**Programming Project Report**

Name: Riley Nadwodny

Date: 3/3/2020

**Academic Integrity Statement:** I pledge that I have neither given nor received unauthorized help on this programming assignment.

**Problem Statement:**

The primary goal of this programming assignment was “to gain experience with recursive binary searching, reading, and processing text files, and object oriented design.” The program to design was to read in a sample passage from a book and calculate the reading level of the passage based on the rarity of words use. Any sample passages that had more common words had a lower reading level compared to those that had rare or very rare words. The only input from the user is which sample passage they would like to read in, and if the file was not found, the program told the user that the file could not be found and terminates.

**Design:**

This program required only one class with two methods and a handful of private variables. For this, the class was called “Dictionary” with a Dictionary.h file and a Dictionary.cpp file for the class header and function delcarations in the .h files, and function definitions in the .cpp file. The private variables in this class are an index variable and two arrays: the first array of size 300 and the second array of size 1000. The first method in the Dictionary class reads in a file of the 1000 most common words, and then the sample text of the user’s choice. If it cannot find the user’s chosen sample text, i.e. the user puts in “junk.txt”, the program terminates. If it does find the user’s chosen file, the file is then opened and while the infile stream continues to read words in, the method stores the data into the array, then increments the index of the array and stores the next word into the next indexed location. This continues until the method has reached the end of the file.

The second method is a recursive binary search method that, in theory, finds the word read in from the sample file in the array of the 1,000 most common words, gets the rank, and returns it. This method is called in the first method which reads in the words from a file; as the words are read in, the binary search is called. The result is returned to a temporary variable that holds the total value of the sample passage. Prior to this, a word counter variable is declared and every time a word is read in, the word counter increments by one. At the end, the total difficulty value of the sample passage is divided by the word counter and returned to the user.

**Implementation:**

The implementation of this program took the full two weeks, and even then, it was not enough time to complete the program and make it function as process. To start, skeleton code was written out for the Dictionary class, and pseudocode was written for the method to read in the words from a sample passage file and store it into an array. Next, that method was heavily tested and fixed to make sure that all the capital letters were lowercase, and all the punctuation was removed from the file. The objective of doing this was to make the word easier to find in the array of word rankings.

Next, the binary search method was implemented using sample code from John Gauch’s “search.cpp” file from the source code section on his website. This code was adapted to compare the string read into the array of word rankings, and then returning the correct value if the word was found. If the word was not found, a ranking of 1001 was returned instead.

**Testing:**

This program was tested with each sample passage provided by John Gauch. This includes “sample1.txt”, “sample2.txt”, “sample3.txt”, “sample4.txt”, and “sample5.txt” from the CSCE 2014 website. Input was also tested with “junk.txt” to make sure that the program outputs the correct message to the user and terminates. This program did not work as expected, primarily due to the binary search function. Sample output is included in the “project3output.txt” file provided in the .zip folder to reduce text in this programming report.

* Describe how you tested your program.
* What were the normal inputs you used?
* What were the special cases you tested?
* Did everything work as expected?
* Include sample input/output from your program.

**Conclusions:**

The overall result of this assignment is a failure, and an incredibly frustrating one at that. The program does not perform as expected, and as such, does not work correctly. Incorrect reading difficulty levels are given for 2 sample files, and the other 3 sample files just crash upon attempting to calculate their difficulty levels. This project was incredibly frustrating to get the binary search working (or supposed to get it working) because it feels as though we were barely taught about binary searching and how to implement it, and especially implementing it in the scope of this project. Next time, the project will be started sooner, and the coding of any search methods will start sooner in order to provide more time to figure out the implementation of the methods. This project took the full two weeks to do.