CSCI 151
Exam 2 Solutions
May 6, 2022

This is a closed-book, closed-notes, closed-Internet exam.

The exam has 6 numbered questions; each is worth 16 points. You get 4 points for free, for a total of 100 points.

The last page of the exam is blank; you can use this for additional space for any of the exam questions.

After you have finished the exam please indicate whether you followed the Honor Code on the exam.

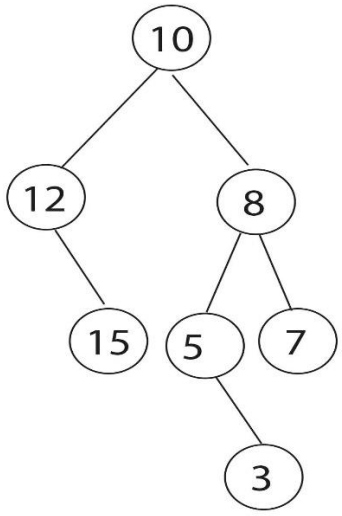
I ☐ did ☐ did not

adhere to the Honor Code while taking this exam.

_____

Signature

1. Here is a picture of a binary tree. Give **pre-order, in-order, and post-order traversals** of this tree.



Pre-order:

10 12 15 8 5 3 7

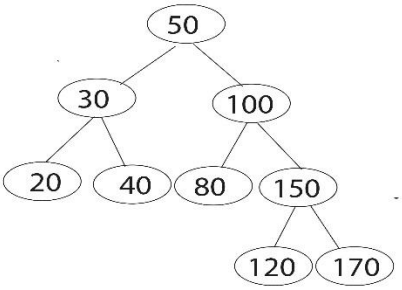
In-order:

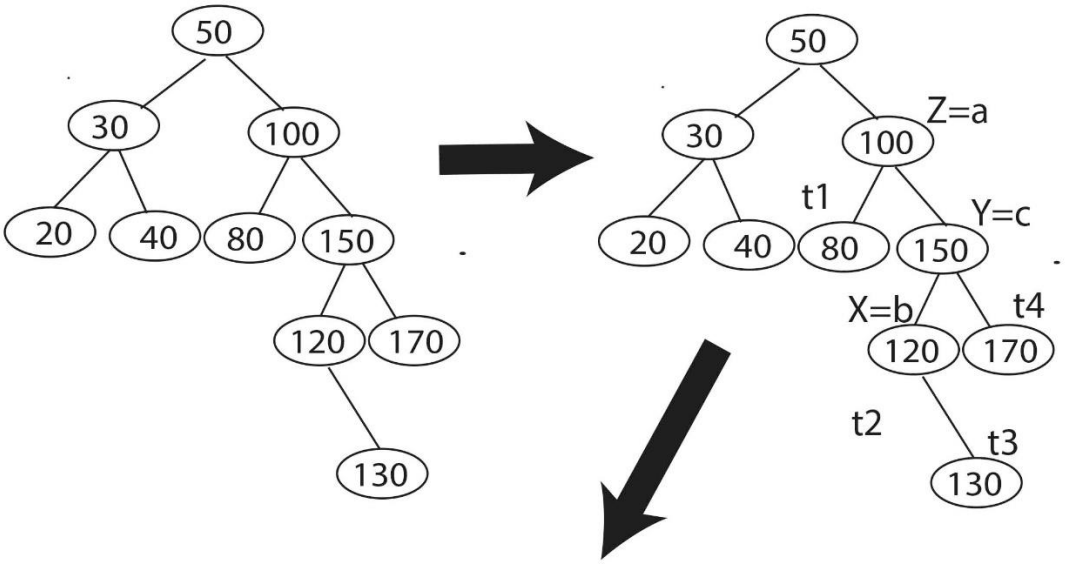
12 15 10 5 3 8 7

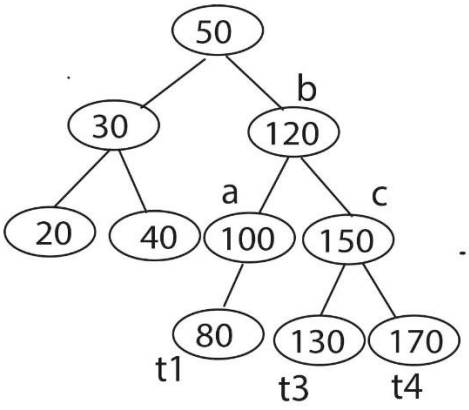
Post-order

15 12 3 5 7 8 10

2. Here is an AVL tree. **Draw the AVL tree that results from inserting 130 into this tree.**







3. **Write method**

   int indexLargest(LinkedList<Integer> L)

   that returns the index of the largest element of L. If L is empty return -1. **If L has N elements your method should run in time O(N).** Note that this asks for the index of the largest value, not for the largest value itself.

   I intended this as a question about iterators but hardly anyone took it that way. I ended up giving credit to any solution that returned the index of the largest element of L, whether you thought of this as a method of the LinkedList class or a method from some other class that uses LinkedLists.

```
int indexLargest(LinkedList<Integer> L) {
      if (L.size() == 0)
            return -1;
      else {
            Iterator<Integer> it = L.iterator( );
            int bigIndex;
            Integer value, bigValue;
            bigIndex = 0;
            bigValue = it.next();
            while (it.hasNext() ) {
                  value = it.next();
                  if (value > bigValue) {
                        bigValue = value;
                        bigIndex = it.nextIndex() -1;;
                  }
            }
            return bigIndex:
   }
}
```

4. Here is a list of key-value pairs and their hash codes:

(key-value)	(A 19)	(B 25)	(C 13)	(D 14)	(E 25)	(F 32)	(G 19)	(H 22)	(I 17)
hashcode	4	2	0	7	2	7	1	0	8

Add these values from left to right (first (A 19), then (B 25) …) to the following hash table. The small numbers are the table indices, so you don't have to count across to see where an index such as 6 occurs.

0	1	2	3	4	5	6	7	8
(C 13)	(G 19)	(B 25)	(E 25)	(A 19)	(H 22)	(I 17)	(D 14)	(F 32)

5. I have a very large file containing information about actors and movies they have acted in. The first few lines are

    Humphrey Bogart###Casablanca
    Judy Garland###The Wizard of Oz
    Daisy Ridley###Star Wars: The Force Awakens
    Humphrey Bogart###The Maltese Falcon
    Humphrey Bogart###Casablanca

    Let's assume you know how to get the actor/movie information from each line. The lines come in an arbitrary order; the movies for any particular actor are not listed together. There is also a problem with the data: because of the way the datafile was generated some lines appear more than once, and not together. The line: "Humphrey Bogart###Casablanca" might appear 5 times at difference places in the data file. You have been asked to write a program that reads this datafile and prints the names in order of the 50 actors who have acted in the most movies. Explain in English how you will do this. **Which of the standard Java data structures that we have discussed this semester (ArrayList, LinkedList, Stack, Queue, TreeMap, TreeSet, HashMap, HashSet, PriorityQueue) will you use and how will you use it?** You don't need to write any code for this question; just tell me in English how to put data structures together to solve this problem.

    Read the data file line by line into a string, and break the string into actor and movie fields.

    Build a HashMap<String, ArrayList<String>> where the keys are the actors' names and the value attached to an actor's name is a list of his or her movies. To solve the multiple lines problem check to see if a new movie is already in the actor's list of movies before adding it. A more efficient solution would use a HashSet<String> or a TreeSet<String> in place of the ArrayList<String>

    After reading the entire file build a Comparator<String> that compares two actors' names based on the length of their movie lists: compare(x, y) returns -1 if actor x has a longer movie list than actor y.

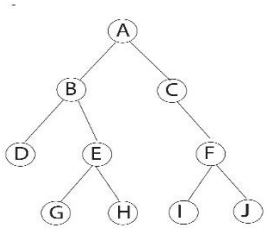
    Finally,. build a priority queue of actors' names based on this comparator and poll it 50 times.

6. Remember the tree structures from Lab 5 (where we wrote recursive methods for trees). BinaryTree<T> is an abstract class with two concrete subclasses: EmptyTree<T> and ConsTree<T>. EmptyTrees have no data or children; ConsTrees have a *data* field of type T, and children leftChild and rightChild, both of which are BinaryTrees.

Give the EmptyTree<T> and ConsTree<T> code for the method

int heightCount(int h)

which returns the number of nodes in the tree that are of height h, for any h>= 0. You can use all of the methods of Lab 5, including int height( ), without rewriting them.



For example, the following tree has one node (A) of height 3, two nodes (B and C) of height 2, two nodes (E and F) of height 1 and five nodes (D, G, H, I, and J) of height 0. Don't forget to write both the ConsTree method and the EmptyTree method.

```
Emptytree:    int heightCount(int h) {
                      return 0;
              }

ConsTree:     int heightCount(h) {
                   int myHeight =  this.height( );
                   if (myHeight < h)   // This condition actually isn't needed; the recursion
                        return 0;      // will handle it.
                   else if (myHeight == h)
                        return 1;
                   else
                        return leftChild.heightCount(h) + rightChild.heightCount(h);
              }
```

You can use this page for additional space for any question.