**CS 2302 - Data Structures**

**Fall 2019**

**Lab 1 - Option B (Password Cracking)**

**Overview**

A cyber-security company called *UltraHackz* is interested in finding talented computer science students. To do so, they post a challenge you find very interesting. They upload a *txt* file that contains 100 records containing information about 100 system accounts. Each record has a username, a salt value, and a *hashed* password.



Source: https://www.cio.com/article/3019670/security/cybersecurity-needs-transformational-leadership-from-women.html

Each record is stored as follows:

<username>,<salt value>,<hashed password>

That is, there is one line per record in the file, where the three values are separated by commas. Your job is to find the *real* password associated with each of the accounts. They tell you that all passwords contain only numbers (0-9), and that each password is at least 3 characters long, and at most 7 characters long. Your task is to implement a recursive method to generate all possible passwords (brute force). To make it interesting, *UltraHackz* posted the following rules:

* The method MUST be recursive
* Numbers 3 and 7 are not be hard-coded inside the method that generates the passwords; they should be parameters of the method
* You can have *at most* two nested loops inside this method (less than two is perfectly fine as well).

Every time you generate a string *s*, you need to check if *s* is the password of any of the usernames in the file. To do so, concatenate *s* with a user’s salt value, and apply the *hashlib.sha256* method to the resulting string. If the output generated by *sha256* matches the hashed string for that account, string *s* is the *real* account’s password! You accept *UltraHackz* challenge as you think this is a perfect opportunity to show off your **recursion** skills!

**Requirements**

You need to install the following on your machine (if you haven’t already):

* [Anaconda (Python 3.7](https://www.anaconda.com/download/))

**What you need to do**

**Due Friday, September 6, 2018**

Download the starter code (from Blackboard), implement your solution, and implement multiple tests (in your comments and report, you must justify your selection of tests). When you are done, upload the code to GitHub and send the link to the TA and instructor through Blackboard.

**Extra Credit**

Implement everything without using any for-loops!

**Rubric**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Proficient** | **Satisfactory** | **Unsatisfactory** |
| **Correctness** | The code compiles, runs, and solves the problem. | The code compiles, runs, but does not solve the problem (partial implementation). | The code does not compile/run, or little progress was made. |
| **Space and Time complexity** | Appropriate for the problem. | Can be greatly improved. | Space and time complexity not analyzed |
| **Problem Decomposition** | Operations are broken down into loosely coupled, highly cohesive methods | Operations are broken down into methods, but they are not loosely coupled/highly cohesive | Most of the logic is inside a couple of big methods |
| **Style** | Variables and methods have meaningful/appropriate names | Only a subset of the variables and methods have meaningful/appropriate names | Few or none of the variables and methods have meaningful/appropriate names |
| **Robustness** | Program handles erroneous or unexpected input gracefully | Program handles some erroneous or unexpected input gracefully | Program does not handle erroneous or unexpected input gracefully |
| **Documentation** | Non-obvious code segments are well documented | Some non-obvious code segments are documented | Few or none non-obvious segments are documented |
| **Report** | Covers all required material in a concise and clear way with proper grammar and spelling. | Covers a subset of the required material in a concise and clear way with proper grammar and spelling. | Does not cover enough material and/or the material is not presented in a concise and clear way with proper grammar and spelling. |