CS 1410 Introduction to Computer Science – CS2
Section 1: MWF 10:30 a.m. – 11:20 a.m.
Section 2: MWF 1:00 p.m. – 1:50 p.m.
Instructor: Xiaojun Qi
Assignment #12 Binary Trees and Recursion

Given: Friday, April 18, 2014
Due: 11:59 p.m. Saturday, April 26, 2014
Total Points: 30 points

You are required to develop a spell checker using a binary search tree, which makes the performance of testing each word in a document to be very fast. You are supplied with a dictionary of about 14,000 words, which are saved in a file "**dictionary.txt**" in alphabetic order.

You must create a binary search tree, which contains the following public methods:

- **[4 points] Copy Constructor**
- **[4 points] Destructor**
- **[2 points] Insert**: Insert a string into the binary search tree.
- **[2 points] Find**: Return true if the string is found in the tree. Return false otherwise.
- **[4 points] Size**: Return a count of the number of nodes in the tree.
- **[4 points] Height**: Return the height of the tree, which is defined as the length of the path from the root to the deepest node in the tree (i.e., the maximum distance from all leaf nodes to the root).

You must implement each of these six functions using recursion. Please refer to Chapter19Example.zip posted on canvas for implementation details on some of these functions.

Write a driver program to do the following tasks:

1. **[4 points]** Read the file "**dictionary.txt**", where each word is on a separate line, and store these words into a binary search tree. **After reading the dictionary into the tree, have your program report the size of the tree (i.e., the number of nodes) and the height of the tree.** Since the dictionary is in the alphabetic order, you cannot sequentially read in each word and insert these words in the tree (i.e., it will essentially be a linked list if doing so). **In other words, you must insert words in a random order to make a tree that can be efficiently searched.**
   Hint: One suggestion is to read the dictionary words into a `vector<string>` and then use the member function `random_shuffle` of the vector library to accomplish the goal. **Refer to Section 16.5 (Introduction to the standard template library) for detailed explanation on the member functions (pages 1006 through 1009 of the seventh edition or pages 1000 through 1013 of the eighth edition).**
2. **[4 points]** Read the file "**letter.txt**" and output to the console all the misspelled words (i.e., any word not found in the dictionary). Keep in mind the grader will use a different file when doing the grading. You'll need to remove any punctuation characters from the words in the "**letter.txt**" file, **the following eight characters should be sufficient: " , : . ! ? ( )**
   Hint: One suggestion is to read the word into a string and then use the member functions of the string class to remove any of the eight characters. Finally, convert the word into all small letters before searching the word in the binary search tree. **Refer to pages 809 and 810 of the seventh**

edition or pages 818 through 820 of the eighth edition for detailed explanation on the string class member functions.

3. **[2 points]** Demonstrate the correctness of all the public member functions by **using the first 10 randomized strings.**

The following figure illustrates the concept of the height of a tree.