Pa2 Kinh Hoang
khh39 151004179

First

Summary-

The program is a dictionary statistics generator for multiple files. Given several matching dictionary and statistic files, given in the form of a mapping file, the program generates statistics about the number of words in the data file matching a dictionary word and the number of words the dictionary word is a super word for.

Data Structure-

For my main data structure in this program, I decided to use a Trie (also known as a prefix tree). This structure consists of various nodes, consisting of a character, 26 children (for the possible next letters of the alphabet), and information on whether the node represents a word or not.

I chose this data structure, because it allows me to arrange all of my dictionary words in relation to each other. A word starts form the root node, and continues down the trie character by character, until it hits the last node, which holds a copy of the word itself. When searching for the node, one only has to move down the trie character by character, and check the final node to see if it is a word or not, meanwhile having access to all prefixes of the word through this path.

In this fashion, given an unknown string of length n, by moving through the trie character by character, we can determine if it exists in the trie and update the various counts in O(n) time.

This optimization is extremely useful, as one might expect to receive extremely large data files, each of which with words that need to compare with all prefixes of the word in the dictionary file.

Runtime-

Given a max word length of k, dictionary size of n words, and maximum number of words in the data or dictionary file m: I compute the runtime and space complexity.

The runtime complexity of one dictionary/data file run is **O(kn+ km)** or **O(k(n+m))**.

- With insertions into the data structure using **add_word** being **O(k)** run **n** times.
- And statistic updates using **matchString** being **O(k)** run **m** times.

This gives us an overall complexity of **O(lk(n+m))** is l is the number of dictionary/datafile pairs.

The space complexity for this program is **O(kn)**

Challenges-

The most difficult part of this assignment was figuring out the nuances of C string manipulation. Over the course of this assignment, I've found that working with C strings come with very many

nuances that can cause a large amount of confusing errors that are not present in a higher level language.