Assignment #6 (45 points; due 12/18/2017 @ 11:59 P.M.)

In this assignment, you are to write a program that analyzes a selection of text, counting the number of times each word appears in the text. Your word counts must ignore capitalization, so the, The, THE, and tHe all increase the count for the word "the" by one. For purposes of this assignment, a word is any consecutive string of letters and the apostrophe character, so don't counts as a single word, and best-selling counts as two words: best and selling. Notice that a blank space will not necessarily occur between two words.

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. Call the structure for the nodes of the tree WordNode, and call the pointers in this structure left and right. Use C++ strings to store words in the tree. Call the class implementing the binary search tree WordTree.

Here is the framework for the WordNode tree struct and WordTree class:

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

typedef string ItemType;

struct WordNode
{
     ItemType 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);

          // Adds the given word to the tree if it is not
          // already in the tree OR increments the appropriate
          // counter if it is already there
          void Add(ItemType 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 words of the tree in alphabetical order,
          // and next to each word, the number of times each
          // word occurs in the text
          friend ostream& operator<<(ostream &out, const WordTree& rhs);

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

The destructor, output operator, and public member functions Add, countWords, and distinctWords must each be implemented in terms of a recursive private member function. Three of these operations (all but Add) must visit every node in the tree. One of these must use preorder traversal, one must use inorder traversal, and one must use postorder traversal. You must decide which method to use for each function, but use comments to document the type of traversal used. Also, use const wherever it is appropriate.

Write the definition for the constructor for WordNode inside the structure definition in the header file, and do the same for the constructor for WordTree. Use initialization lists in these constructors.

The WordTree class may have only one member variable, root, and it must be private. It contains the address of the root node of the tree.

The header file should be called wordtree.h, and the definitions for member functions must be in file wordtree.cpp. You must use the main function words.cpp. You must finish writing this file and create the other two files yourself.

I suggest that you read in the data character by character using get. Here are some test text files to use doi.txt (Declaration of Independence), and odyssey.txt (Book 1 of The Odyssey). Their results should look like the doi_results.txt and odyssey_results.txt files on the website.

What to Turn In

Please submit a zip file in the format FirstInitialLastName-CS2-149-A6.zip to Canvas. For example, go ahead and zip the file as EAmbrosio-CS2-149-A6.zip. In addition to the code, please provide:

- A brief description of notable obstacles you overcame.
- A detailed description of the design decisions you made.

Remember that most computing tasks take longer than expected. Start this assignment now!