CS272 Lab Assignment #10.

Due: Thursday, 5/3 by 11:30pm.

Assignment: The Binary Search Tree and Sorting

1. Implement a class for a bag of int numbers which are stored as a binary search tree. Outline of Java source code for this class is provided at IntTreeBag.java. You must implement methods add, addAll, clone, countOccurrences, remove and union specified in the outline (IntTreeBag.java). Chapter 9.5 from the textbook discusses implementation of the methods. You have to read it to do the assignment. You need to use IntBTNode.java class which represents a node in a binary tree.

2. Binary search trees have their best performance when they are balanced, which means that at each node, n, the size of the left subtree of n is within one of the size of the right subtree of n. Write a recursive static method for IntTreeBag class that takes a sorted array of distinct integers and produces a balanced binary search tree. (The array is sorted with smallest integers at front to largest at the end.) The header of the method is the following:

   ```
   public static IntTreeBag sortedArrayToBST(int[] data, int first, int last)
   ```

   Hint: Set the middle element of the array to be the root of the tree, then recursively build the left subtree of the root, then the right subtree of the root.

   Note: For method `sortedArrayToBST` we assume that integers in the array are distinct (no duplicates). This is because if duplicates are allowed then a balanced BST containing elements from the array may not exist. For example, if array contains integers 5, 5, 5, 5, 5, 5, then a balanced BST containing 6 copies of 5 does not exist. If we use the proposed method to build a balanced BST from this array, we will get that 5 is a right child of 5 which contradicts the BST property that a right child of a node must contain a key that is strictly greater than the key of the node. For all the other methods in the assignments duplicates can be used.

   Note: IntTreeBag class does not support balanced property. If you add and/or remove nodes from a balanced binary search tree using methods from IntTreeBag class then the resulting tree may no longer be balanced.

3. Pick **one** of the sorting algorithms from Chapter 12. Sorting algorithms described in Chapter 12 are selection sort (code on page 618), insertion sort (code on page 670), merge sort (mergesort method on page 632 and the merge method on page 637), quicksort (quicksort method on page 642 and the "almost code" for the partition method on page 646), heapsort (the heapsort method on page 655, the makeHeap and parent pseudocode on pp 656 - 657 and the reheapifyDown method on pp 657-658).

4. Write a program that would do the following:

   1) prompt the user to enter the number of integers, then, prompt the user to enter integers (with no duplicates) and place them in an array in the order they are entered;
   2) use the sorting algorithm which you picked to sort elements of the array in increasing order and display the result (sorted array);
   3) use the method you wrote to produce a balanced binary search tree from the sorted array;
   4) display obtained balanced BST (tree 1) using print method of BTNode class;
   5) use clone method to create the second tree (tree 2) - clone of the first tree (tree 1); (to test your clone method)

6) allow the user to add and remove elements from tree 2 (multiple copies of elements may be added); display the tree after each addition or removal of an element (to test your add and remove methods);
7) prompt the user for an element and count occurrences of this element in tree 2 (to test your countOccurrences method);
8) use union method to build a union of tree 1 and tree 2 and display the resulting tree (to test union method);
9) use addAll method to add elements from tree 2 to tree 1 and display the result (to test addAll method).

Sample run of your program may look like the following (user inputs are in **green**):

```
How many elements are there? 5
Please enter 5 integers: 9 2 5 3 7
Sorted array: 2 3 5 7 9
BST (tree 1) obtained from the array:
5
  2
    --
    3
  7
    --
    9
Clone of the tree (tree 2):
5
  2
    --
    3
  7
    --
    9
Please enter 1 to add, 2 to remove, 0 to exit: 1
Please enter element you would like to add: 5
Updated tree 2:
5
  2
    --
    3
      --
      5
  7
    --
    9
Please enter 1 to add, 2 to remove, 0 to exit: 2
Please enter element you would like to remove: 2
Updated tree 2:
5
  3
    --
    5
  7
    --
    9
Please enter 1 to add, 2 to remove, 0 to exit: 0
Please enter element to count occurrences: 5
5 occurs 2 times in tree 2
```

```
Tree 3 (union of tree 1 and tree 2):
5
  2
    --
    3
      3
      5
        5
        --
  7
    7
    9
      9
      --
Result of adding all nodes from tree 2 to tree 1:
5
  2
    --
    3
      3
      5
        5
        --
  7
    7
    9
      9
      --
```

Note: You may assume that the user always gives correct input (e.g., when asked for integer the user enters integer). You do not need to check correctness of user input.

Specifications for all the methods which you implement should be included as comments in your code. Also, please use inline comments, meaningful variable names, indentation, formatting and whitespace throughout your program to improve its readability.

What to submit:

- Submit your source code electronically on Canvas.