CSC220 Lab05
Searching and Recursion

The goal of this week's lab is:

1. Practice searching
2. Continue learning the significance of special cases!
3. Learning how to write test to check your implementation

Things you must do:

1. There are many details in this lab. Make sure you read the whole thing carefully before writing any code and closely follow this instruction.
2. Always remember Java is case sensitive.
3. Your file names, class names, and package name must match exactly as they are specified here.

Things you must not do:

1. You must not change the file names, class names, package names.
2. You must not change the signature of any of these methods (name, parameters, …). Just fill in the missing code inside them. However, you are more than welcome to create any helper methods you need as necessary.

Part 0

- Create a new Java project and call it **Lab05.**
- Create a package inside your project and call it **lab05**.
- Download the Lab05-Assignment05 ZIP folder from Blackboard. **Copy and paste** the following files into your new lab05 package:

    - **BinarySearchSet.java.** This file will contain the representation of the class you are about to implement.
    - **Tester.java**. This file will contain the main function and tests for BinarySearchSet.

Part 1 - BinarySearchSet Class Description

For this lab (and assignment), you are asked to construct a class called BinarySearchSet that handles a list(s) of doubles. We have provided the skeleton of this class for you. This class requires:

- All of the usual operations for a list, such as the ability to add items, remove items, search for items, grow itself before running out of space.
- You must make your search routine(s) as efficient as possible. **You are not allowed to use sort algorithms**. Instead, you need to consider the fact that any list will begin as an empty list (which is already in sorted order), and you can simply insert items in the correct order to ensure that the list remains sorted **at all times**.
- Furthermore, the list must not contain any **duplicates**.
- The data in BinarySearchSet must be backed by a basic array (**do not use a Java List, ArrayList, or any other Java class**).

- It is **not acceptable** for the array to run out of space for new items, nor is it acceptable to create a gigantic array. Start with a modestly-size array, **let's say 6**, and increase the capacity as needed (see the grow() function description).
- Unlike previous assignments you are not given any tests as a starting point. You must create your own tests and submit them with your program. (Note that the previous assignment will be very useful in helping you accomplish this).

Part 2 – BinarySearchSet Class Implementation

We start by implementing the easier methods. Remember that the list must be sorted at all times. Please pay close attention to the following notes:

- `public BinarySearchSet() //default constructor`
    - This constructor will create a **storage** array of **size 6** (an initial size for an empty list)
    - and set the rest of the field members accordingly. **Pay attention to the member variables of the class! Each should be set to some (appropriate) initial value.**
- `public boolean isEmpty()`
    - returns **true** only if this list contains no elements
- `public int size()`
    - returns the number of elements in this list
- `public void grow()`
    - this function doubles the size of the storage array and modifies **member variables** accordingly.
    - consult this week's slides
- `public String toString()`
    - this method will print the elements of the list, its capacity, and its current size.

- This method is purely to help you test your implementation. We will NOT test your toString() method when grading.

- private int sequentialFind(double target)
  - this method will return the index where the element target occurs. **This method must make use of the Sequential search we learned in class.** If target is not present in the list, then it should return the index where it should be added (-index - 1). Does this formula make sense?

  - There are three cases to consider. What should be returned in each?
    - target is **equal** to storage[index]
    - target becomes **less than** storage[index]
    - the loop terminates without success

- public boolean insert(double newVal)

  - If the list does not contain newVal, add it to the correct position of the list and return **true**. If the list already includes the value, return **false**.

  - The method should first make sure there is enough room in the list to handle the operation. If not, what should you do?

  - It should then make a call to findIndex(), which then calls the private sequentialFind you just implemented, to see if newVal already occurs in the list (how do we know?). If it isn't in the list, the index returned by findIndex specifies the position where newVal is to be inserted (but it is negative! what should you do? :-)

  - Be careful about which **member variables** should be updated before returning true.

- public void clear()

  - Removes all the elements from the list. A call to isEmpty() should return **true** after this method is called.
  - Be careful about which **member variables** should be updated.

Part 3 – BinarySearchSet Class Implementation – Phase 2

THIS PART IS NOT OPTIONAL

Unlike previous labs/assignments you are not given any tests as a starting point. You must create your own tests to examine each method you implemented. These tests can be written inside **Tester.java**.

Here is a testing strategy: create an empty list. Then, check its size, check whether it is empty. Then, start adding a couple of numbers to your list and make sure your list adds them properly (that is, they are in the sorted order, the other member variables reflect the changes, etc). Then, continue adding values to your list till you go beyond its original capacity. Make sure you are not having any error in this case and that the list does indeed grow. Finally, test your clear to see if all elements are removed from the list.

Make sure to upload your code to Box when you have completed the lab.
Don't forget: lab is due tomorrow (Thursday) night @ 11:59pm!