Jeremy Roberts

Programming Assignment 4
Due: 3/25/2017

The Problem:
Implement a spell checker. Using the randomized dictionary (*random_dictionary.txt*) given in the list of files, read in the dictionary (words in the random_dictionary.txt file) into an array of 26 MyLinkedList objects, one for each letter of the alphabet. Then read in the book (*oliver.txt*), examine the first character of each word, and traverse the correlating list of the 26 MyLinkedList.
Each word read from the book should be searched in the corresponding LinkedLists. If it is not found, this word is either misspelled, or not in the dictionary. Collect "words found" and "words not found", as well as the number of comparisons made for each. Display the resulting counters as well as the averages for "Comparisons (Found)" and "Comparisons (Not Found)".

The Algorithm:
The algorithm creates an array of 26 MyLinkedLists each representing a character of the alphabet. The first character of each line (word) defines its list to be added to. The word is stripped of all non-alpha characters when being added. Each word of the ebook "Oliver" is read in stripping all special characters and matched against the dictionary word by word using the "indexOf" method. If that value is not -1 it is assumed found, the collector for "WordsFound" is incremented, and the "indexOf" value is added to the "Comparisons Found" collector. If the "indexOf" did return -1, the "WordsNotFound" is incremented, and the total size of the correlating MyLinkedList is assumed to be the number of total comparisons is added to the "Comparisons Not Found" collector.

The Results:
The script takes approximately 26 seconds to complete and results in
915,594 words found
62,997 words not found
2,147,483,647 comparisons were made for words found
469,305,787 comparisons were made for words not found

Average comparison for words found were about half of those of the average comparisons for words not found:
3244 comparisons for words found
7450 comparisons for words not found

Observations:
Having tried a few methods to parse the words and all yielded nearly the same results (+/- 2000 matches). Using a regular expression against both the dictionary and the words read in from the ebook to ensure they were being handled the same way. 915,594 was the best result. The real observation from this assignment is just how different the results can be from using nearly identical methods. Time remained consistent amongst all three methods attempted.