Assignment Four

SpellChecker using MyLinkedList

CSCI-C 202

Tabitha Rottman

April 11, 2017

Abstract

In Programming Assignment Four, we are utilizing an implementation of the List interface, specifically LinkedLists. A LinkedList is used when an element must be deleted or inserted into the list initially created. Within Assignment Four, we are using LinkedLists to compare a text file to another text file specifically searching for common words between the two. In order to perform this process, unnecessary information (spaces, symbols, uncommon words, etc) must be removed from the lists, thus leaving only the information requested. This act of removing the unnecessary information is why the implementation of LinkedLists within Assignment Four is required.

This is done by first creating a LinkedList that is limited to 26 characters. We limit this array due to there only being 26 characters in the alphabet by the use of a for loop. Next, we incorporate the two text files: random_dictionary.txt and oliver.txt into the program. The set of for loops within the TestLinkedList class allows the program to compare each character of each individual word within the oliver text and determines if the said word can be found within the dictionary text file. If this is true, the word is counted and the program moves on to the next word. If not, the word is "thrown out" of the potential data set. In other words, if the word from the text file is not found within the dictionary, it is removed. As this process continues, the sum of wordsFound, wordsNotFound, compsFound (comparisons) and compsNotFound accumulates and the total sums are displayed via an s-out statement. Also displayed is the average comparisons found and average comparisons not found. This is done by the use of initializing a simple calculation.

Using Assignment Four's program allows the user to "accurately" determine the number of words found within the oliver text file that also in the dictionary. Unfortunately, this "accuracy" maybe not be 100% as there are a few problems. Depending upon how each individual's code is written, there will be a margin of error. This is due to a possible number of words that were either misspelled, contractions (can't, won't etc) or space/symbols and was not accepted, as the word was not exactly as it is in the dictionary text file. There is also the fact that LinkedLists, though faster than some methods, can process slow. The time complexity of LinkedLists is O(n), moderately average. This program took about 33 second to successfully run.