**CSE1322 – Java Lab 1B – Search**

*This lab may contain intentional minor errors; they should be reasonably easy to find and exist so that you must think as you go through the lab.  Like many things in life, "trust but verify."  Ideas will always be correct; only minor code and syntax may be incorrect.*

**Lab 2: Searching**

**Objectives**

* Write programs in Java to search a 1 dimensional array using a linear search and a binary search.

For this lab, you should work alone.  You will use Eclipse as your IDE.

**Directions**

This lab has two parts:

1. Writing a simple linear search application

2. Writing a simple binary search application

**Part 1 – Linear Search**

**Background**: Remember that the linear search will compare all elements of an array with a supplied “key” value for a positive match. On average, the algorithm will have to compare half the elements in the array before finding a match, if it exists. Think about how much time is increased for every element you add to an array. Is the linear search efficient?

First, you'll need to create a project.

1. Start Eclipse.
   1. Go to the *File* menu and select *New -> Project*
   2. Notice that you have a choice of languages.  Let's choose a Java project.
   3. Change the directory location to something "safe".  You may want to temporarily choose the Desktop and then save the file to something permanent once you're done (e.g. flash drive).
   4. Close the welcome screen tab.
   5. You should now see an open and blank project.
2. Create a new Java class, with the name Lab2ALinear.
3. Put your name as a comment on the top of the class.
4. Write the main method header, “public static void main(String[] args)”.
5. You will be generating random numbers so before your class and after your commented header, import java.util.Random.
6. You should consider developing the methods for this project incrementally (one at a time).
7. Write a method, called linearSearch, that returns the index of the key element if found in the array, otherwise it will return a -1.
8. In main, use an initializer list create an array of ints called *nums* holding the following values: 1, 4, 13, 43, -25, 17, 22, -37, 29. Test your linearSearch method on *nums* with a key value number in it (e.g. 4) and one that is not (e.g. 100).
9. Create an int array of size 20, called *data* and write a method to fill the array with random ints [-100, 100]. Recall that random’s nextInt( x) method returns a value from 0 to x-1. The upper bound is exclusive. Modify the values to fit the required ranges.
10. Use the linearSearch method you previously wrote to return the index of the key value searched for in *data* for the client to print. Let’s use 10 as the key value for every run. Notice that 10 may not be part of the data array every run.
11. Print the value of the key value searched for (10) and the index of the value, if found, or the phrase “Value not found in the array”, if not found.

**Part 2 – Binary Search**

**Background**: The binary search must be performed on an array that is sorted in order. The binary search starts by comparing the key to the value in the middle of the array. There are three cases:

* The key is less than the middle element, then you need only search the lower half of the array.
* The key is equal to the middle element, then your search is over.
* The key is greater than the middle element, then you need only search the upper half of the array.

First, you'll need to create a project.

1. Start Eclipse.
   1. Go to the *File* menu and select *New -> Project.*
   2. Notice that you have a choice of languages.  Let's choose a Java project.
   3. Change the directory location to something "safe".  You may want to temporarily choose the Desktop and then save the file to something permanent once you're done (e.g. flash drive).
   4. Close the welcome screen tab.
   5. You should now see an open and blank project.
2. Create a new Java class, with the name Lab2BBinary.
3. Put your name as a comment on the top of the class.
4. Write the main method header, “public static void main(String[] args)”.
5. You will be generating random numbers so before your class and after your commented header, import java.util.Random.
6. You should consider developing the methods for this project incrementally.
7. Write a method, called binarySearch, that returns the index of the key element if found in the array, otherwise it will return the value of minus(insertion point +1).
8. In main, use an initializer list create an array of ints called *nums* holding the following values: 1, 4, 13, 43, -25, 17, 22, -37, 29. Test your binarySearch method on *nums* with a key value number in it (e.g. 4) and one that is not (e.g. 100).
9. Ex. Given the array values {1, 4, 4, 22, -5, 10, 21, -47, 23}, the array will be sorted as -47, -5, 1, 4, 4, 10, 21, 22, 23. The index for the key value of 4 will be 3. The return value when searching for 6 will be the insertion point for 6 which is index of 5, so the return will be minus(5+1) = -6.
10. Create an int array of size 20, called *data* and write a method to fill the array with random ints [-100, 100]. Recall that random’s nextInt( x) method returns a value from 0 to x-1. The upper bound is exclusive. Modify the values to fit the required ranges.
11. Use the binarySearch method you previously wrote to return the index of the key value searched for in *data* for the client to print. Let’s use 10 as the key value for every run. Notice that 10 may not be part of the data array every run.
12. Print the value of the key value searched for (10) and the index of the value, which is the value of -(insertion point +1).

**What to Turn In**:

Follow submission guidelines on this page: <https://ccse.kennesaw.edu/fye/submissionguidelines.php>