CSE331 – Project 3
Binary Search Trees

Due Date: 11:59 pm Oct.27, 2017

1. Project Description

In this project you will complete an implementation of the binary search tree data structure and use it to store and manipulate data from the given txt file. You have been given code to implement a program that will load and manipulate the data from the numbers.txt file. The main program stores the numbers in a vector, and then builds a binary search tree based on the vector.

The program does not need any input argument.

The code to load the database is already written for you.

Your job is to complete the Tree.h file which has been provided. You must provide an efficient implementation of a binary search tree. Most of your functions should have a best case running time of O(log N), however depth and preorder will always take O(N), since they must recurse over every node.

You may assume there are no duplicates in those numbers.

You may not use anything from the STL for this project. You may use IO for debugging, but be sure to remove it from your completed project. You will be marked down for any debugging IO left in your project. As always you may not change the signatures of the public methods, the node class, or the way in which the tree class interacts with the rest of the program. You may add any private member variables or methods to CTree to help you complete the project. You may also assume that any templated types passed to CTree will have overloaded output stream operators (aka "<<").

2. Programming Notes

A sample database file is (numbers.txt) provided in the project directory. This is a subset of the full dataset. The sample file is provided to help you test primary functionality. You should perform much more rigorous testing on your completed project and make sure to test them on Black! When we grade your project 3 we will be using the full dataset and a much longer command file.

An even smaller database file (numbers_smaller.txt) and its corresponding result is also provided. Make sure your result matches to our given result before you test your code on that larger database.

In the directory we have placed a windows program that you can use to explore the behavior of trees, both balanced and unbalanced. The executable is called "VisualTree.exe".

We recommend that you implement all of your functions in the private space, and then place calls to private methods in the public methods. This will reduce the amount of code you write when overloading the const methods.

3. Programming Details

- There is no duplicates in numbers.txt (or numbers_small.txt).
- For this project, you don't need to balance your binary search tree. (In fact, don't balance your tree because this would generate inconsistent result.)
- Follow what is taught on the slide/ in the text book. For example, the number at the left child node should be smaller than the number at the parent node.

- Make sure your code works with the main.cpp provided.
- Use template based methods.
- You probably need to declare additional functions to finish Delete().
- Depth of a tree. By definition, a tree with only one node would have a depth value of 0. As we are using recursion to find the depth, treating a tree like this as depth of 1 might be more convenient. Thus we accept both versions of depth implementation. The sample result is based on the latter one. If your depth output gives 6 and 5 with number_small.txt, you answer would still be correct.
- Make sure you format your output as is given in result_small.txt. The preorder format should be an output number followed by " " (four spaces). Make sure you remove all debug output in your code.

4. Project Deliverables

The following files must be submitted via Handin no later than 11:59 pm Friday October 27, 2017:

- Tree.h – contains your implementation of binary search tree.
- project4.pdf – files containing your answers to the written questions.

5. Written Questions

1. Suppose we have the following set of numbers {10, 11, 15, 19, 23, 78, 42, 56, 18, 13, 12, 38, 47}
    a. Determine the order of insertions with this set of numbers that will result in a perfectly balanced BST.
    b. Show the results of a preorder traversal of this tree.
    c. Delete the root, make a diagram of the resulting tree.