The purpose of this spell-checking project was to create a spell-checking application that utilizes a hash table. While creating this spell-checking project, I have learned more about handling and creating hash tables, manipulating strings, passing vectors to a search function, and storing the contents of a file into a vector. These practices have allowed for insight into how different abstract data structures are used in the industry.

The biggest part of this project was to create a hash table class in order to make searching for strings more efficient. The keyword being efficient. In order to achieve this, the hash table utilized singly list chaining with 100 buckets. This allowed for conflicts with different keys that are generated from the list of words that the table is built for. After the table has the different linked lists populated, then we can more efficiently search specific buckets depending on what the hash key generated for the word that the user inputs are, which results in a much smaller search scope.

Another aspect of this project was to suggest potential values that the user meant to input but testing the following against the hash table as well: swapping adjacent characters in the user’s word, inserting a single character in between two adjacent characters in the user’s word, deleting a single character from in the user’s word, and replacing a character in the user’s word with another character. With this, each of those functions returned a list of potential strings to be passed to the searching algorithm, and any potential words found in the hash table are then we will output said a potential word to the screen.

In order to get the “dictionary”, or list of words to create the hash table for, a text file was read into a vector of strings, which was then passed to the hashing function to create the hash table. The text file was formatted so that each word was on a new line, and the program finds the new line character to determine where each word ends.

The biggest focus of this project was efficiency. While working to make the program more efficient, light was shed on the purpose of the different abstract data structures and algorithms used in the software development/engineering industry and how they can make programs more efficient. In this case, if there were no hash table then we would wind up with a program that more or less performed a comparison of the user’s input string and every string that was listed in the “dictionary” text file, which would result in a slower run time and lots of resource consumption. With abstract data structures, while they can seem complicated, there are similarities that can be noticed; an example of this would be that a bucket/singly chained list-based hash table has a similar look and structure to an adjacency list graph structure. Another step into both of those data structures would show that they each use a linked list structure when being implemented. These structures allow for more efficient solutions to the problems that occur out in the world of technology.