Programming, and Data Structures

Workbook 3

This worksheet WILL be graded. Be sure to use the Class template design provided in Average.java from Workbook0.

Today’s lab is about searching for data in linear data structures. We are going to implement what we saw in the lecture notes. For each part, create an array of integer data yourselves. For Part3.java, make sure that your array is already sorted (remember, Binary Search only works on sorted data).

Before you begin.

1. Ensure that your workspace is in a folder which is backed up to the web/ network e.g. college network drive, google drive. You may like to have it in the following folder structure …/GriffithCollege/PDS/workspace
2. Load Eclipse selecting the appropriate workspace
3. Make a new java project called Workbook03
4. Make a new package in this project called workbook03
5. Make a 3 new java files with the name provided below.

**/\***

**\* COMMENT YOUR CODE!!!!!!! SERIOUSLY. I CANNOT GIVE MARKS IF YOU DON’T**

**\* EXPLAIN WHAT YOU ARE DOING.**

**\*/**

**Tasks**

1. **LinearSearch.java** Create a class meeting the following specification
   * A public class with the same name as the java file.
   * An private integer array as member data
   * A public constructor with the same name as the java class.
   * A public method called *toString*  which when called will create a string showing the private member data of the class.
   * A public method called *search* with accepts a variable valueToBeFound as a parameter. It will search the integer array for the value and return either the location if it finds it or -1 if it does not. You may like to call the *toString* method to show how the private member data of the class is being updated.
   * A public main method which
     1. Creates an object of the class
     2. Calls the methods of the class on this object i.e. searches for a specified value in the array
     3. Prints out the changes made to the objects
   * Do part i and ii a few times.
2. **SelfOrganisingSearch.java** Let’s implement an example of Self Organising Search. Rewrite your *search()* method, but this time, each time an item is found, move it to the first position in the list (and move everything else back one space to make room). Test your search algorithm by printing the contents of the array every time you do a search to see how it changes. Your array should print horizontally like follows using a toString method which you will write.

A = 1 2 3 4 5 (rather than in vertically)

1. **Binarysearch.java** In Part3.java, change your earlier search approaches to build *binarysearch().* Make a new file Your *binarysearch()* method should implement the algorithm for Binary Search discussed in the lecture notes. Note; make sure that the data is sorted beforehand! You can either use one of the previous sorting techniques or you can generate a sorted array to begin with.