Garrett Mestel
C202
Programing Assignment 5
Due 7-21-2017

The overall problem is simple. We have a file and we want to see how many words are spelled correctly and how many are not. We also want to know the average of how many times the words are checked against the words in the dictionary. To do this though, we will need to read in both a dictionary and the file we want to check. It is also stated that the dictionary should use binary search trees.

There are two main steps in the algorithm. The first is reading in the dictionary. The dictionary the problem uses contains just the words and not any definitions of those words. All we need to do for this one is have a list for each letter and read in the next word and add it to the correct binary search tree. We do not need to worry about getting each list in alphabetical order. The second half is reading in the file we want to check. To do this we need to go line by line. First, seeing if the line is empty and if it is not removing any special characters other than apostrophes, because some of the words do have them but we do need to check that it is not at the beginning because no word starts with an apostrophe. The last part is to take each word and see if it is in the binary search tree. To make this work we will need four counters; one to count the word found, one for words not found, and two counting the amount of string comparisons for both other counters. These would each be changed by the result of comparing the final word to the dictionary. What we want to program to output is the first two counters and the average or the second two counters divided by its respective first counter.

The program is designed to store the dictionary in a custom binary search tree. There is an array of twenty six trees, one for each letter. Also this custom class has a special search method that takes in a long (type) array and sets the first element to the number of comparisons that were used to find it.

Using the test file the output is (Test files are in github)

```
914759 The Avg of the word found is 16.351742918080063
63832 The Avg of the word not found is 11.374530016292768
```

Comparing this to the result of doing the same thing but with linked lists.(See Programing Assignment 4 on github)

```
914750 The Avg of the word found is 3549.9994971303636
63841 The Avg of the word not found is 7442.381525978603
```

The words found and not found are about the same which is what you would want when you only change the data structure and not the data itself. The big difference is the comparison. Using the BST takes almost no comparisons compared to the linked list. The reason for this is that the linked list takes O(n) and the BTS only O(log n). Even with the BST being really unbalanced, as long as at least one node has two child nodes the average time will be faster.