CS2302: Lab 1 – Recursion Report

# Introduction

The problem presented today is trying to find all anagrams from an input word, compare it to a dictionary of words in a separated text file, store them in a set, and then print the set out alphabetically and with a time describing how long it took to process the word.

# Design and approach

The first approach taken to solve this anagram problem/search was to see how to permutate the letters from the word (“***s***”) so it forms combinations of that word as well as the anagrams and then compare these combinations words to the file’s words so that it returns the anagrams found within the file and discard the combinations of words which aren’t liable words.

For the first step, the scramble method was introduced from the CS 2302 book, where it takes “***s****”* and an empty string, then recursively attempts to find all combinations of the word “***s***” by subtracting one letter from the original word every recursion, thus returning all combinations of the word “***s***”. By implementing this algorithm to the code, it helped with getting all combinations from the word “***s***” and creating a list where all combinations of the word “***s***” is stored in.

Secondly, the next step was to attempt to get the word to be read and stored in a set by using the “***.read()***”, “***.split()***”, and “***set()***” functions. “***.read()***” function reads the entire file. Using “***.split()***” function alongside, it will read the file while the whitespaces are being eliminated. Doing this will clear any unwanted space, so the words can be stored in a set. Then, ultimately, with “***set()***” function, it stores the wanted words into a set.

Thirdly, now that the two sets were created (using “set()” on the combinations of word list), a comparison is needed to find all the anagrams in the combinations of word set. So, using the “***.intersection()***” function, the two sets can be compared and any word that are not found in each set are discarded, only taking the ones that are found in both the compared sets.

Afterwards, using the “***sorted()***” function makes sure that the words in the set are in alphabetical order.

# Experiments

For the first experiment, taken from the CS 2302 book, the word “cat” was used, which lead to not many anagrams but acceptable output. A picture containing screenshot

Description automatically generated

On the second experiment, the example word “poster” was used which led to a moderate amount of anagram outputs than the previous test word. A screenshot of a social media post

Description automatically generated

For the third experiment, the other example word “university” was used which took longer to permutate and compare. A close up of a logo

Description automatically generated

And lastly, a custom word, “spear” was used found online which permutated 13 different anagrams. A screenshot of a social media post

Description automatically generated

# Conclusion

In conclusion, what was learned from this project was how to use recursion to do permutations in order to solve this specific problem, how to work with newly learned language of Python to do basic tasks like counting time to calculate the code’s process or naming and creating methods in Python, and how to think recursively when using recursion. Although, it was difficult at first where sets and methods were involved since errors were prompt to happened, it was manageable and got through the code with favorable results.

# Source Code Page

A screenshot of a cell phone

Description automatically generated

# Academic Honesty Certification

I certify that this project is entirely my own work, I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

Axel E Manzanares – September 09, 2019