CS 341
MP #4
Binary Search Trees

Due May 8

For this machine problem you are to writer recursive functions to perform basic operations on binary search tree. You are given code for which you must fill in the missing parts. Under NO circumstances are you to use any loops! The function `void  treeInsert( int, treeNode * *);` inserts new values into a binary search tree. The tree is built by starting with an empty tree and inserting nodes. Notice that this tree allows for multiple occurrences of an item by maintaining a counter field. The first time a value is inserted in the tree a new node is required. Future insertions of a value already in the tree should result in incriminating the counter field. The function `void printTree(treeNode *);` should print out the values stored in the tree in sorted order. This should be accomplished by an in order traversal. Each value should be output as many times as it was input (that as many times as indicated by the counter field). The functions `int minTree(treeNode *);` and `int maxTree(treeNode *);` should return the minimum and maximum values in the tree respectively. The input is terminated by 0.

Use the following test data:

```
1, 2, -3, 4, 5, -3, 6, 2, 5, -4, 3, 4, 2, 1, -1, 0

1, 2, 3, 4, 5, 6, 7, 8, 9, 0

9, 8, 7, 6, 5, 4, 3, 2, 1, 0

2, 2, 2, 2, 2, 2, 2, 0

1, 3, 5, 7, 9, 7, 5, 3, 1, 0
```

```
#include<iostream>
using namespace std;

class treeNode {

private:
int _value;
int _count;
treeNode * leftChild;
treeNode * rightChild;

public:
treeNode();
treeNode( int );
treeNode * * Right(){ return & rightChild; }
treeNode * * Left() { return & leftChild; }
void setLeft(treeNode *);
void setRight(treeNode *);
int Value() {return _value; }
int Count() {return _count; }
void Increment() { _count++; }
};

void treeNode::setLeft(treeNode * tPrt) {
     leftChild = tPrt;
}

void treeNode::setRight(treeNode * tPrt) {
     rightChild = tPrt;
}


treeNode::treeNode( ){
     _value = 0;
     _count = 1;
     leftChild = 0;
     rightChild = 0;
     }
```

```
treeNode::treeNode( int value ){
     _value = value;
     _count = 1;
     leftChild = 0;
     rightChild = 0;
     }


// End of treeNode class

void  treeInsert( int, treeNode * *);

void printTree(treeNode *);

int minTree(treeNode *);

int maxTree(treeNode *);

int main(){
treeNode * treeRoot = 0;
int inPutValue = 0;

cout << " -> " << flush;
cin >> inPutValue;

while ( inPutValue ) {

     treeInsert( inPutValue, & treeRoot );

     cout << " -> " << flush;
     cin >> inPutValue;

     }

     printTree(treeRoot);

cout << "\n\nminimun value is: " << flush;
cout << minTree( treeRoot ) << endl;

cout << "\n\nmaxmum value is: " << flush;
cout << maxTree( treeRoot ) << endl;


}

void treeInsert( int ivalue, treeNode * * tNode ){
/*
Supply missing code.
*/
}
/*
//You may use a repeatPrint function as a
//helper funcion for printTree.
void repeatPrint(int value, int times) {
}
*/
void printTree(treeNode * tRoot){
/*
Supply missing code.
*/
}


int minTree(treeNode * tRoot ) {
/*
Supply missing code.
*/
}

int maxTree(treeNode * tRoot ) {
/*
Supply missing code.
*/
}
```