Homework 4: Trees (due Monday, August 17th at 11:59 PM)

Here is a C++ class definition for an abstract data type `WordTree` of `string` objects. You must store the words and the counts of the words in a single binary search tree. Each word occurring in the text can only be stored once in the tree. Implement each member function in the class below. The `WordTree` class may have only one member variable, root, and it must be private. You may add additional private members functions to the `WordTree` class. Some of the functions we may have already done in lecture, that's fine, try to do those first without looking at your notes. Remember to provide an appropriate copy constructor, destructor and assignment operator for the `WordTree` class as well.

```
#include <iostream>
#include <string>

typedef std::string WordType;

struct WordNode {
   WordType m_data;
   WordNode *m_left;
   WordNode *m_right;
   // You may add additional data members and member functions
   // in WordNode
};

class WordTree {
   private:
      WordNode *root;
   public:
      // default constructor
      WordTree() : root(nullptr) { };

      // copy constructor
      WordTree(const WordTree& rhs);

      // assignment operator
      const WordTree& operator=(const WordTree& rhs);

      // Inserts v into the WordTree
      void add(WordType v);

      // Returns the number of distinct words / nodes
      int distinctWords() const;
      // Returns the total number of words inserted, including
```

```
      // duplicate values
      int totalWords() const;

      // Prints the LinkedList
      friend ostream& operator<<(std::ostream &out, const
WordTree& rhs);

      // Destroys all the dynamically allocated memory in the
      // tree
      ~WordTree();
};
```

The `add` function enables a client to insert elements into the `WordTree`. If an element has already been added, repeated calls to `add` will not add a new `WordNode`. Instead the count of the occurrences of that word will increase.

```
WordTree k;

k.add("Kim");
k.add("Kanye");
k.add("Kanye");
k.add("Kanye");

assert(k.distinctWords() == 2);
assert(k.totalWords() == 4);
```

The output operator << enables a client to print elements of a WordTree. The key and the number of occurrences of the key are printed. The output should be sorted according to the key.

```
WordTree w;

w.add("Harry");
w.add("Niall");
w.add("Niall");
w.add("Liam");
w.add("Louis");
w.add("Harry");
w.add("Niall");
w.add("Zayn");

cout << w;
```

must write (including the order)

```
Harry 2
Liam 1
Louis 1
Niall 3
Zayn 1
```

For the output operator in a class, you must overload it. You must also implement it as a non-member friend function.

When comparing items, just use the == or != operators provided for the string type by the library. These do case-sensitive comparisons. In other words, the The THE will count as different words for this assignment, that's fine.

Turn It In

By Sunday, August 16, there will be a link on CCLE that will enable you to turn in this homework. Turn in one zip file that contains your solutions to the homework problem. The zip file must contain only the files `WordTree.h`, `WordTree.cpp`, and `main.cpp`. The header file `WordTree.h` will contain all the code from the top of this specification (`include`s, `typedef`, `struct WordNode`, `class WordTree`) and proper guards, while the C++ file `WordTree.cpp` will contain the `WordTree` member functions you will write. If you don't finish everything you should return dummy values for your missing definitions. The main file `main.cpp` can have the main routine do whatever you want because we will rename it to something harmless, never call it, and append our own main routine to your file. Our main routine will thoroughly test your functions. You'll probably want your main routine to do the same. Your code must be such that if we insert it into a suitable test framework with a main routine and appropriate #include directives, it compiles. (In other words, it must have no missing semicolons, unbalanced parentheses, undeclared variables, etc.). The main routine, if you have one, must at least have a `return 0;` within the body.