Ryan Ellis
COSC 320 --- 001 Lab 5 3/9/2025

Lab report: In this lab, we were given the header files for the d_stree and integer class structure and were asked to implement a main function that would build a binary search tree with the integer class objects and output the findings of inserting and counting how many integer objects had the same occurrence. This lab took me about 2 hours to complete cumulatively.

Pre-lab:

1. Review what we learned about binary search tree.
2. Finish the implementation of the class integer in “int.h”

Lab 3.1

Exercise 1:

Header file (int.h)

```
//============================================================
// Author: Ryan Ellis
// Creation Date: 3/4/2025
// Last Update: 3/9/2025
// Description: Header file for integer class and implementation for class functions.
// Notes:
//============================================================

#ifndef Integer_H
#define Integer_H
#include <iostream>
#include <iomanip>

using namespace std;

class integer {
public:
  // constructor. initialize intValue and set count = 1
  integer(int n);

  // return intValue
  int getInt();

  // return count
  int getCount();

  // increment count
  void incCount();

  // 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;
};

//=====================================================
// Description: Constructor for integeger class, sets default
// values of count to 1, and intValue to passed parameter.
//=====================================================

integer :: integer(int n){
    count = 1;
    intValue = n;
}

//=====================================================
// Description: Function to return intValue
//=====================================================

int integer :: getInt(){
    return intValue;
}

//=====================================================
// Description:Function to return count
//=====================================================

int integer :: getCount(){
    return count;

}

//=====================================================
// Description: Function to increment private count variable
//=====================================================

void integer :: incCount(){
    count += 1;
}

//=====================================================
// Description: Overloaded comparison (less than) operator
// for the integer class, will return boolean value true/false
```

```
//=================================================

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

   if(lhs.intValue < rhs.intValue)
      status = true;
   else
      status = false;
   return status;

}
//=================================================
// Description: Overloaded equivalency operator
// for the integer class, will return boolean value true/false
//=================================================

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

   if(lhs.intValue == rhs.intValue)
      status = true;
   else
      status = false;
   return status;
}

//=================================================
// Description: Overloaded streaming operator
// for the integer class, will return formatted output
// for private data members of integer class.
//=================================================

ostream &operator<<(ostream &ostr, const integer &obj){
   ostr << obj.intValue<<" "<<"("<<obj.count<<")";
   return ostr;
}

#endif
```

Implementation File (main program)

```
//=============================================================
// Filename: lab_05.cpp
// Author: Ryan Ellis
// Creation Date: 3/4/2025
// Last Update: 3/9/2025
// Description: Main function to test d_stree class and integer class
```

```
// which creates a binary search tree of integer class data types.
// Notes:
//=============================================================
#include "d_stree.h"
#include "int.h"
#include <ctime>
#include <iostream>

using namespace std;

void div();     //prototypes
void printInOrder(stree<integer> &tree);

const int LIMIT = 10000;   // global constant int to set limit for loop

int main() {

  srand(time(0));   //set random seed

  stree<integer> tree;      //declare stree object

  int counter = 0;          //loop counter

  while(counter < LIMIT){     //while loop for 10,000 insertions

     integer num(rand()%+7);     //create integer object with random number 0-6
     stree<integer> ::iterator flag = tree.find(num);  //tree integer iterator to parse tree and find num

     if(flag != tree.end())
        (*flag).incCount();   //if iterator is found then increment the count
     else
        tree.insert(num);   //otherwise insert into tree

     counter++;    //increment loop counter
  }

  cout<<"Values in the tree: "<<endl;   //output findings
  printInOrder(tree);
  div();
  cout<<"Binary Tree "<<endl;
  tree.displayTree(2);
  div();


  return 0;
}
void div() { cout << "\n============================" << endl; }

void printInOrder(stree<integer> &tree){    //function to print tree leaf nodes in order
```

```
  stree<integer>::iterator titer;
  for(titer = tree.begin(); titer != tree.end(); ++titer){
    cout<<*titer<<endl;
  }
}
```

Output: