CS 2302

Lab 1.B Report

Author: David Ayala

ID: 80617838

Instructor: Diego Aguirre

TA: Gerardo Barraza

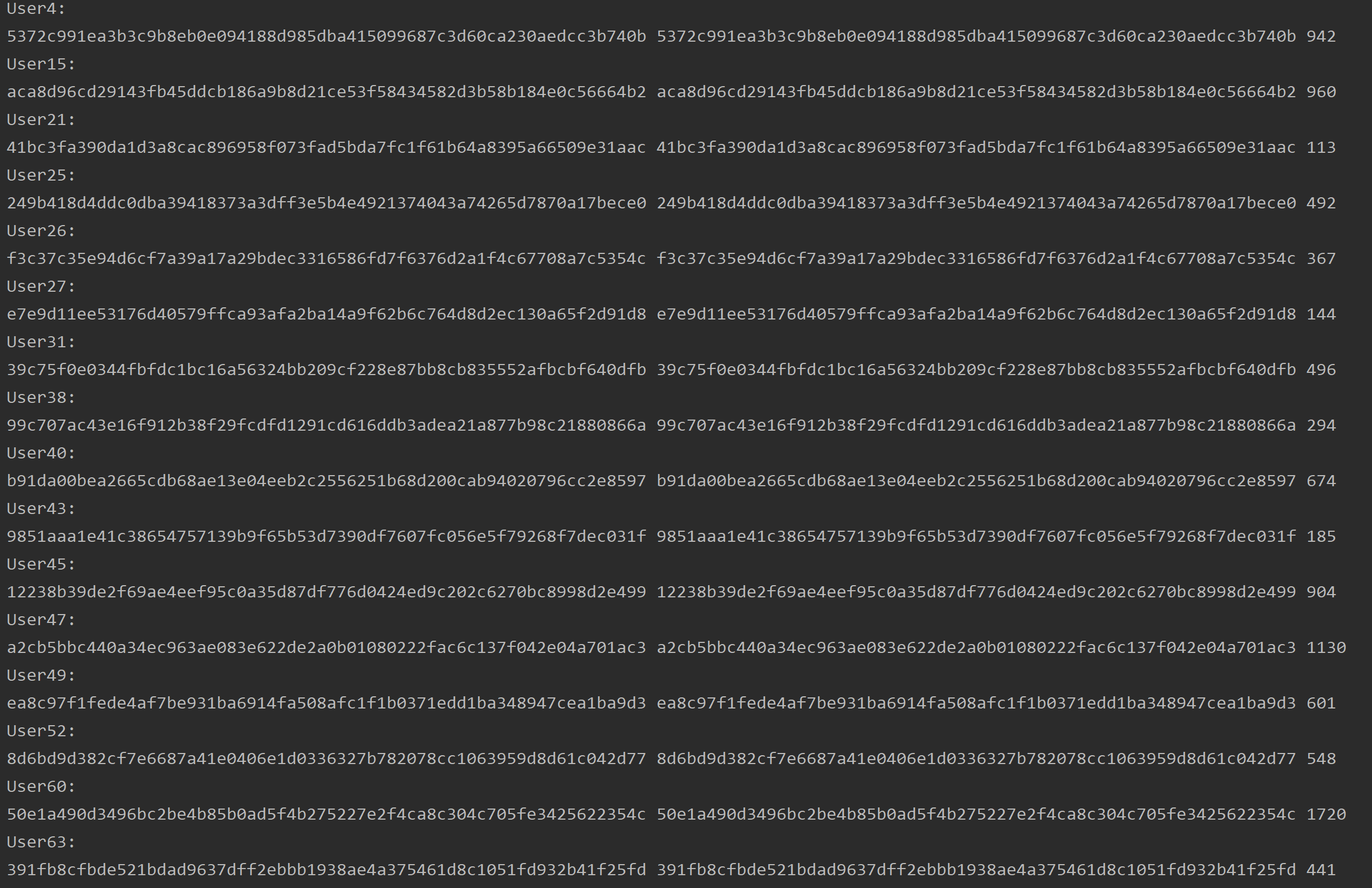
**Introduction**

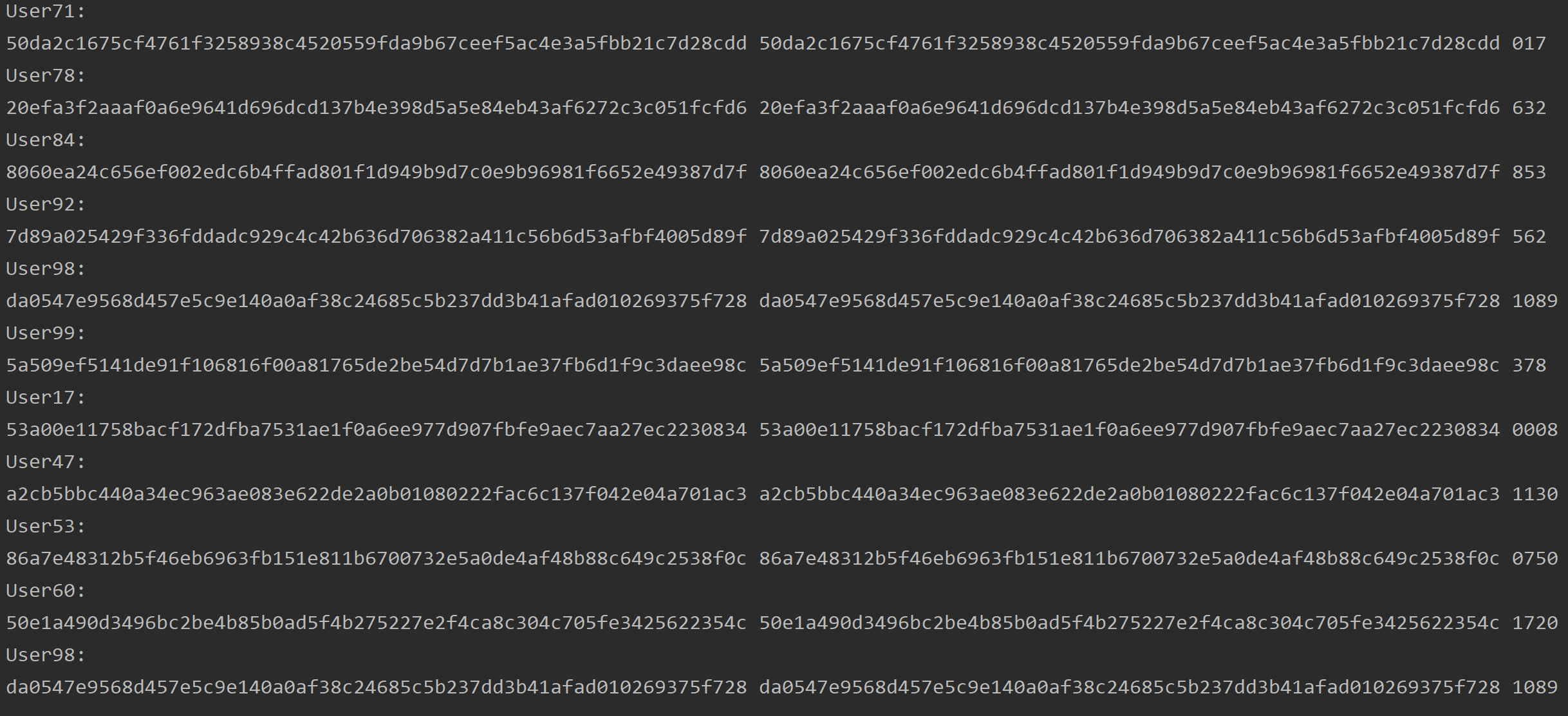
The purpose of this lab was to take a file that will have a set of users, hashed passwords, salt values. Once you read the file you are supposed to find the passwords for each user using the brute force method by recursion. Once you find the password you must then call the hash function that was given to us to use, so that you hash the password and check if it matches the hashed password in the file.

**Proposed Solution & Design Implementation**

So I knew that I would need to read the file and then split each line into user, original hashed password and salt value which are each lists. Once I have that done then I proceeded to generate password by using two for loops that are called recursively (erzeugenPasswort). Once the passwords are made then I had to save them to a list (bestätigenPasswort). I then had to check if the different passwords that were generated are the same as the original hashed passwords using (vergleichenHash), meaning I had to call the hash function (hash\_with\_sha256) to hash the found passwords. Once they are checked and a match is found then I then saved them to a new file named (PasswortsCracked).

**Experimental Results**





**Conclusion**

This lab allowed me to remember how to read a file, split each line by comma. But it helped me get better at recursion, having me find each password recursively. This lab also had me learn how our passwords are stored and let me know what a Salt value is. I also found that this lab really helps you learn how python works vs Java due to what I used to find each password, meaning I learned how to make method(functions), for loops, lists, variables, etc..

**Appendix**

# Assignment:Lab #1, Instructor: Diego Aguirre  
# Teaching Assistant: ????, Date of last Modification: 9/10/2019  
# Purpose of program:Use recursion to generate all possible passwords  
# and check to see if once hashed, that it matches the given hash.  
import hashlib  
  
# I put some German in this lab, since I'm currently taking it and  
# I'm trying to get better at it.  
  
# i is pronounced ee in German.  
# j is pronounced yott in German.  
# password is Passwort in German.  
# Salt Value is Salzwert in German.  
# line is Linie in German.  
# Produce is erzeugen in German.  
# compare is vergleichen in German.  
# file is Datei in German.  
# found is Gefunden in German.  
# Characters is Schriftzeichen(for computing) in German.  
# Confirm is bestätigen in German.  
  
originalHashedPasswort = []  
userID = []  
Salzwert = []  
  
# code that was given to use so that we can hash.  
def hash\_with\_sha256(str):  
 hash\_object = hashlib.sha256(str.encode("utf-8"))  
 hex\_dig = hash\_object.hexdigest()  
 return hex\_dig  
  
# saves passwords  
def bestätigenPasswort(userID, minimumSchriftzeichen, maximumSchriftzeichen, Salzwert, originalHashedPasswort):  
 Passworts = []  
 if minimumSchriftzeichen < maximumSchriftzeichen:  
 Passwort = erzeugenPasswort(userID, minimumSchriftzeichen, Salzwert, originalHashedPasswort)  
 Passworts.append(Passwort)  
 return Passworts + bestätigenPasswort(userID, minimumSchriftzeichen+1, maximumSchriftzeichen, Salzwert, originalHashedPasswort)  
 return []  
  
# produces passwords that will be return to and used in bestätigenPasswort  
def erzeugenPasswort(userID, minimumSchriftzeichen, Salzwert, originalHashedPasswort):  
 for ee in range(len(originalHashedPasswort)):  
 for yott in range((len(originalHashedPasswort))\*20):  
 yott = str(yott).format(yott).zfill(minimumSchriftzeichen)  
 Gefunden = vergleichenHash(yott, Salzwert[ee], originalHashedPasswort[ee], userID[ee])  
 return Gefunden  
  
# checks to see if the hashed password is the same as the one that was given.  
def vergleichenHash(Passwort, Salzwert, originalHashedPasswort, userID):  
 tempStr = Passwort + Salzwert  
 tempHash = hash\_with\_sha256(tempStr)  
  
 if tempHash == originalHashedPasswort:  
 with open("PasswortsCracked.txt", "a+") as List:  
 List.write(userID + " " + Passwort+ " ")  
 print(userID + ":")  
 print(originalHashedPasswort, tempHash, Passwort)  
 return Passwort  
 return  
  
try:  
 Datei = open("password\_file.txt", "r")  
except FileNotFoundError:  
 print("File not located, please check again.")  
  
for currentLinie in Datei:  
 Salzwert.append(currentLinie.rstrip().split(",")[1])  
 userID.append(currentLinie.rstrip().split(",")[0])  
 originalHashedPasswort.append(currentLinie.rstrip().split(",")[2])  
  
bestätigenPasswort(userID, 3, 7, Salzwert, originalHashedPasswort)