Data Structures II - HW2

Instructions:

1. Download the hw2-files.zip file from the COL submission folder for HW2. As you did for HW1, unzip the file into the src folder that is in the workspace you created for this class.

2. Start up Eclipse. Use the same workspace that was created when you set up Eclipse. If you do not see hw2, right-click on the project or the src folder in the explorer window in Eclipse and select refresh.

3. Open the hw2 folder inside of Eclipse. You will find 6 files. The only 2 files you will be modifying are SequentialSearchST.java and RSequentialSearchSt.java.

4. Modify the 3 methods, put, get, and delete in SequentialSearchST.java so that the list is maintained in increasing order of the keys. When implementing put, get and delete, make sure to take advantage of the fact that the list is in order to avoid looking at all the items in the list. Also, whenever the list is modified, you will need to make sure it is still in sorted order after the modification. None of the functions may use recursion**. You may not use/call the keys() method**. **Note:** You are modifying the implementation of the methods, but not their interface or contract. To the external world, the methods should behave exactly as before. Make sure you code takes advantage of the sorted order. We will be checking for this and you will receive no credit for a method that blindly checks the entire list when it isn't necessary. The code you were given already does exactly that.

5. Use the HW2Test.java file to test the correctness of your SequentialSearchST.java code.

6. Run the JUnitTiming.java file. It will call put, get, and delete multiple times, checking for how long the calls take. If these tests fail, it is almost certainly because your code is not taking advantage of the sorted order of the keys to stop its search early.

7. Modify the 3 methods, put, get, and delete in RSequentialSearchST.java so that the list is maintained in increasing order of the keys. When implementing put, get and delete, make sure to take advantage of the fact that the list is in order to avoid looking at all the items in the list. Also, whenever the list is modified, you will need to make sure it is still in sorted order after the modification. **All of these functions must use recursion instead of loops.** This means each one will need a private helper method that takes a node (the front of the list) as an additional argument. Your RSequentialSearchST.java file cannot contain any loops except in the keys() method method which you are not changing. **You may not use/call the keys() method in your code.** **Note:** You are modifying the implementation of the methods, but not their interface or contract. To the external world, the methods should behave exactly as before. Make sure you code takes advantage of the sorted order. We will be checking for this and you will receive no credit for a method that blindly checks the entire list when it isn't necessary.

8. Use the RHW2Test.java file to test the correctness of your RSequentialSearchST.java code.

9. Run the RJUnitTiming.java file. It will call put, get, and delete multiple times, checking for how long the calls take. If these tests fail, it is almost certainly because your code is not taking advantage of the sorted order of the keys to stop its search early.

10. **IMPORTANT:** You may not change or remove the line that says "package hw2" nor may you change any public method headers.

11. This assignment will require 2 videos. Each video is limited to 3 minutes.

12. The first video is for SequentialSearchST. Answer the questions below, highlighting the relevant code in your solution:
    a. When inserting, your code must traverse the list, how does it know when to stop? (Show me your code and explain it)
    b. Suppose someone asked to delete a key and it happened to be the last key in the list. How does your code handle this case? Make sure to show me how the deletion works.
    c. Suppose someone asked to delete a key and it happened to be the first key in the list. How does your code handle this case? Make sure to show me how the deletion works.

13. The second video is for RSequentialSearchST. Answer the questions below, highlighting the relevant code in your solution:
    a. Explain all the base cases (all the non-recursive cases) in your recursive get helper method.
    b. Suppose someone asked to put a key that is not currently in the symbol table and belongs in the middle of the list. How does your code avoid searching the entire list?
    c. Suppose someone asked to delete a key that is smaller than all the keys in the symbol table (so it is not in the symbol table). What does your code do in this case? Make sure to show me the relevant code and explain it.

Sumission:

- Submit both **your SequentialSearchST.java** file and your **RSequentialSearchST.java** file.
- Make sure to provide two links in the comment box of the submission folder, one to each video you are required to make.
- Double check your submission. Download the files and make sure they are the ones you intend to submit! Do not skip this step. Every quarter I have students who have more than one version of the file on their computer and they submit the wrong one.

Grading:

The name of each function in the HW2Test and RHW2Test files ends in a number indicating how much that test is worth. The numbers in each test file add up to 100. Your base grade will be the sum of all the tests that pass divided by 2 to get a number between 0 and 100.

Note that the HW2Tests and RHW2Test files only test if your code gives the right answer. It doesn't indicate if you make use of a sorted list. You will not get credit if your code doesn't maintain the keys in sorted order or if it doesn't make use of the sorted order to terminate early when searching for a key. This is what the timing tests will help us to determine. Note that the code I gave you passes the vast majority of the HW2Test, and RHW2Test functions, but fails the Timing tests because it doesn't make use of the sorted list. You will also not get credit if you use any loops in RSequentialSearchST.java or if you call the keys() method in either file.

Like HW1, this is 70% of your grade for HW2. The other 30% will be based on your videos. Each video is worth 15%.