**Introduction**: For this Lab, I have to find a way to get the passwords of 99 users and to do that I need to create my own passwords with digits of 0-9 find out if it’s the correct one after concatenating the user’s salt value to it and then hashing the created password.

**Proposed** **Solution**: For this lab I had 2 ways I wanted to solve this problem. My first attempt involved the use of the random function in order to get many different combinations of digits from the range of 0 to 9; I also used the random function to create a password with 3 to 7 amounts of characters. However, using the random function proved to create a bigger issue which was that there was a higher chance that I would get repeating numbers and not produce the ones that I needed instead; This reason is why I had to switch to the next solution provided below.

My second attempt would require me to go in a more linear approach which will create all possible solutions instead of repeating some of them. Also, when I use the file with the info of the users then I would need to read and store it in a list for future use (when creating all possible passwords).

**Implementation**: After reading the given code I first needed to create my global variables which will hold the password file information and the password that I will be creating, also those variables will use empty lists which contain strings later on. Now with my variables created I needed to read the password file and to be able to do that was to use a while loop instead of a for loop because it proved later on that the for loop was much slower at producing the results of the of the program. When reading the file, I need to make sure that when storing the info, I need to split at each comma and when there is a whitespace after each of the hash values. Assuming that I have the password file in a list I now need make a different for loop outside the previous while loop. This for loop will go in a range of 3 to 7 which uses the number of digits a password requires (at least 3 digits and no more than 7 digits). This for loop would then call the method random password which makes the new password that might be the same as the users.

In the random password method, I use 3 parameters, one is an empty string, the size of the times the for loop iterates, then the limit of digits that can be used (0-9). Inside the method, I made two if statements, the first one to check if the times that I iterated is greater than 0 then and if its true, the if statement would use a for loop to iterate 9 times and using the integer i (holding the number of iterations) will be used to call the same method recursively while concatenating the I to the empty string, subtracting 1 from the size that was iterated in the first for loop and passing the same limit of digits I could use. The other if is for when the size of iterations is equals to 0 which means that once its 0 then there is a password made. Inside that if there is a method call to check the password and passes the current string I have made.

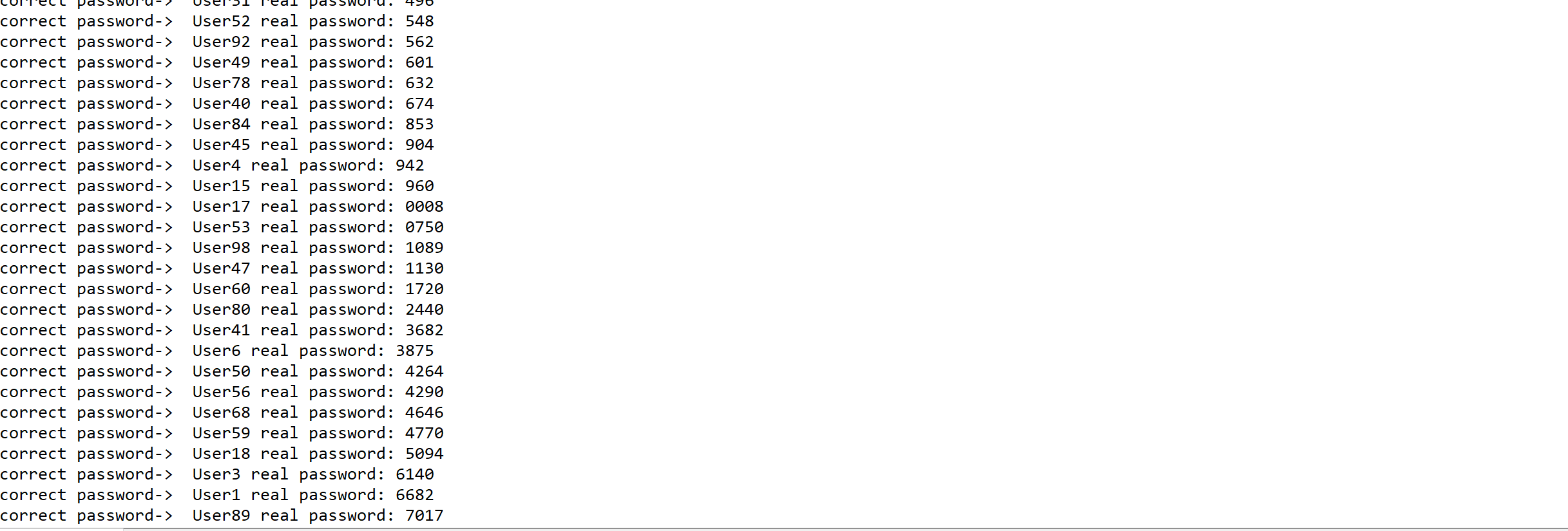
In checking if the password is the same one as the user, I first need to use a for loop that goes in range of the length of the file in order to compare each of the 99 users. In the loop, I need to compare my current password and adding the salt value to it, after those two strings are together, next I need hash it and with the current password hashed I compare the new hash with the users hashed password. If its true that both are the same, then the global variable that holds my new password will store the correct password in it and prints it as it finds more passwords. As it prints then my original list that holds the passwords from the text file is removed only when it finds a correct password.

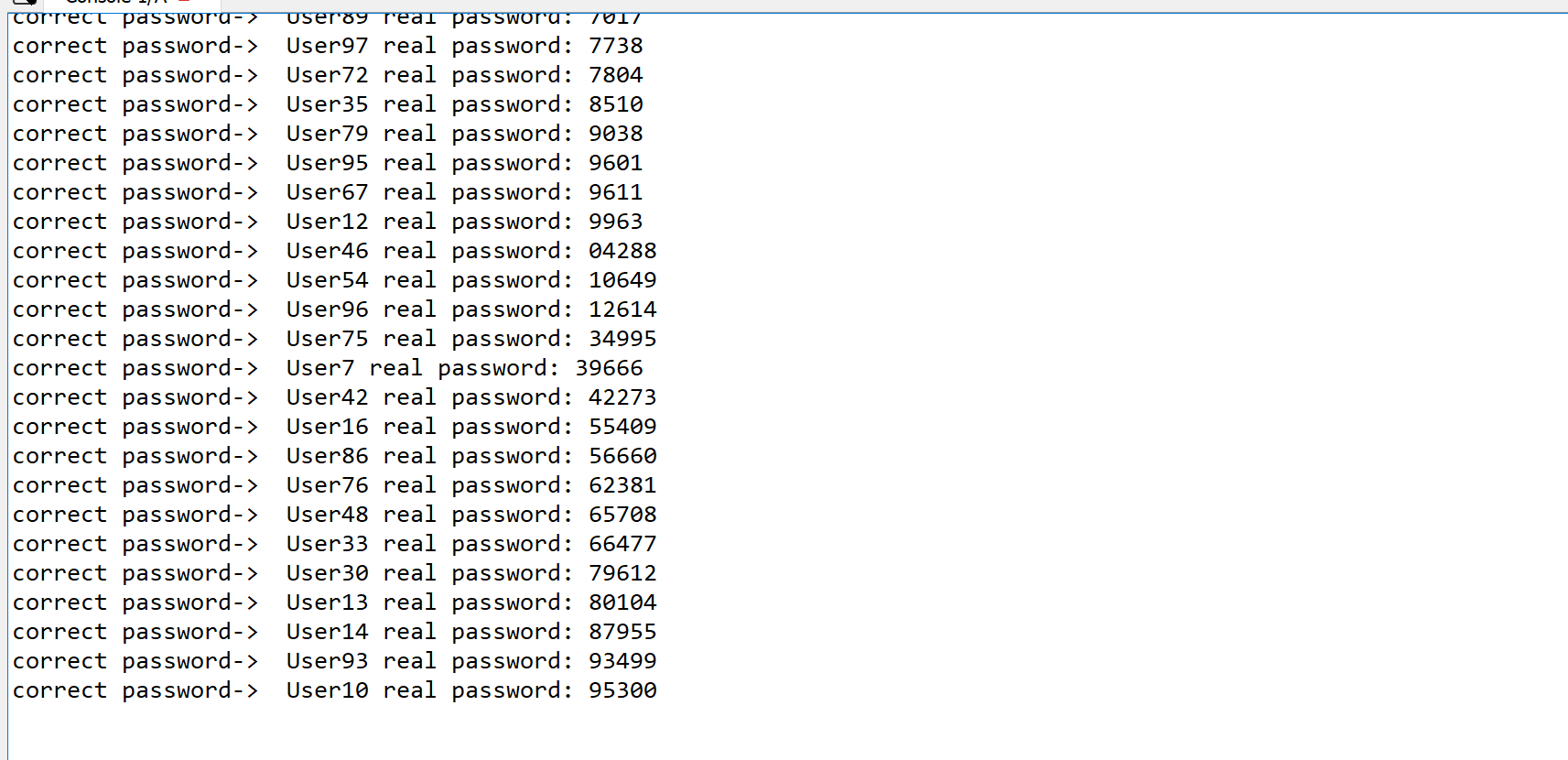
**Conclusion**: Learned how to work with recursion better, found out how there are many different efficient and inefficient ways to make a program run. Made various ways how the code could be programed to output the correct answer but in different ways.

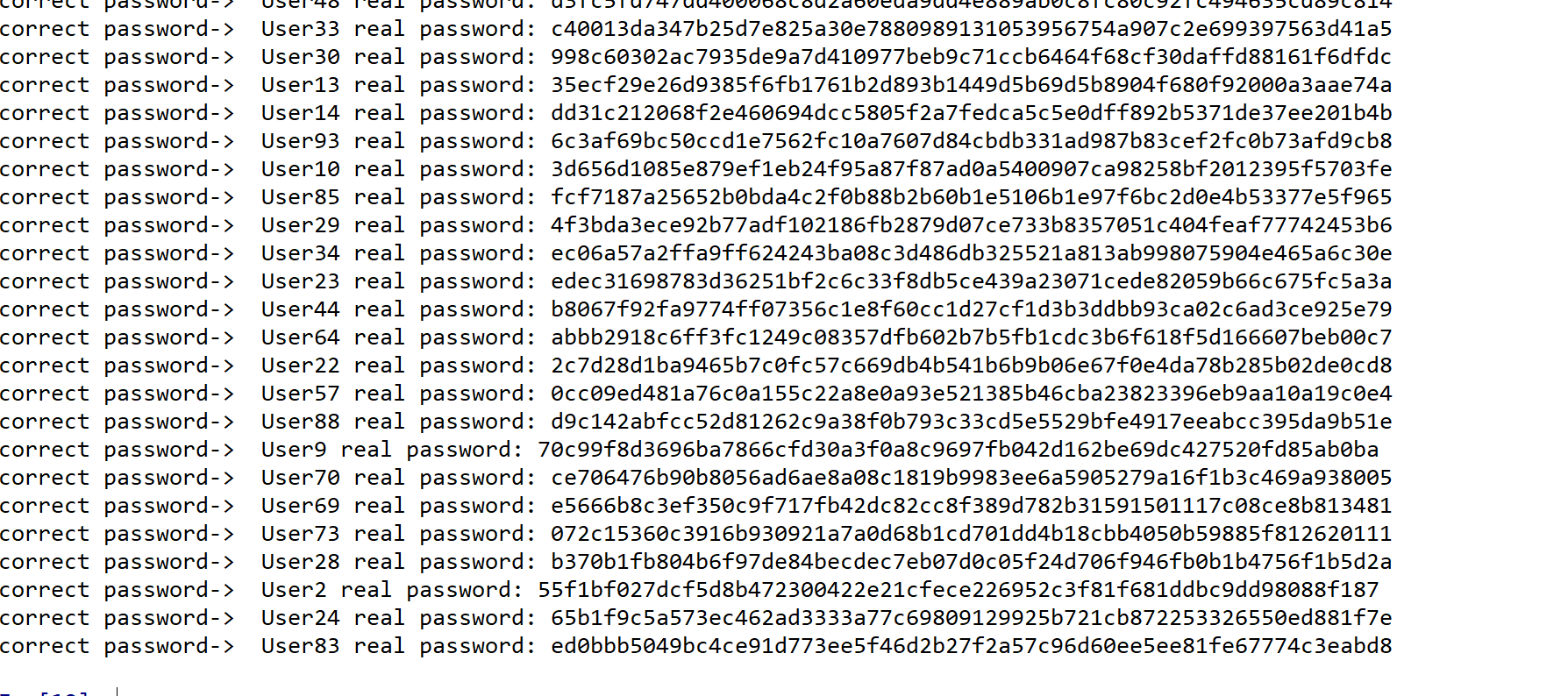
Academic dishonesty includes but is not limited to cheating, plagiarism and collusion. Cheating may involve copying from or providing information to another student, possessing unauthorized materials during a test, or falsifying data (for example program outputs) in laboratory reports. Plagiarism occurs when someone represents the work or ideas of another person as his/her own. Collusion involves collaborating with another person to commit an academically dishonest act.

Professors are required to - and will - report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.

**Results**:







**APPENDIX:**

'''

Author: Alexis Navarro

CS2302 -Lab 1 Option B

MW 1:30-2:50 PM

Professor: Diego Aguirre

Purpose: make a code that uses recursion to find the password of 99 users by using brute force

and learning recursion for myself.

'''

import hashlib

'''

One of my work in progress methods to create the password, but wouldn't concatnate my salt to the empty string. WOULD LEAVE X EMPTY ALWAYS.

def random\_password(x,size,salt\_val, hash\_val):

#print(x+salt\_val)

#print(size)

#print('length',len(x))

a=int(x)

if a==9:

return a

#x+=salt\_val

#print(A)

if check\_password(x,hash\_val) == True:

return x

for i in range(10):

#print(x)

a+=i

#print(int(x)+i)

print(a)

random\_password(a,size,salt\_val,hash\_val)

#random\_password(x+str(i),size,salt\_val,hash\_val)

'''

#Given Method to hash the passwords

def hash\_with\_sha256(str):

hash\_object = hashlib.sha256(str.encode('utf-8'))

hex\_dig = hash\_object.hexdigest()

return hex\_dig

#working method to make the new password

def random\_password(current\_password, str\_size, number\_limit):

if(str\_size == 0): # once the values of iterations is 0 then a password is made and checks if that password is correct

#print (current)

check\_password(current\_password) #checks if the password is correct

elif(str\_size > 0):

for i in range(number\_limit):

random\_password(current\_password+str(i),str\_size-1,number\_limit)# here I concatenate the empty string to the value i has in the for loop, I decrease the size of my string minus 1 after each recursive call

#to be able to get to my base case and because it means that a NEW password has been created.

#checks the password and followed the instructions that were taught in class on how to do it.

def check\_password(gen\_password):

for i in range(len(file\_info)):

x=hash\_with\_sha256(gen\_password + file\_info[i][1]) #try to concatenate like this however nothing happened

#if (hash\_with\_sha256(gen\_password + file\_info[i][1])== file\_info[i][2]): # if you want the results clean and only with the digits

if (x == file\_info[i][2]): # comment this if you want the cleaner results.

#complete\_password.append(file\_info[i][0] + ' real password: ' + gen\_password +" ")

complete\_password.append(file\_info[i][0] + ' real password: ' + x+" ") # prints the complete extended password with hash and salt values

#print("correct password-> ",complete\_password) # prints the correct passwords

del file\_info[i] # used del instead of remove cause it would output an error

return

#------------------------------MAIN---------------------------------------------------------------------------------------------------------------

def main():

# hex\_dig = hash\_with\_sha256('This is how you hash a string with sha256')

# print(hex\_dig)

#with open('password\_file.txt') as password\_file:

#for line in password\_file:

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THIS IS THE WORKING FUNCTION TO MAKE THE PASSWORD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

#create variables to hold my data just like how it was taught in class

global file\_info

global complete\_password

file\_info=[] # will be used to hold to info in the text file

complete\_password=[]

#while loop makes the password creation faster than using a for loop

file = open('password\_file.txt', 'r') # Reading from file

line = file.readline()

while line != '':

line = line.strip() # REMOVES THE WHITE SPACE FROM THE TEXT FILE

parts=line.split(',')

file\_info.append(parts)

line = file.readline()

#print(file\_info)

#nested for loop to make the random password and then checking with each use faster

for i in range(3, 7):

random\_password('', i, 10) #pass an empty string, the index, then 10 is for the digits that will be used 0-9

for j in range(len(complete\_password)): # prints the working/true generated passwords might be faster depending how its set up.

print("correct password-> ",complete\_password[j]) #different way to view how it prints

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

#gen\_pass=random\_password('00',3,parts[1],parts[2])#original and first attempt to make my program make a random password

# print("new password: ",gen\_pass)

main()