2100.021 - Program 3 - Binary Search Tree

My Solutions

Implement a binary search tree for positive integers using C++ classes. Your program must be able to add nodes, remove nodes, search for nodes, list nodes in in-order, and display the height of the tree. You must implement the tree yourself and are not allowed to use the C++ standard template library, i.e. you will need to implement a class for the tree and a class for the nodes (header and class file each).

Upon starting your program, it will display the prompt bst> followed by a space:

```
bst>
```

The user can then select between the following commands: add, delete, search, inorder, height, and quit.

add

Add takes the value of the node to add as an argument. Your program will create a node and insert it into thecorresponding place. Then it will repeat the prompt.

```
bst> add 5
bst> add 4
bst>
```

delete

Delete takes the value of the node to delete as an argument. Your program will search for the node and delete it. As usual, leaf nodes are simply deleted For this assignment, always replace an interior node by the largest node of the left sub-tree. **Note that this causes a recursive call to delete the largest node of the left sub-tree.** Only if there is no left sub-tree, then replace the interior node by the smallest node of the right sub-tree. Then repeat the prompt.

```
bst> delete 4
bst> delete 3
bst>
```

search

Search takes the value of the node to search, and returns true if the node exists in the tree, false if the node does not exist. Then it will repeat the prompt.

```
bst> search 5
true
bst> search 2
false
```

inorder

This command does not take any arguments. It prints the result of an inorder traversal. The individual numbers are separated by hyphens. Then it will repeat the prompt.

```
bst> inorder
2-5-6
bst>
```

height

This command does not take any arguments. It prints the height of the tree. Then it will repeat the prompt.

```
bst> height
2
bst>
```

quit

This command does not take any arguments. It terminates the program.

```
bst> quit
```

You must make your program "fool-proof", i.e. implement error checking. Each error output must start with "Error!" followed by an appropriate message.

Submit at least the following files:

- Your main class file (e.g Main.cpp) controling the flow of the program
- The prototype for your node class
- The implementation of your node class
- The prototype for your tree class
- The implementation of your tree class

SAMPLE PROGRAM EXECUTION:

```
bst> add 2
bst> add 8
bst> add 3
bst> add 10
bst> search 2
true
bst> search 5
false
bst> inorder
2-3-8-10
bst> height
```

```
2
bst> quit
```

Grading

Your program will be judged on the following:

- 45% - Passes I/O requirements
- 40% - Code satisfies requirements of assignment
- 15% - Professional coding style
  - 5% Adequate comments
  - 5% Modularity (small main function, separate functions, etc)
  - 5% Readability (line length, indentation, variable names)

As with all other programming assignments in this class, your program will receive a 0 if it does not compile. If it does not compile on Praktomat it MUST compile on cse01.cse.unt.edu.