**Obstacles:**

I struggled with finding a way to keep a counter when using recursion in the function to test if the dictionary word was an anagram. I was able to solve this by creating a recursive helper function which I pass the counter to as a parameter and then adding 1 to the counter parameter as needed when recursively looping through the same function.

**Test Data:**

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

cin >> word;

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

**if** (!numMatches) {

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

}

**else** {

divulgeSolutions(results, numMatches);

}

**return** 0;

Input: rat

*Tests to see if it can handle 3 combinations at once*

Input: regardless

*Tests to see if the program can handle lengthy words and return no matches*

Input: hek

*Tests to see if it can handle small words even if there were no matches*

Input: ngthlye

*Tests to see if it can handle longer words and find the matches*

Input: hey there

*Tests to see how program handles spaces. As I use getline, it takes the first string as the input word and finds anagrams for that*

Empty words.txt file

*Tests to see if program can handle an empty dictionary file*

string results[MAXRESULTS];

string dict[MAXDICTWORDS];

string exampleDict[] = {"kool", "moe", "dee"};

**int** numResults = theJumbler("kloo", exampleDict, 3,

results);

assert(numResults == 1 && results[0] == "kool");

*Tests to see if the program can handle words with multiple of the same letter*