Evan Nguyen

Joseph Guzman

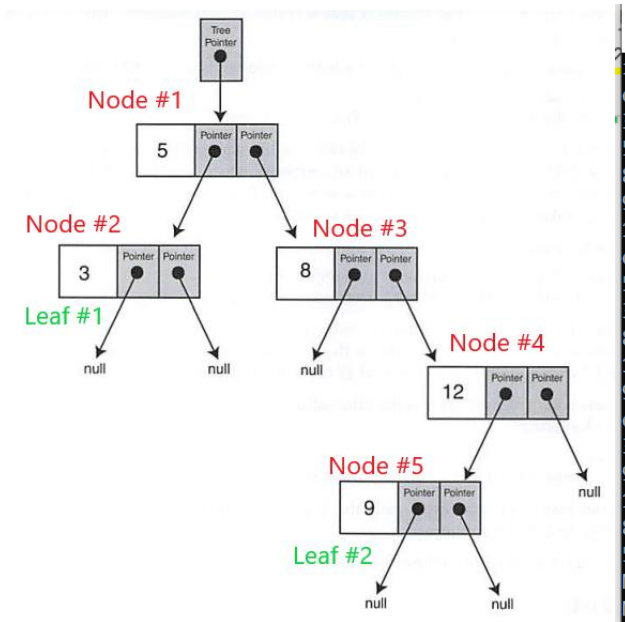
CECS 275

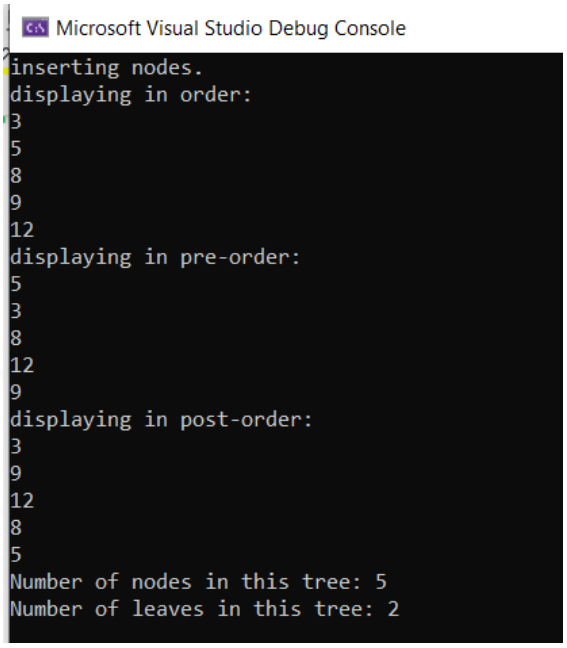
Spring 2022

Lab 7

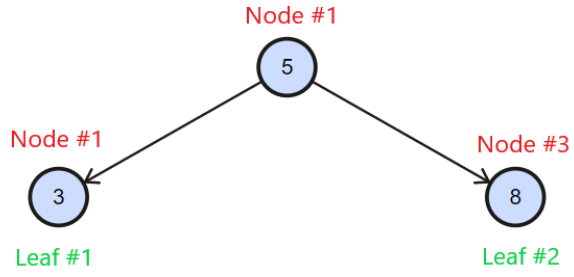
Outputs are shown first, then code screenshots.

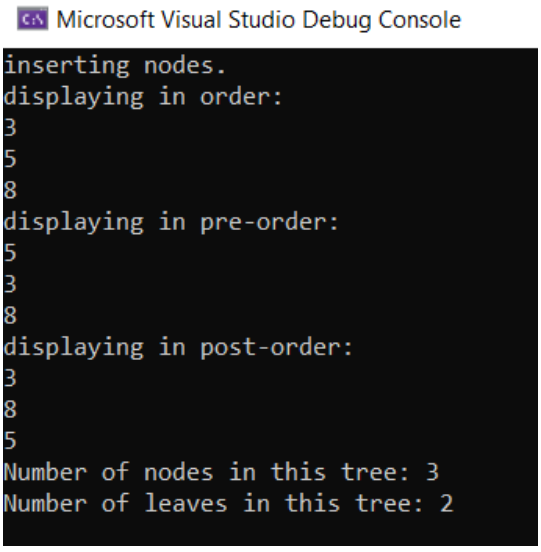
Example 1: Given the following binary tree from the book:



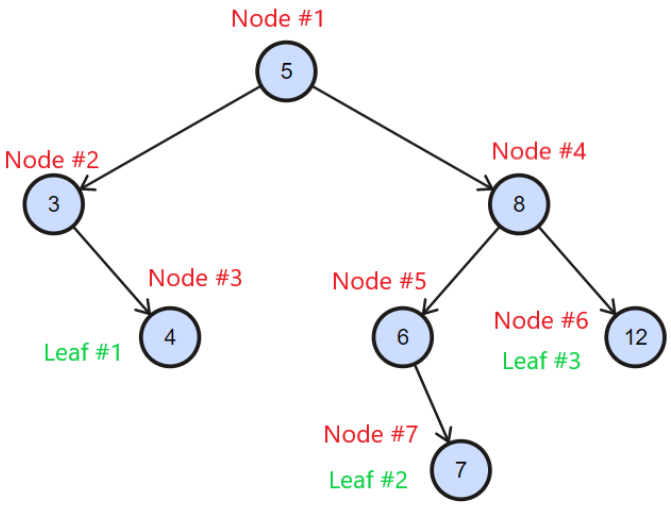


Example 2: Given this binary tree:





Example 3: Given this binary tree:



Microsoft Visual Studio Debug Console

```
inserting nodes.
displaying in order:
3
4
5
6
7
8
12
displaying in pre-order:
5
3
4
8
6
7
12
displaying in post-order:
4
3
7
6
12
8
5
Number of nodes in this tree: 7
Number of leaves in this tree: 3
```

Code:

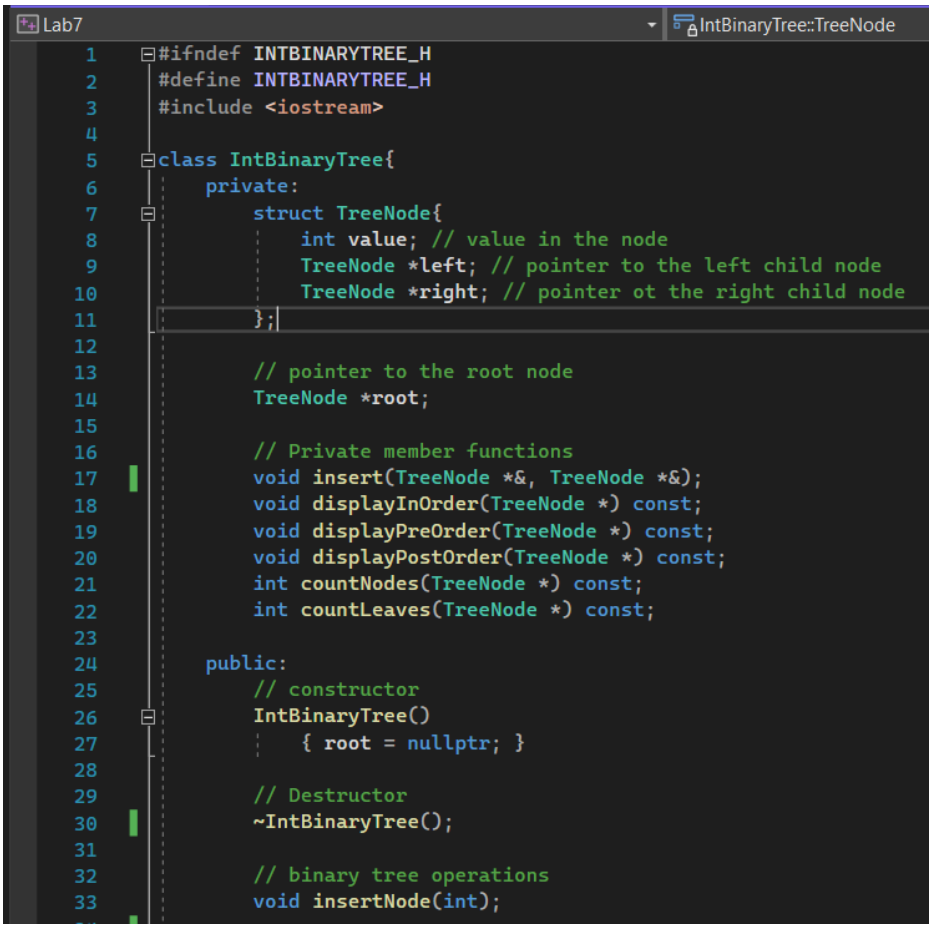
Main.cpp

main.cpp

C: > Users > nguye > AppData > Local > Temp > Temp1_code.zip > main.cpp

```cpp
/*
 * Answer to Lab 7
 * CECS 275 - Spring 2022
 * @author Evan Nguyen
 * @author Joseph Guzman
 * @version 1.0.0
 *
 */

#include <iostream>
#include "IntBinaryTree.h"
using namespace std;

int main() {

    IntBinaryTree tree;
    cout << "inserting nodes. \n";
    tree.insertNode(5);
    tree.insertNode(3);
    tree.insertNode(8);
    tree.insertNode(4);
    tree.insertNode(6);
    tree.insertNode(7);
    tree.insertNode(12);

    cout << "displaying in order: \n";
    tree.displayInOrder();
    cout << "displaying in pre-order: \n";
    tree.displayPreOrder();
    cout << "displaying in post-order: \n";
    tree.displayPostOrder();
    cout << "Number of nodes in this tree: ";
    tree.displayCountNodes();
    cout << endl;
    cout << "Number of leaves in this tree: ";
    tree.displayCountLeaves();
    cout << endl;
    return 0;
}
```

IntBinaryTree.h



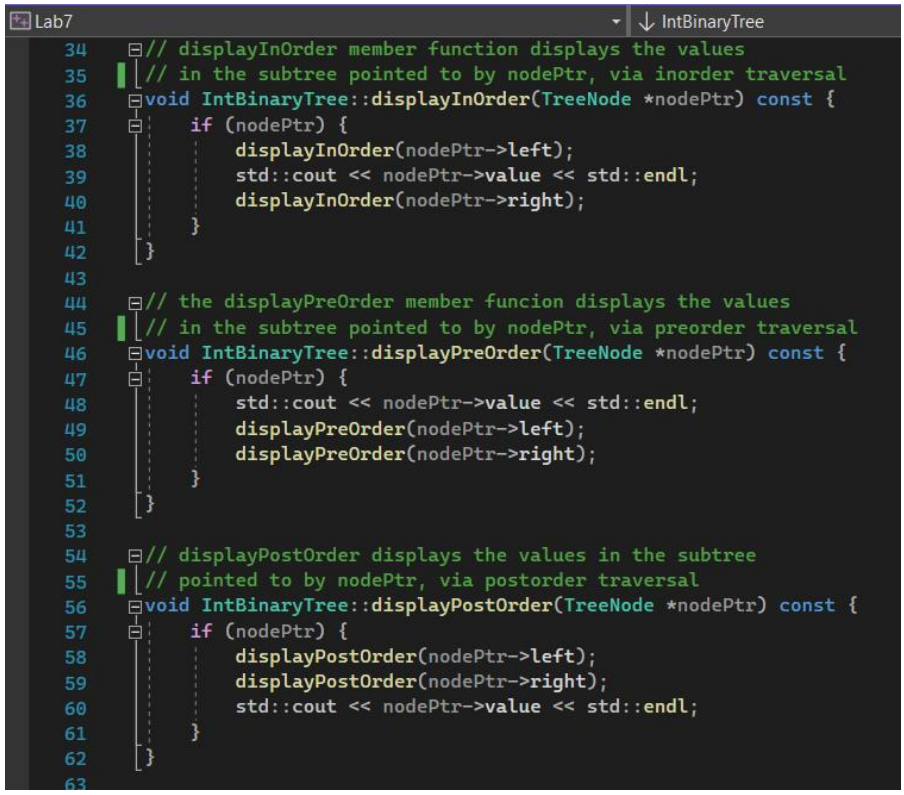

```h
#ifndef INTBINARYTREE_H
#define INTBINARYTREE_H
#include <iostream>

class IntBinaryTree{
    private:
        struct TreeNode{
            int value; // value in the node
            TreeNode *left; // pointer to the left child node
            TreeNode *right; // pointer ot the right child node
        };

        // pointer to the root node
        TreeNode *root;

        // Private member functions
        void insert(TreeNode *&, TreeNode *&);
        void displayInOrder(TreeNode *) const;
        void displayPreOrder(TreeNode *) const;
        void displayPostOrder(TreeNode *) const;
        int countNodes(TreeNode *) const;
        int countLeaves(TreeNode *) const;

    public:
        // constructor
        IntBinaryTree()
            { root = nullptr; }

        // Destructor
        ~IntBinaryTree();

        // binary tree operations
        void insertNode(int);
```

```
32          // binary tree operations
33          void insertNode(int);
34
35          void displayInOrder() const
36              { displayInOrder(root); }
37
38          void displayPreOrder() const
39              { displayPreOrder(root); }
40
41          void displayPostOrder() const
42              { displayPostOrder(root); }
43
44          void displayCountNodes() const
45              { std::cout << countNodes(root); }
46
47          void displayCountLeaves() const
48              { std::cout << countLeaves(root); }
49      };
50
51      #endif
```

IntBinaryTree.cpp

Lab7 | IntBinaryTree

```
1   #include <iostream>
2   #include "IntBinaryTree.h"
3
4   // Insert accepts a TreeNode pointer and a pointer to a nodePtr
5   // The function inserts the node into hte tree pointed to
6   // by the TreeNode pointer. This function is calle recursively
7   void IntBinaryTree::insert(TreeNode *&nodePtr, TreeNode *&newNode){
8
9       if (nodePtr == nullptr) {
10          nodePtr = newNode; // insert the node
11      } else if (newNode->value < nodePtr->value) {
12          insert(nodePtr->left, newNode); // search the left branch
13      } else {
14          insert(nodePtr->right, newNode); // search the right branch
15      }
16  }
17
18  // insertNode creates a new  node to hold num as its value
19  // and passes it to the insert function
20  void IntBinaryTree::insertNode(int num){
21
22      // pointer to a new node
23      TreeNode *newNode = nullptr;
24
25      // create a new node and store num in it
26      newNode = new TreeNode;
27      newNode->value = num;
28      newNode->left = newNode->right = nullptr;
29
30      // insert the node
31      insert(root, newNode);
32  }
```




```
34  // displayInOrder member function displays the values
35  // in the subtree pointed to by nodePtr, via inorder traversal
36  void IntBinaryTree::displayInOrder(TreeNode *nodePtr) const {
37      if (nodePtr) {
38          displayInOrder(nodePtr->left);
39          std::cout << nodePtr->value << std::endl;
40          displayInOrder(nodePtr->right);
41      }
42  }
43
44  // the displayPreOrder member funcion displays the values
45  // in the subtree pointed to by nodePtr, via preorder traversal
46  void IntBinaryTree::displayPreOrder(TreeNode *nodePtr) const {
47      if (nodePtr) {
48          std::cout << nodePtr->value << std::endl;
49          displayPreOrder(nodePtr->left);
50          displayPreOrder(nodePtr->right);
51      }
52  }
53
54  // displayPostOrder displays the values in the subtree
55  // pointed to by nodePtr, via postorder traversal
56  void IntBinaryTree::displayPostOrder(TreeNode *nodePtr) const {
57      if (nodePtr) {
58          displayPostOrder(nodePtr->left);
59          displayPostOrder(nodePtr->right);
60          std::cout << nodePtr->value << std::endl;
61      }
62  }
63
```

IntBinaryTree.h | main.cpp | IntBinaryTree.cpp

Lab7 | IntBinaryTree

```cpp
// countNodes counts the nodes in a tree
int IntBinaryTree::countNodes(TreeNode *nodePtr) const {
    int count = 0; // count root node
    if (nodePtr == NULL){
        return 0;
    } else {
        count += countNodes(nodePtr->left);
        count += countNodes(nodePtr->right);
    }
    return count++ + 1;
}

// countLeaves counts the number of leaves in the tree
int IntBinaryTree::countLeaves(TreeNode *nodePtr) const {
    int count;
    if ( nodePtr == NULL ){
        return 0;
    }
    // if there is no left or right node, return 1
    if (nodePtr->left == NULL && nodePtr->right == NULL) {
        return 1;
    }
    else {
        count = countLeaves(nodePtr->left) + countLeaves(nodePtr->right);
        return count;
    }
}

IntBinaryTree::~IntBinaryTree() {

}
```