CS 2302 Lab Report

Lab 4\_Option B

Jose Lujan

Lab 4 Option B

Hash-Table

**Objective of the Lab:**

Generate the hash table with the words that we will be getting from the file, which contains 3423 words. Once the hast-table has been created, the user will be asked to give a word as an input. The function will look for all the different anagrams that this word has.

Create a function that returns the number of anagrams that a given word has, and return the word that has the greatest amount of anagrams in a certain file.

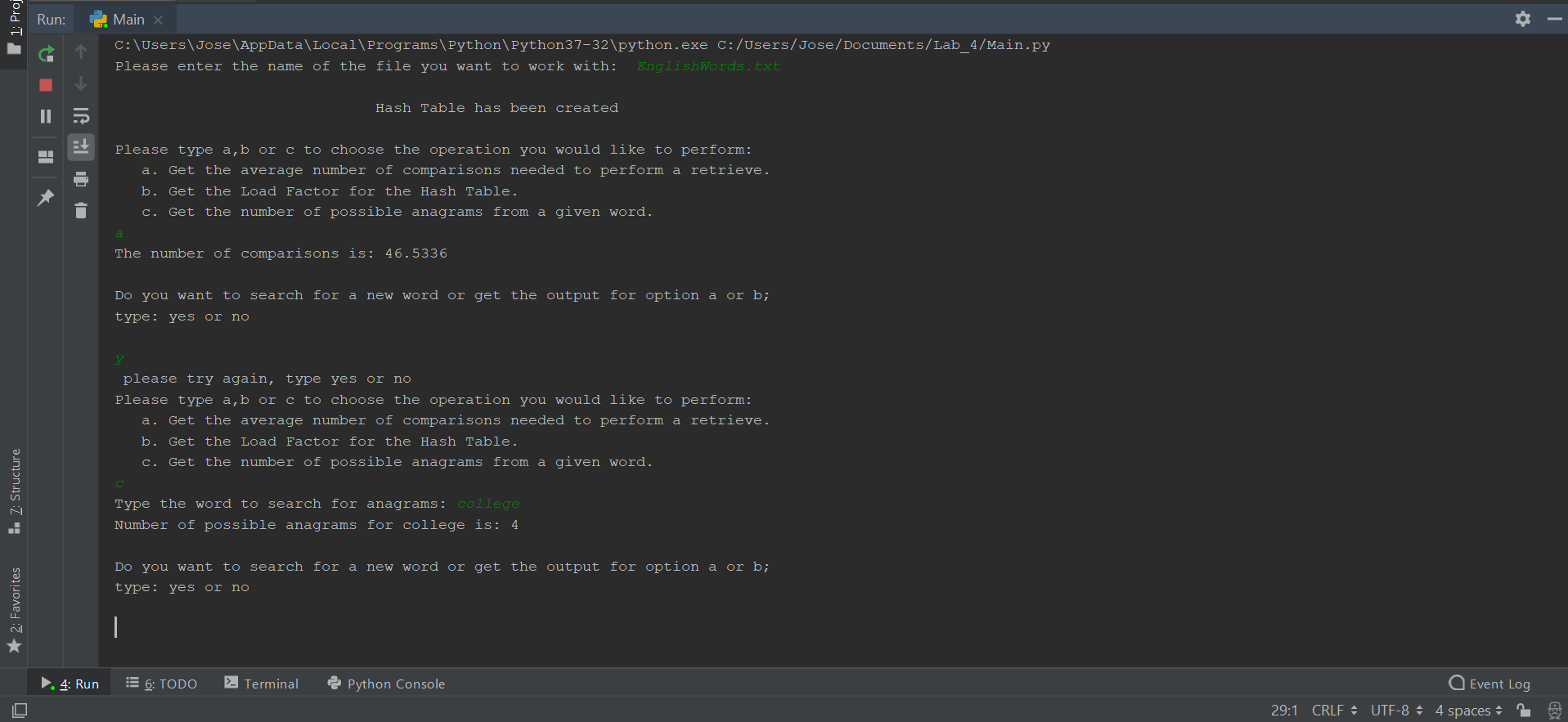
**Solution:**

# Lab 4 option B  
# Name: Jose Lujan  
# ID: 80572649  
# class: cs2302  
# class time 10:30-11:50  
  
from Hash\_Table import HashTable, HashTableNode  
  
  
# Creates the Hash-Table, using the words obtain from the file named: "EnglishWords.txt"  
def create\_hash\_table(filename):  
 try:  
 english\_words = HashTable(5000)  
  
 # Open file and read lines  
 file = open(filename, "r")  
 line = file.readline()  
  
 while line:  
 english\_words.insert(line.rstrip())  
 line = file.readline()  
  
 return english\_words  
 # Prints a message, to let know the user that he enter the wrong file name  
 except FileNotFoundError:  
 print("Please try again, name of file is incorrect")  
  
  
# Generates all possible permutations from a given within the working file  
def get\_permutations(word):  
 if len(word) <= 1:  
 return word  
 else:  
 permutations\_list = []  
 for perm in get\_permutations(word[1:]):  
 for i in range(len(word)):  
 permutations\_list.append(perm[:i] + word[0:1] + perm[i:])  
 return permutations\_list  
  
  
# Returns the amount of anagrams for a given word by the user  
def count\_anagrams(word, table):  
 permutations = get\_permutations(word)  
 count = 0  
  
 for i in range(len(permutations)):  
 if table.search(permutations[i]):  
 count += 1  
  
 return count  
  
  
# Returns the number of words in a cell  
def get\_num\_col(node):  
 count = 0  
 temp = node  
  
 while temp is not None:  
 count = count + 1  
 temp = temp.next  
  
 return count  
  
  
# Returns the amount of comparisons needed to find a word  
def amount\_of\_comparisons(table):  
 number\_of\_cols = 0  
  
 for i in range(len(table.table)):  
 number\_of\_cols = number\_of\_cols + get\_num\_col(table.table[i]) // 2  
  
 return number\_of\_cols / len(table.table)  
  
  
# Returns the load factor  
def get\_load\_factor(table):  
 num\_elements = 0  
 for i in range(len(table.table)):  
 temp = table.table[i]  
  
 while temp is not None:  
 num\_elements = num\_elements + 1  
 temp = temp.next  
  
 return num\_elements / len(table.table)  
  
  
def main():  
 filename = input("Please enter the name of the file you want to work with: ")  
 keep\_going = True  
 hash\_table = create\_hash\_table(filename)  
 print("")  
 print(" Hash Table has been created")  
 print("")  
  
 if hash\_table is not None:  
 while keep\_going:  
 print("Please type a,b or c to choose the operation you would like to perform:")  
 print(" a. Get the average number of comparisons needed to perform a retrieve.")  
 print(" b. Get the Load Factor for the Hash Table.")  
 print(" c. Get the number of possible anagrams from a given word.")  
 answer = input()  
  
 if answer == 'a':  
 print("The number of comparisons is:", amount\_of\_comparisons(hash\_table))  
 elif answer == 'b':  
 print("The Load Factor is:", get\_load\_factor(hash\_table))  
 elif answer == 'c':  
 word = input("Type the word to search for anagrams: ")  
 print("Number of possible anagrams for " + word + " is:", count\_anagrams(word, hash\_table))  
 else:  
 print("The can only type a,b or c")  
  
 loop = input("\nDo you want to search for a new word or get the output for option a or b; \ntype: yes or no\n\n")  
  
 if loop == 'yes':  
 keep\_going = True  
 elif loop == 'no':  
 keep\_going = False  
 else:  
 print(" please try again, type yes or no")  
  
  
main()

# Hash Table Node class  
class HashTableNode:  
 def \_\_init\_\_(self, item, next):  
 self.item = item  
 self.next = next  
  
  
# Hash Table class  
class HashTable:  
  
 def \_\_init\_\_(self, table\_size):  
 self.table = [None] \* table\_size  
  
 # hashing function  
 def hash(self, word):  
 lower\_case\_word = word.lower()  
 sum = 0  
 # Adds values for each letter in a word to  
 # store anagrams in the same chain  
 for i in range(len(lower\_case\_word)):  
 sum = sum + ord(lower\_case\_word[i])  
  
 return sum % len(self.table)  
  
 # Inserts new words in the Hash Table  
 def insert(self, word):  
 loc = self.hash(word)  
 if self.table[loc] is None:  
 self.table[loc] = HashTableNode(word.lower(), None)  
 else:  
 self.table[loc] = HashTableNode(word.lower(), self.table[loc])  
  
 # Searches for words in the Hash Table  
 def search(self, k):  
 loc = self.hash(k)  
  
 temp = self.table[loc]  
  
 while temp is not None:  
 if temp.item == k:  
 return True  
  
 temp = temp.next  
  
 return False

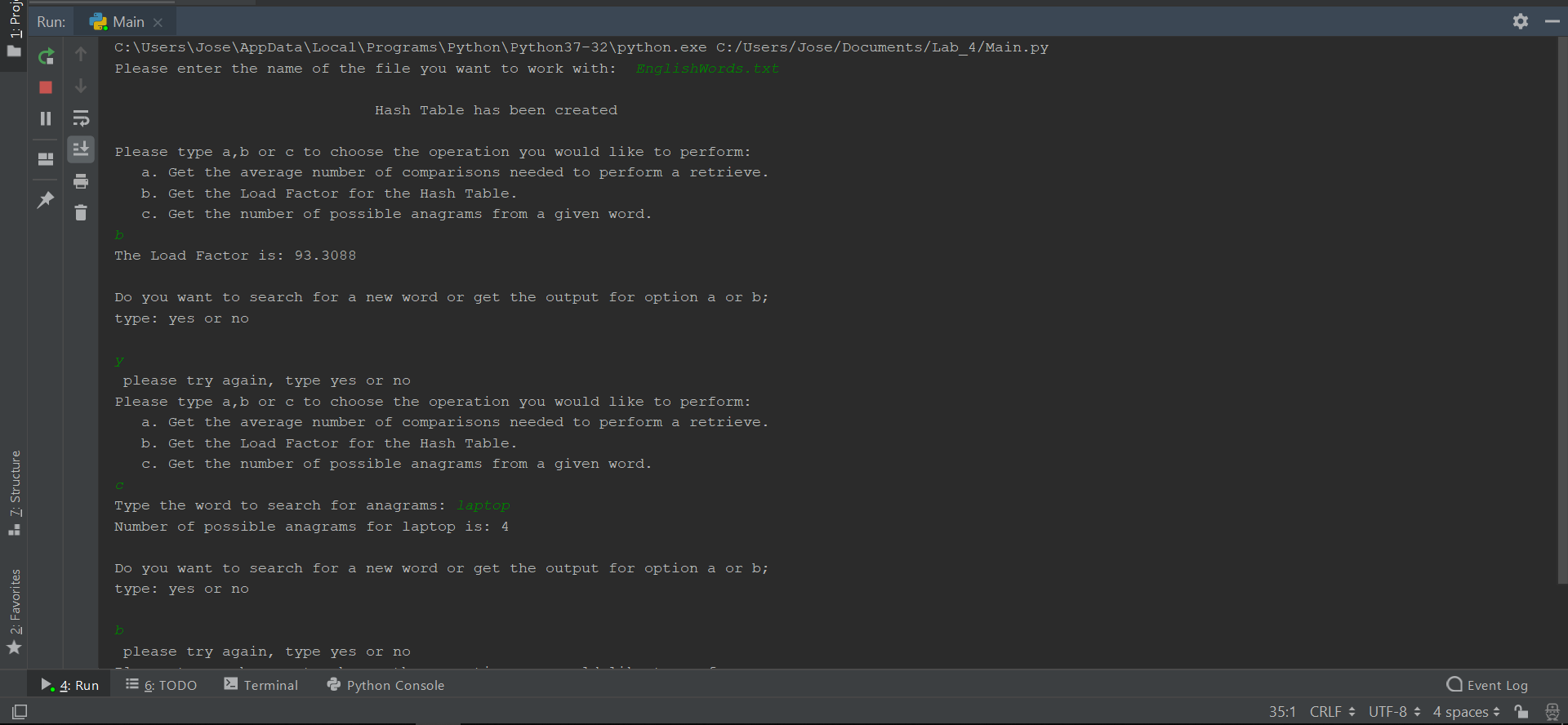
Test #1:

This test creates the Hash table, by obtaining all the words from the file “EnglishWords.txt”, The first operation is to get the number of comparisons the hash table does, and the last operation in this test is to get back the number of anagrams that a word has, in this case is: “college”. The program return that the word college has 4 possible anagrams.



Test #2:

In this test, I used the same file, but this time I got in return the Load factor that the program uses to store data in the nodes. The second operation is to tell us how many anagrams does the word : “laptop” has, in this case is 4. The last operation in the test was to input any letter that wasn’t in the options listed, in which prints out a text that says to try again , will all the options in which he can continue the test.



Test # 3

For the last test, I enter a file which is not in the code, and it return a text. That asks the user to try again because the file he entered is incorrect.

