CS 2302 Lab Report

Lab 2\_Option B

Jose Lujan

Lab 2 Option B

Linked-List

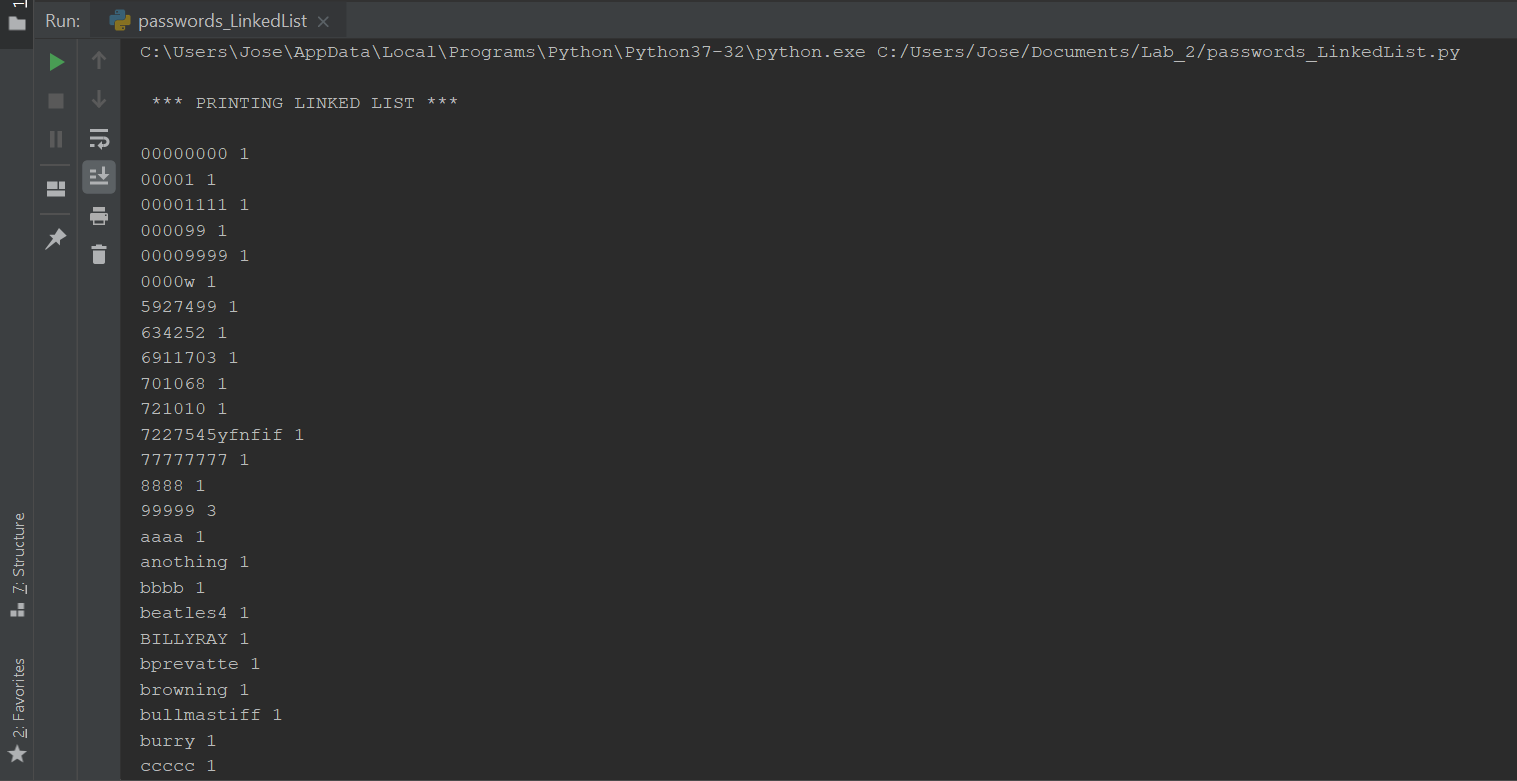
Objective of the Lab:

The objective of this lab is to generate a linked-list with all the items from a file which is given by the instructor. This file contains 10 million usernames and passwords, but in this case we will only work with 10 thousand.

Solution:

class Node (object):  
 password = ""  
 count = -1  
 next = None  
  
 # Constructor for a Node  
 def \_\_init\_\_(self, password, count, next):  
 self.password = password  
 self.count = count  
 self.next = next  
  
  
# Linked List class.  
class LinkedList:  
  
 #Constructor for a Linked List  
 def \_\_init\_\_(self):  
 self.head = None  
 return  
  
 #Add a Node to a linked list  
 def add(self, value):  
 if self.head is None:  
 self.head = Node(value, 1, None)  
 return  
  
 current = self.head  
  
 #If head is not None, it will be added at the end of the list.  
 while current is not None:  
 #Check if password is already present on the list  
 if current.password == value:  
 current.count = current.count + 1  
 return  
 if current.next is None:  
 current.next = Node(value, 1, None)  
 return  
 current = current.next  
  
 # Print the whole linked list and the number of times each password is repeated  
 def print(self):  
 temp = self.head  
  
 while temp is not None:  
 print(temp.password + " " + str(temp.count))  
  
 temp = temp.next  
  
 return  
  
 # Returns the length of the linked list  
 def length(self):  
 count = 0  
 current = self.head  
  
 while current is not None:  
 count = count + 1  
 current = current.next  
  
 return count  
  
  
# Function that reads a .txt file and creates a linked list out of it.  
def solution\_a(file\_name):  
 pw\_list = LinkedList()  
  
 # Open file and read first line  
 file = open(file\_name, "r")  
 line = file.readline()  
  
 # Loop will go trough every line in the file  
 while line:  
 # Creates an array that holds the username and the password  
 user\_and\_password = line.split()  
  
 # Check if there is a password and adds it.  
 if len(user\_and\_password) > 1:  
 pw\_list.add(user\_and\_password[1])  
 line = file.readline()  
  
 # Returns Linked List  
 return pw\_list.head  
  
  
# Function that creates a dictionary out of a .txt file.  
def solution\_b(file\_name):  
 pw\_dict = {}  
  
 # Open file and read first line.  
 file = open(file\_name, "r")  
 line = file.readline()  
  
 # Loop will go through file until there are no lines left.  
 while line:  
 # Creates an array to hold username and password.  
 user\_and\_password = line.split()  
  
 # Checks if a password exists  
 if len(user\_and\_password) > 1:  
 # Checks if password is already present in dictionary  
 if user\_and\_password[1] in pw\_dict:  
 pw\_dict[user\_and\_password[1]] = pw\_dict[user\_and\_password[1]] + 1  
 else:  
 pw\_dict[user\_and\_password[1]] = 1  
 line = file.readline()  
  
 # Returns dictionary  
 return pw\_dict  
  
  
# Function that sorts linked list using Bubble Sort Algorithm  
def bubble\_sort(pw\_list):  
  
 # Loop will go through the whole linked list.  
 for i in range(pw\_list.length()):  
 j = 0  
 current = pw\_list.head  
 prev = pw\_list.head  
 # Loop will only check for items that have not been sorted already.  
 while j < pw\_list.length() - i and current.next is not None:  
 # Case for when comparing head and head.next  
 if prev is current:  
 if current.count < current.next.count:  
 temp = current.next  
 current.next = current.next.next  
 temp.next = current  
 current = temp  
 prev = current  
 current = current.next  
  
 # Case for when comparing any node and node.next  
 if current.count < current.next.count:  
 temp = current.next  
 current.next = current.next.next  
 temp.next = current  
 prev.next = temp  
 current = temp  
  
 current = current.next  
 prev = prev.next  
 j = j + 1  
  
 # Return sorted linked list  
 return pw\_list  
  
  
# Function that returns the middle element in a linked list  
def middle(head):  
 slow = head  
 fast = head  
  
 while fast is not None and fast.next is not None:  
 if fast.next.next is not None:  
 fast = fast.next.next  
 slow = slow.next  
 fast = fast.next  
 return slow  
  
# Function merges and sorts two lists. Part of the merge sort algorithm  
def merge\_and\_sort(list1, list2):  
 sorted\_list = Node("", 0, None)  
 temp\_head = sorted\_list  
  
 # checks if any list is empty  
 if list1 is None:  
 sorted\_list.next = list2  
 if list2 is None:  
 sorted\_list.next = list1  
  
 # Merges lists according to their value (sorting them).  
 while list1 is not None or list2 is not None:  
 if list1 is None:  
 sorted\_list.next = list2  
 list2 = list2.next  
 elif list2 is None:  
 sorted\_list.next = list1  
 list1 = list1.next  
 else:  
 if list1.count <= list2.count:  
 sorted\_list.next = list1  
 list1 = list1.next  
 sorted\_list.next = list2  
 list2 = list2.next  
 sorted\_list = sorted\_list.next  
 sorted\_list = temp\_head.next  
  
 return sorted\_list  
  
# Function that uses recursion to sort a linked list following the merge sort algorithm.  
def merge\_sort(head):  
 # Base cases  
 if head is None:  
 return None  
 if head.next is None:  
 return head  
 else:  
 # Finds middle element  
 mid = middle(head)  
 # Sets the head for second list  
 after\_middle = mid.next  
 mid.next = None  
  
 # Recursive call to divide lists until each list is a single element  
 left = merge\_sort(head)  
 right = merge\_sort(after\_middle)  
  
 # Merge each pair of lists  
 sorted\_list = merge\_and\_sort(left, right)  
 return sorted\_list  
  
  
# Testing Solution A  
pw\_list = solution\_a("test.txt")  
current = pw\_list  
  
print()  
print(" \*\*\* PRINTING LINKED LIST \*\*\* ")  
print()  
# Loop prints list gotten from solution A  
while current is not None:  
 print(current.password + " " + str(current.count))  
 current = current.next  
  
  
# Testing Solution B  
pw\_dict = solution\_b("test.txt")  
  
print()  
print(" \*\*\* PRINTING DICTIONARY \*\*\* ")  
print()  
  
# Loop prints dictionary gotten from solution B  
for k, v in pw\_dict.items():  
 print(k, v)

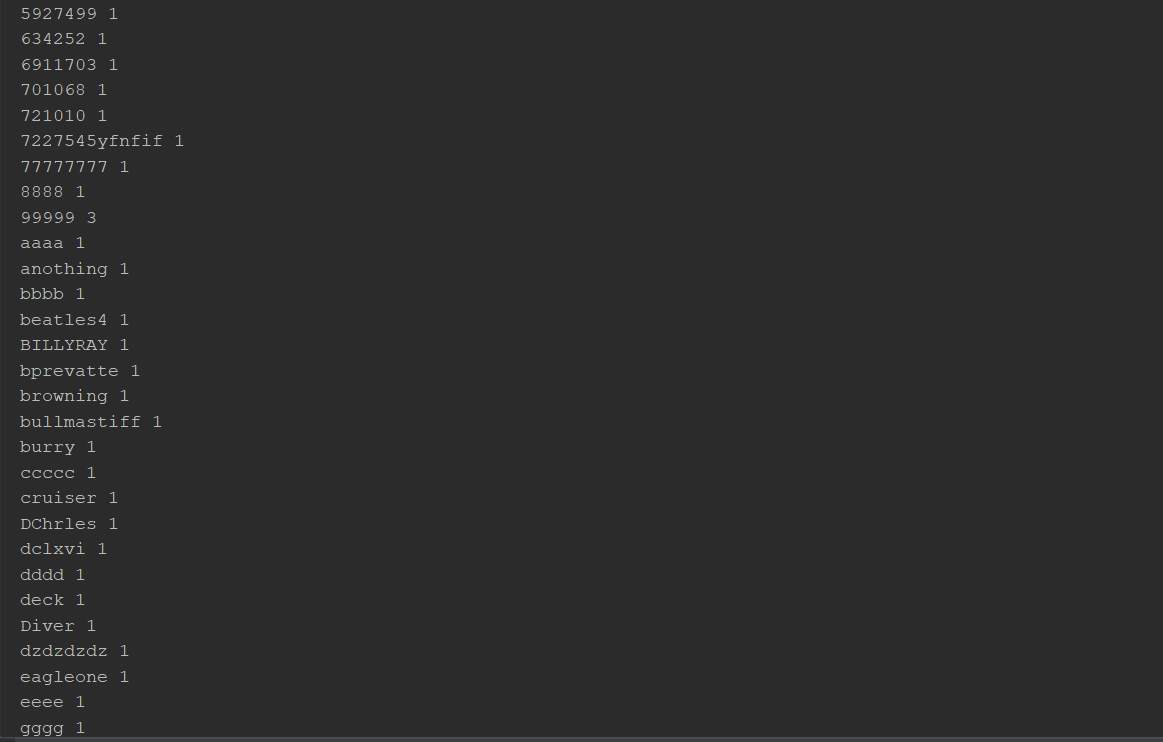
test #1

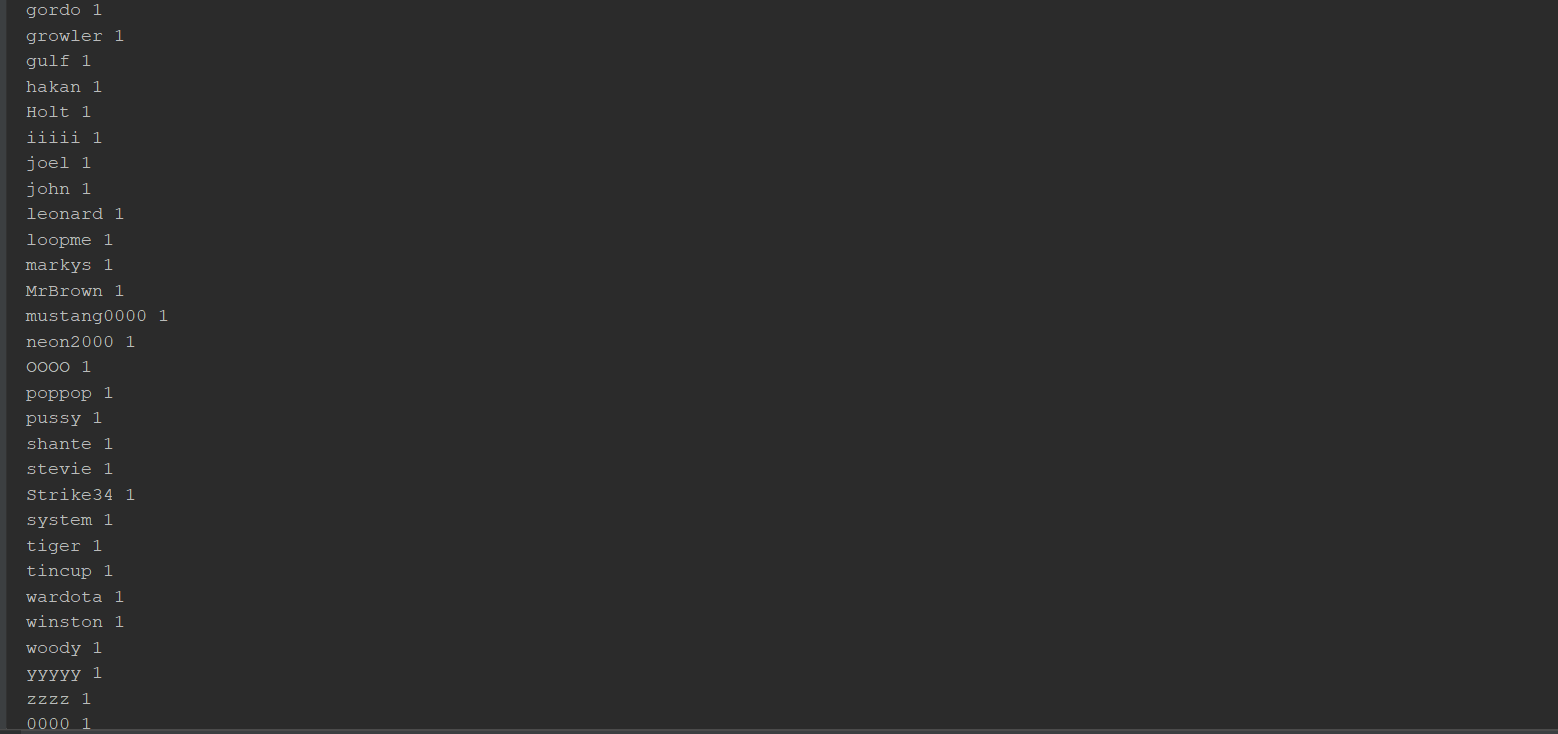




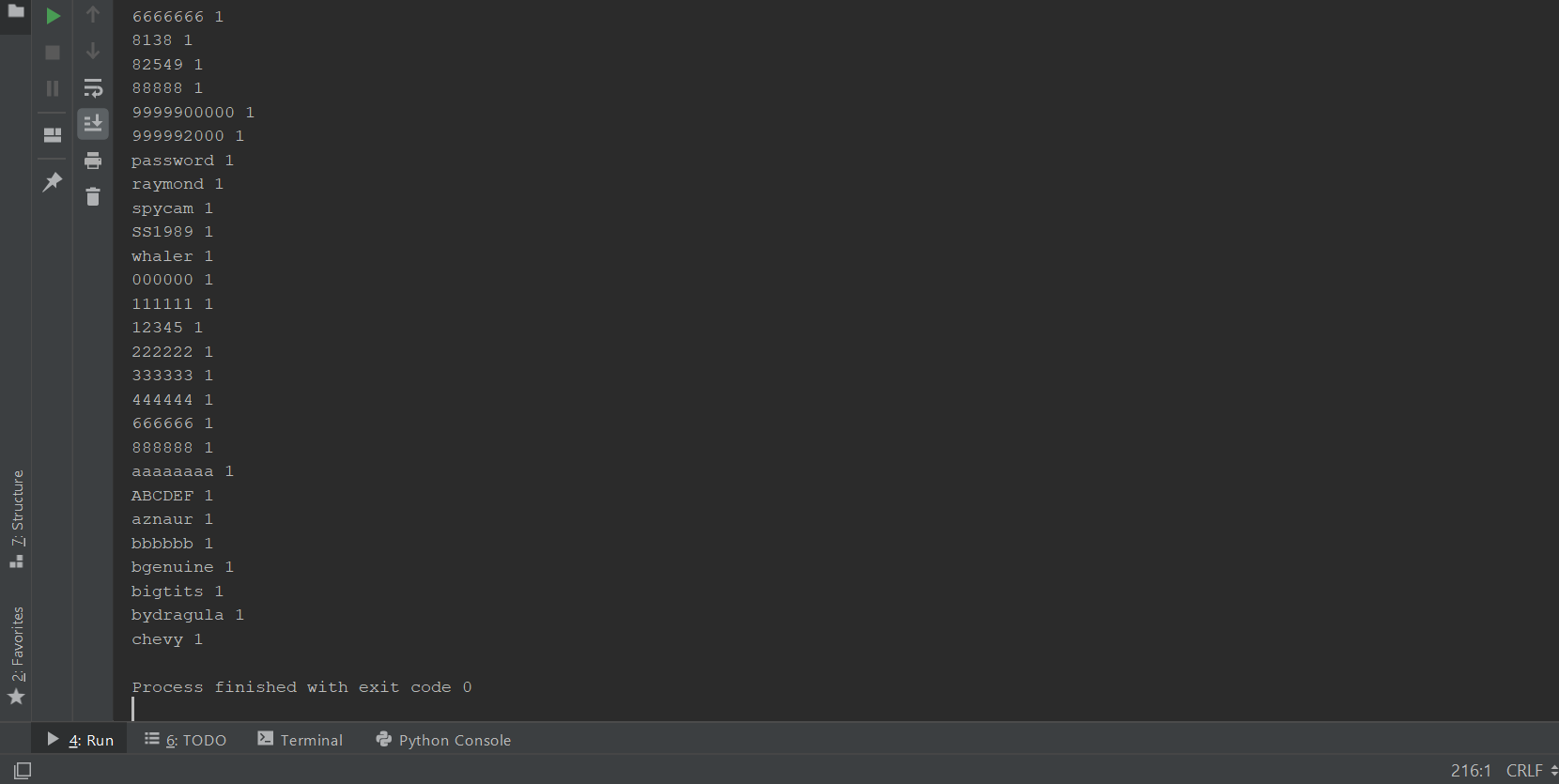










****

**Test #2**

C:\Users\Jose\AppData\Local\Programs\Python\Python37-32\python.exe C:/Users/Jose/Documents/Lab\_2/passwords\_LinkedList.py

\*\*\* PRINTING LINKED LIST \*\*\*

7953 1

88888 2

8888811111 1

888882000 1

8888888888 1

8888898 1

9999 1

721010 1

7227545yfnfif 1

77777777 1

8888 1

99999 3

aaaa 1

anothing 1

bbbb 1

beatles4 1

BILLYRAY 1

bprevatte 1

browning 1

bullmastiff 1

burry 1

ccccc 1

cruiser 1

DChrles 1

dclxvi 1

dddd 1

deck 1

Diver 1

dzdzdzdz 1

eagleone 1

eeee 1

gggg 1

gordo 1

growler 1

gulf 1

hakan 1

Holt 1

iiiii 1

joel 1

john 1

leonard 1

loopme 1

markys 1

MrBrown 1

mustang0000 1

neon2000 1

OOOO 1

poppop 1

pussy 1

shante 1

stevie 1

Strike34 1

system 1

tiger 1

tincup 1

wardota 1

winston 1

woody 1

yyyyy 1

zzzz 1

0000 1

000001 2

102030 1

11111 1

29081974 1

44444 1

66666 1

6666666 1

8138 1

82549 1

9999900000 1

999992000 1

password 1

raymond 1

spycam 1

SS1989 1

whaler 1

000000 1

111111 1

12345 1

222222 1

333333 1

444444 1

666666 1

888888 1

aaaaaaaa 1

ABCDEF 1

aznaur 1

bbbbbb 1

bgenuine 1

bigtits 1

bydragula 1

chevy 3

7270934 1

bigjos 1

bigjosh 1

cubs1 1

dorf 1

eddie76 1

bkincade 1

1957fenw 1

rockys 1

1969chevy 1

1011cjb 1

mrsrobinson 1

donna46 1

cast21 1

chev 1

dracu 1

falcon74 1

gabs 1

gismo 1

\*\*\* PRINTING DICTIONARY \*\*\*

7953 1

88888 2

8888811111 1

888882000 1

8888888888 1

8888898 1

9999 1

721010 1

7227545yfnfif 1

77777777 1

8888 1

99999 3

aaaa 1

anothing 1

bbbb 1

beatles4 1

BILLYRAY 1

bprevatte 1

browning 1

bullmastiff 1

burry 1

ccccc 1

cruiser 1

DChrles 1

dclxvi 1

dddd 1

deck 1

Diver 1

dzdzdzdz 1

eagleone 1

eeee 1

gggg 1

gordo 1

growler 1

gulf 1

hakan 1

Holt 1

iiiii 1

joel 1

john 1

leonard 1

loopme 1

markys 1

MrBrown 1

mustang0000 1

neon2000 1

OOOO 1

poppop 1

pussy 1

shante 1

stevie 1

Strike34 1

system 1

tiger 1

tincup 1

wardota 1

winston 1

woody 1

yyyyy 1

zzzz 1

0000 1

000001 2

102030 1

11111 1

29081974 1

44444 1

66666 1

6666666 1

8138 1

82549 1

9999900000 1

999992000 1

password 1

raymond 1

spycam 1

SS1989 1

whaler 1

000000 1

111111 1

12345 1

222222 1

333333 1

444444 1

666666 1

888888 1

aaaaaaaa 1

ABCDEF 1

aznaur 1

bbbbbb 1

bgenuine 1

bigtits 1

bydragula 1

chevy 3

7270934 1

bigjos 1

bigjosh 1

cubs1 1

dorf 1

eddie76 1

bkincade 1

1957fenw 1

rockys 1

1969chevy 1

1011cjb 1

mrsrobinson 1

donna46 1

cast21 1

chev 1

dracu 1

falcon74 1

gabs 1

gismo 1

Process finished with exit code 0

**Conclusion:**

In this lab basically what I had to do is to create a dictionary to store all the passwords, and the amount of times that each password will appear in the file. Once the dictionary is filled, it creates the linked list, it a way that has two values, first the password and next the amount of time it appears, and at last the next node, which connects to a new node.

Tests: