**Project 3**

**A brief description of notable obstacles you overcame.**

Not being able to use loops in the functions was difficult and took a lot of time to think through. I started this project by writing pseudocode and making a plan of how to write the functions, but each of my functions involved using for loops to iterate through the text file and words.

I ended up using several helper functions in place of each loop I would have used. The FAQ was especially helpful in doing this, because I realized I could introduce functions that took in extra parameters that involved counter variables to be able to iterate through arrays and the words.

In the **theJumbler** function, I used distinct helper functions to compare the permutations of words with each of the result and dictionary arrays. Initially, I did not compare with the result array and ended up having repeats, because my functions did not check whether the word had already been added. This resulted in errors because some words were repeatedly added to the results array.

Also, comparing with results helped with efficiency of the program as it was a shorter list than dict to traverse through.

The largest notable obstacle I overcame was in my **permute** function. Initially, I had two lines of code that set **inDict** a Boolean (around line 70). This caused my functions to be called twice the desired amount of times: once when setting the Boolean and the second when calling it in the if statement. I noticed this does not affect the inResult function.

**A list of the test data**, **along with the reason for each test.** Y**ou must note which test cases your program does not handle correctly.**

Main Function used for testing:

int main()

{

string results[MAXRESULTS];

string dict[MAXDICTWORDS];

ifstream dictfile; // file containing the list of words

int nwords; // number of words read from dictionary

string word;

dictfile.open("words.txt");

if (!dictfile) {

cout << "File not found!" << endl;

return (1);

}

nwords = lexiconBuilder(dictfile, dict);

//cout << nwords << endl;

// cout << dict[nwords - 1] << endl;

cout << "Please enter a string for an anagram: ";

cin >> word;

int numMatches = theJumbler(word, dict, nwords, results);

//cout << endl<< "numMatches: " << numMatches << endl;

if (!numMatches) {

cout << "No matches found" << endl;

}

else {

divulgeSolutions(results, numMatches);

}

return 0;

}

|  |  |
| --- | --- |
| **Tested Word** | **Reason** |
| “a”  “lemons”  “straightforward”  “lumberjack”  “abovementioned”  “kloo”  “AAA”  “on” (no)  “sz”  “rat” (art, rat, tar)  “tea”  “race” (care, acre)  “glean” (angle, angle)  cout << dict[nwords – 1];  zygote | 1 letter  no matches (not in list)  very long word  no matches  no matches  repeated letters  repeated letters (3), no matches  2 letters, 2, matches  2 letters, no matches  3 letters, 3 matches  3 letters, 4 matches  4 letters, 3 matches  5 letters, 3 matches  Check last word  last word in the list |