**CS 2302 Data Structures**

**Fall 2019**

**Lab #1**

Due: March 10th, 2019

Professor: Olac Fuentes

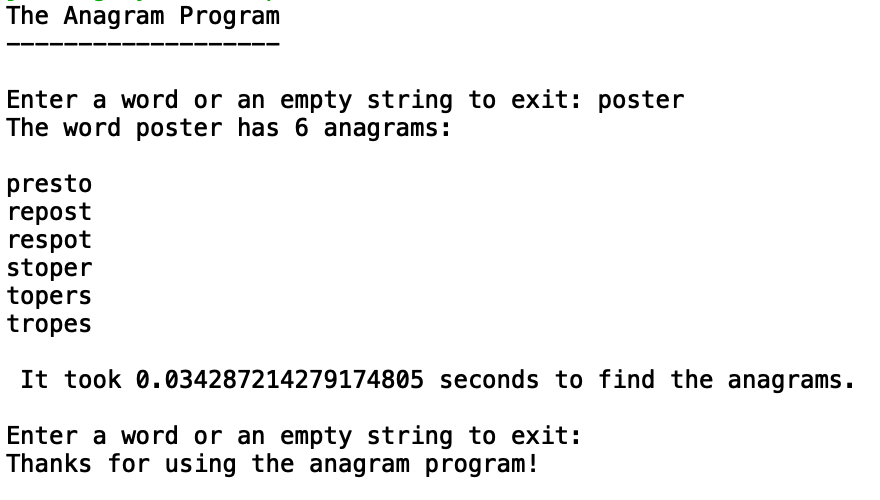
TA: Anindita Nath

Name: Jared Aguayo

**Introduction**

In this lab we were asked to work with recursion in order to find anagrams in a large word set using an inputted word. The main point in this lab is to get a deeper understanding of how to work with recursion and how to manipulate your code to become more efficient.

**Proposed Solution Design and Implementation**

Design – 

This is what the user will see, the program will continue to ask for a string until an empty string is given. If a number is inputted it will ask for a string again until one is inputted, and the program can take a capital word or letters

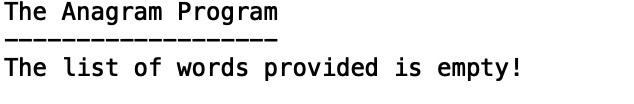
Operation #1 - After downloading the text file given into a set, I asked for a word to be inputted that would be used to find the anagram. I began by using a program similar to the one in zybooks that scrambles letters except I had the outputs be put into a set to make sure there were no duplicates. This method to produce scrambled words that will be used as permutations for the next step. After a set of permutations is returned, I change it to a list before sending it to my next method to find the anagrams. The method I use to find anagrams takes the list of permutations and checks if the permutation is in the large set of words and after this is checked and added to a set if it is found or not being added to a set if it is not found there is a recursive call that will return a smaller list of permutations each time eventually leading to the permutation list hitting length 0 which will validate the base case of the list of permutations being length 0 and it will then return the anagrams.

Operation #2 – The permutations and anagrams produced will never have duplicates.

**Experimental Results**

* Case #1

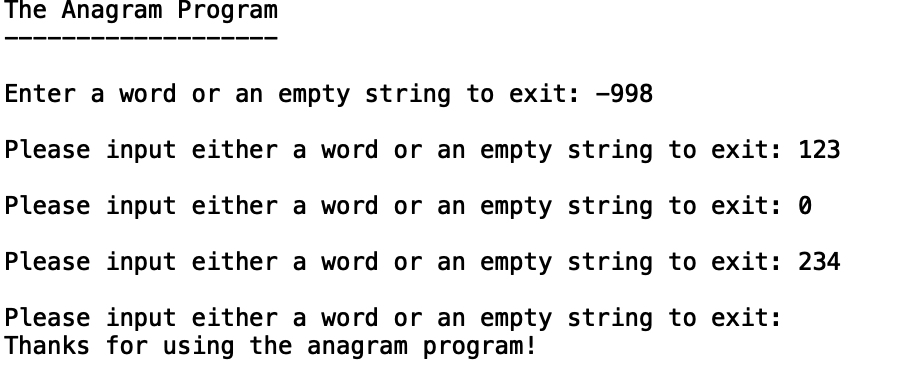
Giving an empty list to the program:



Description: I added a line of code to test this by setting the set variable that I use to take in the text file to an empty set.

* Case #2

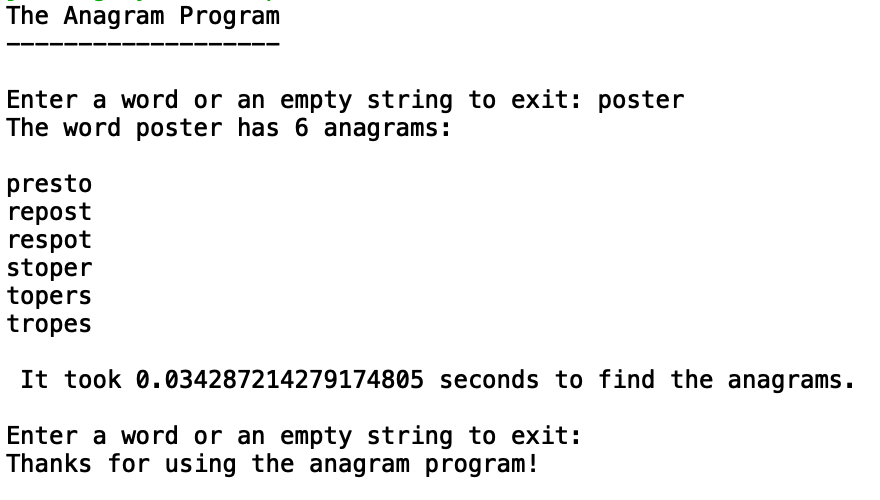
Giving the program an input other than a string:



Description: When the program runs it will ask the user for an input, when it asks for an input just input any number or negative number and it will ask the user you re input a string

* Case #3

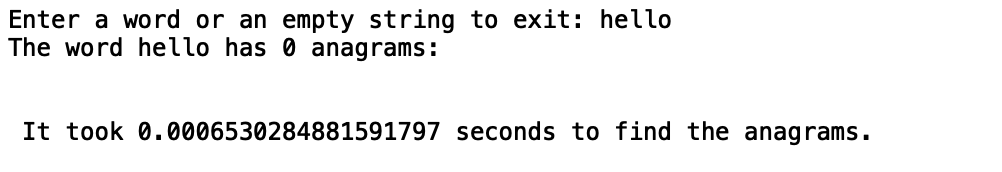
Giving the program a string that has anagrams:



Description: When it the program runs and prompts the user to input a string just input the string ‘poster’, and it will output all of the anagrams

* Case #4

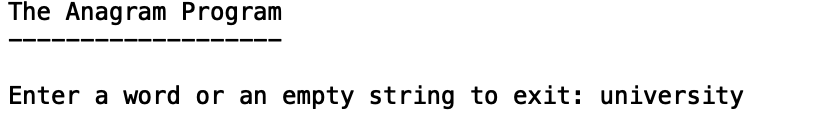
Giving the program a string that has no anagrams:



Description: When the program runs and asks for an input just input the string ‘hello’ and it will output that there are no anagrams for this string

* Case #5

Large input string to find anagrams:

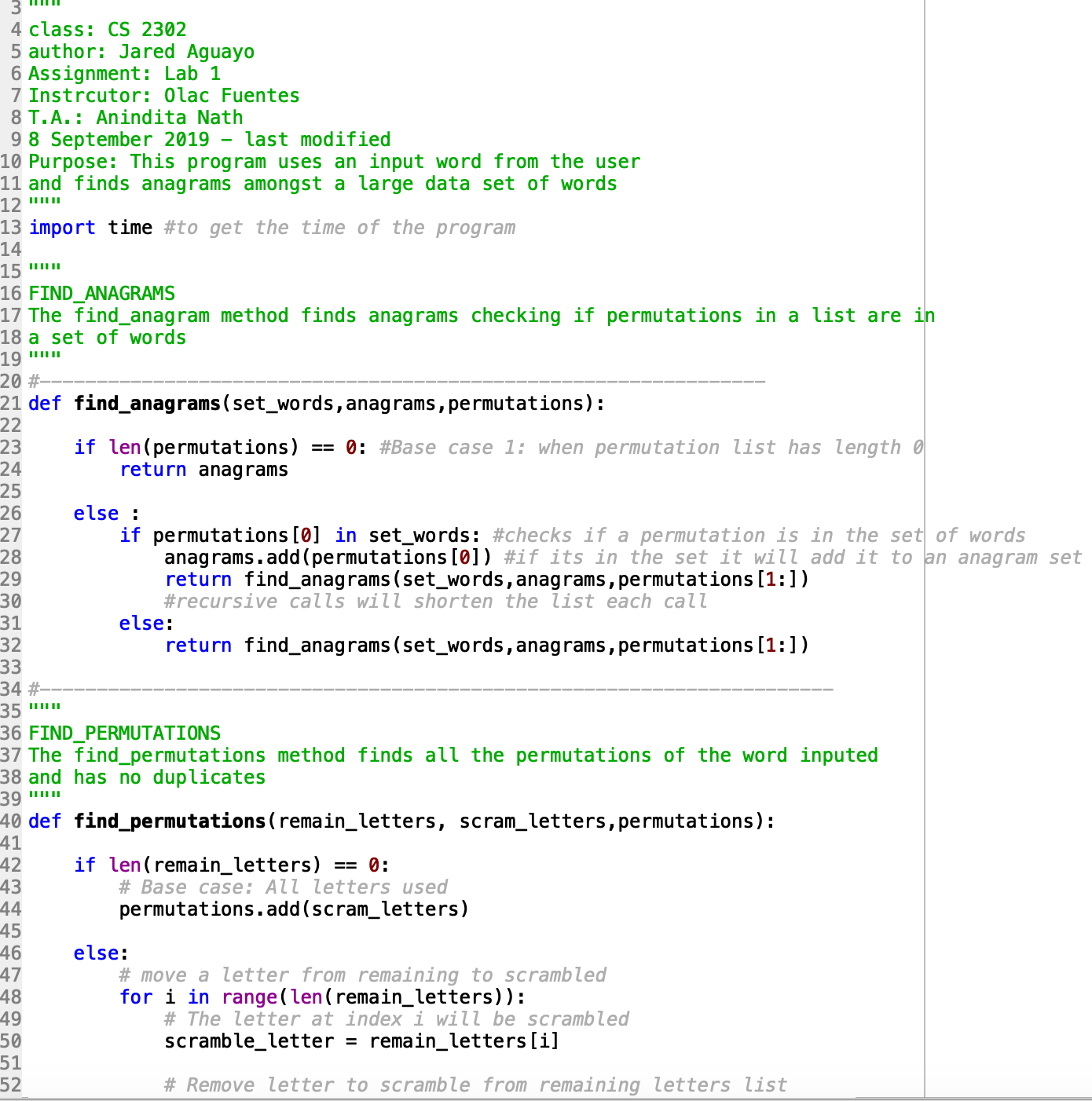


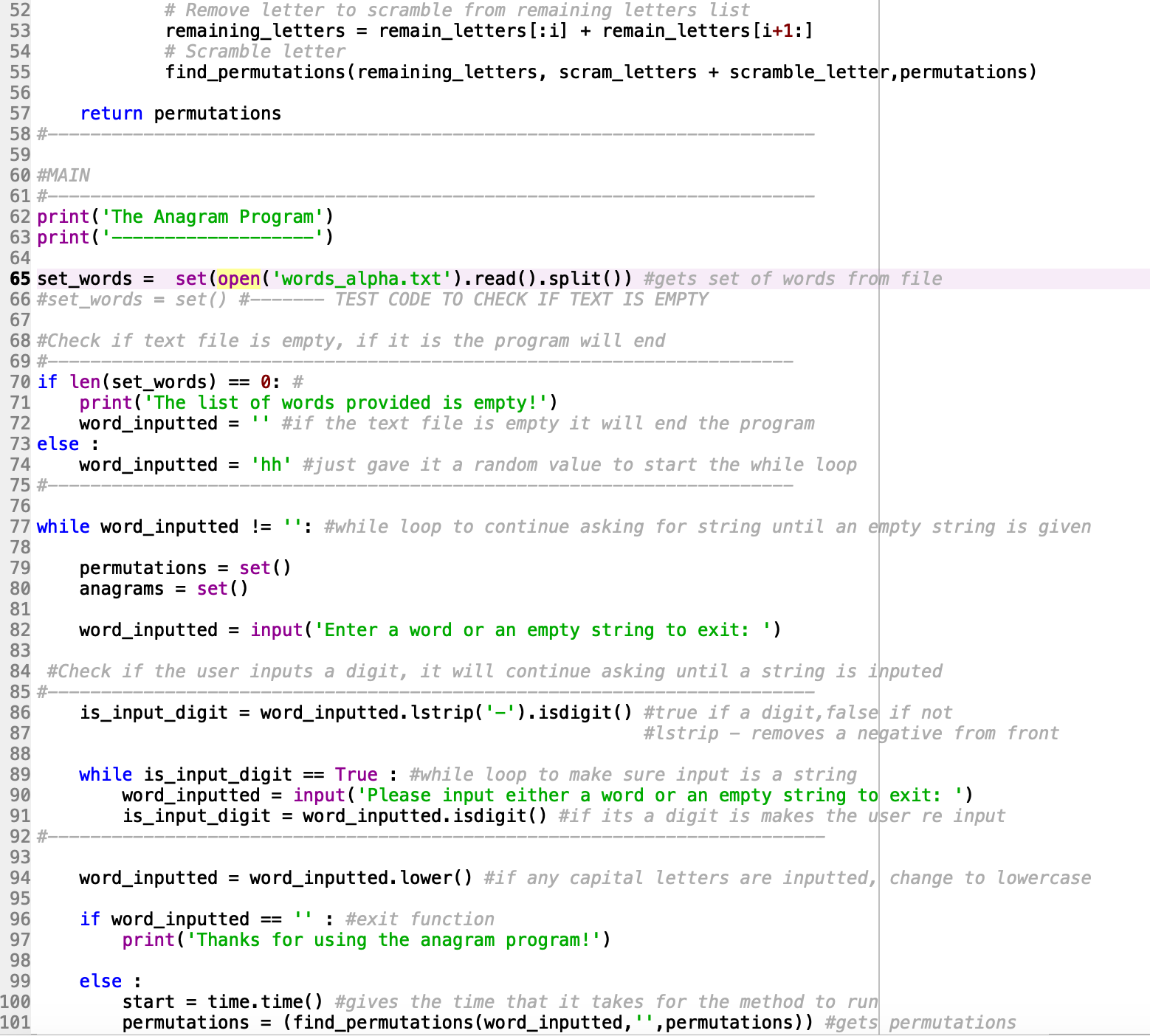
Description: When you input a string that’s over 6 characters it reaches maximum recursion since the permutation is list is to long

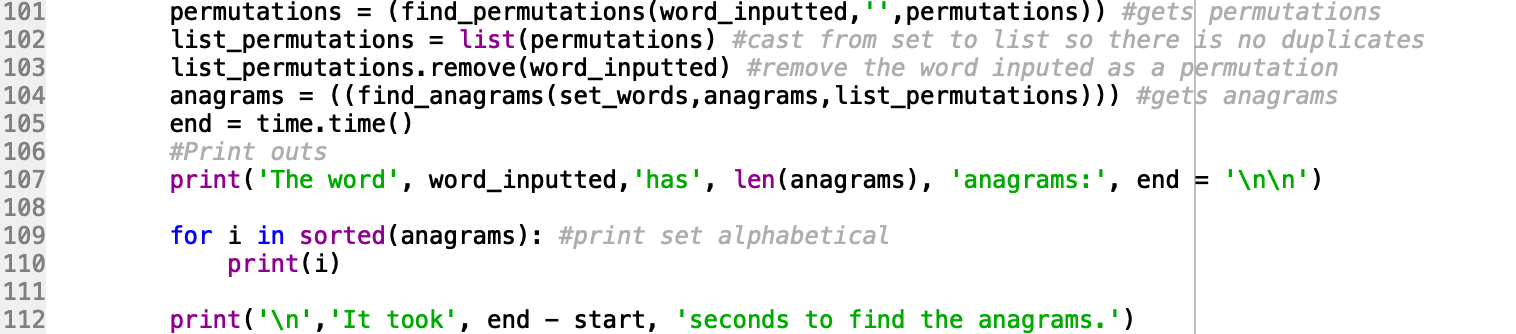
**Conclusions**

I’ve learned how to efficiently use recursion to traverse a list as well as learn more about what sets do and the properties that they entail. I also ran into an issue with maximum recursion when it came to finding the anagrams which helped me understand problems like this that can arise when programming and it led to me finding other ways that wouldn’t produce this outcome. I’ve learned more about how recursion works at a deeper understanding than I previously had known.

**Appendix**

****

****

****

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