The first major obstacle that I overcame was strictly using recursion without much help from functions of the string class. I had mapped out in my head a way to complete the second function theJumbler using some functions from the string class and thought it would be simple to make the same function I had mapped out without those functions. I was wrong. It took me some time to rework my program in order to not use excess functions from the string class. Another major obstacle I overcame was straightening out my helper functions. The first time I ran my completed code for the function theJumbler, I got the wrong output, so I had to go back through all of my helper functions to figure out how to fix it. This took a long time as each of the helper functions was relatively complex, and some of them called each other.

Test Cases

Test cases will be written in the format

functionName():

(parameter1, parameter2, parameter3, etc.) //reason for test case

//the first function is a little tough to show parameters for as one of them is an entire file, so I will simply describe the test case

lexiconBuilder():

(dictfile, dict) //dictfile contains words.txt given in the project spec

(dictfile, dict) //dictifle contains an empty text file

(dictfile, dict) //dictfile contains a file with my own list of words

(dictfile, dict) //file contains more than the maximum number of words

(dictfile, dict) //file contains exactly the maximum number of words

theJumbler():

(word, dict, nwords, results) //word is a word with no anagrams

(word, dict, nwords, results) //word is a word with one anagram

(word, dict, nwords, results) //word is a word with multiple anagrams

(word, dict, nwords, results) //word is a word with anagrams that are not contained in dict

(word, dict, nwords, results) //word is a word that is not found in dict

(word, dict, nwords, results) //dict is an empty array

(word, dict, nwords, results) //word is a word that has an anagram that is on the list multiple times

(word, dict, nwords, results) //word is on the list multiple times  
(word, dict, nwords, results) //word is not a real word but has anagrams

(word, dict, nwords, results) //word is not a real word and does not have anagrams

divulgeSolutions():

(results, numMatches) //results has words

(results, numMatches) //results is empty

(results, numMatches) //results is greater than MAXRESULTS

(results, numMatches) //results is equal to MAXRESULTS

(results, numMatches) //word has no anagrams

(results, numMatches) //word is not in the list

(results, numMatches) //word has anagrams not in the list

(results, numMatches) //word has multiple anagrams

(results, numMatches) //dict was an empty file