Jacob Ciurca

Dr. Lu

Lab 5

Cosc 320

10/8/2020

**Pre Lab:** Reviewed binary trees as well as implemented int.h

**Int.h:**

#ifndef Integer\_H

#define Integer\_H

#include <iostream>

using namespace std;

class integer

{

public:

// constructor. initialize intValue and set count = 1

integer(int n){

intValue = n;

count = 1;

};

// return intValue

int getInt(){

return intValue;

};

// return count

int getCount(){

return count;

};

// increment count

void incCount(){

count++;

};

// compare integer objects by intValue

friend bool operator< (const integer& lhs, const integer& rhs);

friend bool operator== (const integer& lhs, const integer& rhs);

// output object in format intValue (count)

friend ostream& operator<< (ostream& ostr, const integer& obj);

private:

// the integer and its count

int intValue;

int count;

};

bool operator< (const integer& lhs, const integer& rhs){

if(lhs.intValue < rhs.intValue){

return true;

}

else{

return false;

}

}

bool operator== (const integer& lhs, const integer& rhs){

if(lhs.intValue == rhs.intValue){

return true;

}

else{

return false;

}

}

ostream& operator<< (ostream& ostr, const integer& obj){

ostr << obj.intValue << " (" << obj.count << ")";

return ostr;

}

#endif

**Lab:**

**Lab04.cpp:**

#include <iostream>

#include <random>

#include <utility>

#include "int.h"

#include "d\_stree.h"

#include "d\_stiter.h"

#include "d\_random.h"

#include "d\_except.h"

using namespace std;

int main()

{

integer i0(0);

integer i1(1);

integer i2(2);

integer i3(3);

integer i4(4);

integer i5(5);

integer i6(6);

stree<integer> tree;

stree<integer>::iterator iter;

for (int i=0; i<10000; i++){

int f = rand()%7;

if(f == 0){

iter = tree.find(i0);

if(\*iter == i0){

i0.incCount();

tree.erase(i0);

tree.insert(i0);

}

else{

tree.insert(i0);

}

}

if(f == 1){

iter = tree.find(i1);

if(\*iter == i1){

i1.incCount();

tree.erase(i1);

tree.insert(i1);

}

else{

tree.insert(i1);

}

}

if(f == 2){

iter = tree.find(i2);

if(\*iter == i2){

i2.incCount();

tree.erase(i2);

tree.insert(i2);

}

else{

tree.insert(i2);

}

}

if(f == 3){

iter = tree.find(i3);

if(\*iter == i3){

i3.incCount();

tree.erase(i3);

tree.insert(i3);

}

else{

tree.insert(i3);

}

}

if(f == 4){

iter = tree.find(i4);

if(\*iter == i4){

i4.incCount();

tree.erase(i4);

tree.insert(i4);

}

else{

tree.insert(i4);

}

}

if(f == 5){

iter = tree.find(i5);

if(\*iter == i5){

i5.incCount();

tree.erase(i5);

tree.insert(i5);

}

else{

tree.insert(i5);

}

}

if(f == 6){

iter = tree.find(i6);

if(\*iter == i6){

i6.incCount();

tree.erase(i6);

tree.insert(i6);

}

else{

tree.insert(i6);

}

}

}

cout << "Display tree\n";

for(iter = tree.begin(); iter != tree.end(); iter++){

cout << \*iter << endl;

}

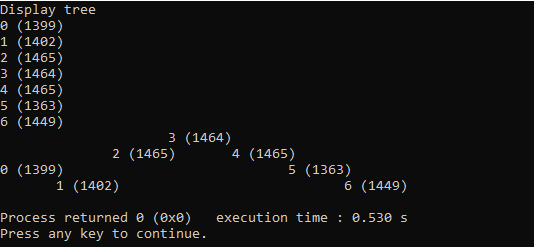
cout << tree.size() << endl;

tree.displayTree(6);

return 0;

}

**Sample Output:**



**Post Lab:** My sample output is very similar to the output given to us as a reference. This lab took me about 2 hours to complete and I completed this lab by myself with no help.