**PROJECT 3 REPORT**

***NOTABLE OBSTACLES:***

1. I struggled to simultaneously count words and input them into an array within one function (makeDictionary). It took me trial and error to develop different types of helper functions, including one that filled the dictionary and another that attempted to count how many words were inputted into the array after filling. I settled on a single auxiliary function with a counting parameter that returned the number of times a word was inputted into the dict[] array.
2. I had difficulty following the given pseudocode on the Project3FAQ to generate permutations. Eventually, I realized that I didn’t need to directly modify the input variables within my function. Instead, I could create temporary variables and use these as arguments for my recursive calls.
3. After I had generated permutations and made a dictionary, I could not figure out how to use these functions to correctly increment my results[] array within my permutation function in order to fill the array. To solve this, I developed two helper functions: one that compared each permutation to the results[] array to prevent duplicate results, and one that compared each permutation to the dict[] array to determine whether to input it into results. I call these two functions whenever a permutation is generated (when wordSize==0).

***TEST DATA:***

* **Testing using given “words” dictionary file**

|  | CHECKING ANAGRAMS OF INCREASING SIZE  (tests memory allocation rules, run time length, etc.) |
| --- | --- |
| “a” | Shows that a single-letter word will have at most one match  Should return shuffleChars == 1  Should return  Matching word: a |
| “on” | Shows that program works for a 2-letter word, which can have at most 2 anagrams  Should return shuffleChars == 2  Should return  Matching word: on  Matching word: no |
| “rat” | Shows that program works for a 3-letter word with 3 anagrams  Should return shuffleChars == 3 Should return  Matching word: rat  Matching word: art  Matching word: tar |
| “brag” | Shows that program works for a 4-letter word with 3 anagrams  Should return shuffleChars == 3 Should return  Matching word: brag  Matching word: grab  Matching word: garb |
| “gnat” | Shows that program works for a 4-letter word with 2 anagrams  Shows that the program will only return two results, even if “gnat” is present within multiple longer words  Should return shuffleChars == 2 Should return  Matching word: gnat  Matching word: tang |
| “elbow” | Shows that the program works for a 5-letter word with 3 anagrams  Should return shuffleChars == 3 Should return  Matching word: elbow  Matching word: below  Matching word: bowel |
| “listen” | Shows that the program works for a 6-letter word with 3 anagrams  Should return shuffleChars == 3  Should return  Matching word: listen  Matching word: silent  Matching word: tinsel |
| “observe” | Shows that the program works for a 7-letter word with 3 anagrams  (takes longer to run)  Should return shuffleChars == 3  Should return  Matching word: observe  Matching word: obverse  Matching word: verbose |
| “arrogant” | Shows that the program works for a 8-letter word with 2 anagrams  (takes a LONG time to run- over a minute)  Should return shuffleChars == 3  Should return  Matching word: arrogant  Matching word: tarragon |
|  | CHECKING WORDS THAT SHOULD HAVE NO RESULTS  (and longer words that didn’t have any anagrams in this dictionary file) |
| “macadamia” | Takes over an hour to finish  Should return shuffleChars == 0  Should return  No matching words |
| “regardless” | Takes almost 2 hours to finish  Should return shuffleChars == 0  Should return  No matching words |
| “abcd” | Shows that a made-up 4-letter word will have no results  Should return shuffleChars == 0  Should return  No matching words |
|  | CHECKING WORDS WITH REPEATED LETTERS  (confirm that there are no duplicate results) |
| “kloo” | // word with double letters  Should return shuffleChars == 1  Should return  Matching word: look |
| “AAA” | // word with triple repeated letters  Should return shuffleChars == 1  Should return  Matching word: AAA |