CSCE 3100 Assignment 3

Due: 11/03, 11:59PM

Question 1 [30pt]

A binary tree is called full if every non-leaf node has exactly two children.

(a)Emily has a 9-node full binary tree in which every node is labeled with a unique letter of the Roman alphabet or the character &. Preorder and postorder traversals of the tree visit the nodes in the following order:

- Preorder: M B P D R Q T H J
- Postorder: D R P Q B H J T M

Your task is to draw Emily's tree.

Hint: The preorder traversal implies the root must be M, and B must be its left child. What does the postorder list tell you about B's descendants?

(b) In what order does an inorder traversal of Emily's tree visit the nodes?

(c) It turns out any full binary tree can be uniquely identified by its preorder and postorder node sequences, but that may not be the case for arbitrary binary trees. Draw a pair of **distinct** binary trees that have identical preorder node sequences and identical postorder node sequences.

Advice: Your trees should contain very few nodes.

Question 2 [70pt]

In this assignment you will implement AVL trees. You are responsible for implementing insertion and deletion, so you are also responsible for rotating subtrees to maintain the balance property, find the minimum, and a few additional auxiliary methods.

In `assignment3.zip`, we are providing you with the following code:

```
avl.cpp -> your AVL implementation, mostly dummy text
avl.hpp -> header file, you do not have to change it
```

avl.cpp already has a couple methods implemented that you **should not change**:

- a method to print trees, and
- a main method that reads test cases (sequences of operations) and executes them

A sample README file is also provided. You may find how to compile and test the code.

We will test your implementation with test cases that spell out operations (insert a number, delete a number, and print) to be executed on an (initially) empty AVL tree. For example,

```
insert 100
insert 30
insert 20
insert 50
print
delete 30
print
insert 990
insert 900
print
```

means that you should insert 100, 30, 20, 50 and print the resulting AVL tree. Then you should delete 30 and print the resulting AVL tree. Finally, you need to insert 990 and 900 and print the resulting AVL tree. The expected outcome is the following:

```
30
|l 20
|r 100
| |l 50

50
|l 20
|r 100

50
|l 20
|r 900
| |l 100
| |r 990
```

where the last tree has root 50 and two children: 20 and 900, and node 900 has two children: 100 and 990. Also, 50 is the left child of 100 in the first tree.

A few more notes:
• Your insertion and deletion must run in O(log(n)), and you must update the height of nodes after each operation.

Instructions

Submit the coding report with screenshot within the pdf together with question1, submit your code on Canvas. Include a brief README file explaining your code.