CS 2402 Data Structures

**Lab 1 - Anagrams**

Roque, Thomas ID#88732157

horizontal line

# 

# Introduction

An anagram is a permutation of the letters of a word that produces another word. Design a program that asks a user to input a word and then prints all the anagrams of that word. This program is to be efficient and will not end until user enters an empty string.

## Implementation

The 1st Challenge is to figure out how to compare two strings to determine if they are anagrams of each other. For this I got the user input and named it “comparie”. This was the word i would be testing against words from the text file.

In this case “comparie”==stuff[0]

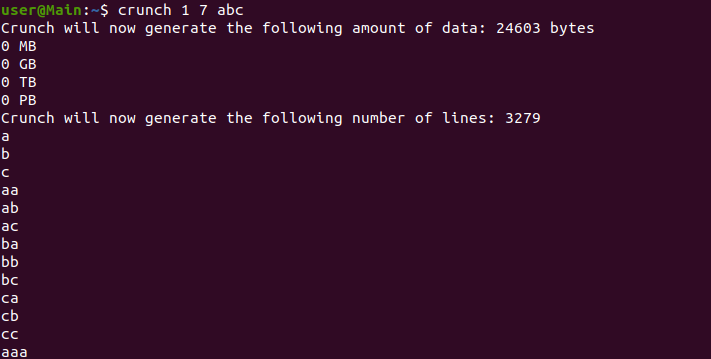
After this i would sort the word “comparie” and store the output in “str1” to avoid repeatedly sorting the given word.

At this point i would compare the sorted comparie (str1) with a word from the list by sorting it but not storing its output, only comparing it. At this point I would print out the word in the list if they were equal.

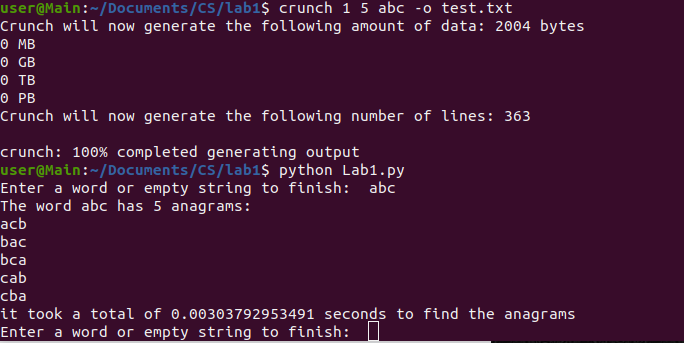
To avoid printing out the given word “comparie” if found in the list I would also simply check to see if “comparie” was equal to the word being checked.

### Test cases

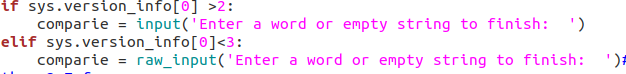
I was unsure of how to implement part 2 of the lab and my workstation has low computing capabilities. This created a challenge for me as my computer would lock up and become unresponsive until it was restarted or several minutes had past. To avoid this i would generate a word list myself.



By using “crunch” i would generate a wordlist that my computer was able to handle. My computer could only handle around 3-5 hundred words but the image above is an example of how the utility was useful for my testing.

With the word “abc” you can see all anagrams that are associated with it: 

Another error that came up was that i know that the class is using python 3 but at the time of the lab i was using both python 2 and 3 so i needed to add a check to see what python version the user is running.



# Summary:

Unable to complete part 2 the code tries to run as efficiently as possible. By checking string sizes to limit the amount of checks the program tries to omit as many words as possible to only compare words of similar length. The program works until the user no longer wishes to use it and displays such as shown in the lab document.