `main` method indicating the method invocations needed to accomplish each lettered task.**
4. To make your output easier to follow, use `System.out.println` in the `main` method to display sentences describing the output that follows.
5. For each method you write, answer the following question and display the answer in your output using `System.out.println`: Why is a `for` loop or a `while` loop better suited to accomplishing this task that involves moving through the array?

6. **On the hard copy of your output, annotate your output by hand to indicate which parts of the output correspond to tests for each lettered task.**
7. Present methods in alphabetical order in the `.java` file.
8. Make sure you display the contents of any array that has been modified.
9. You may use additional variables and syntactic constructs as needed in conjunction with `System.out.println` to show that your methods work correctly.
10. Please see the course handout to remind yourself about the criteria for grading programming assignments.

Lab 02 Due Tuesday, 4 Febuary 2014 by 5:00 p.m.

Write and test methods for Exercise 10.1 parts a through h.

Lab 03 Due Tuesday, 11 February 2014 by 5:00 p.m.

Write and test methods for Exercise 10.1 parts i through q.

Lab 04 Due Tuesday, 18 February 2014 by 5:00 p.m.

Write and test methods for tasks r through w described below.

Tasks: Write a method that…

r. Displays the elements in the array `myInts` until their sum exceeds an `int` parameter. You should display the element that causes the sum to exceed the parameter.
s. Determines and returns the index that is the first occurrence of an `int` parameter in the array `myInts`.
t. Determines and returns the index of the first occurrence of an element in the array `myInts` that is greater than an `int` parameter.
u. Determines and returns the index that is the last occurrence of an `int` parameter in the array `myInts`.
v. Determines and returns the index of the last occurrence of an element in the array `myInts` that is greater than an `int` parameter.
w. Assuming that the array `myInts` has **not** been sorted, determines and returns the first element of the array that is not in ascending order.