Password Salt System Implementation and Brutal Force Cracker

1. Implementation of the Password Salt System

In this section, students are required to implement a password salt verification system. With the given UID and Hash files, students need to implement the verification system, such that the given example of the password and salt can match with the hash value in the Hash.txt file. For example, the first UID is 001, the password is 0599, the salt associated with the first UID is 054. When applying the **MD5 Hash Function** with the **encode format as 'utf-8'** as shown in the figure below, the expected output should be 4a1d6f102cd95fac33853e4d72fe1dc5. It is worth to mention that, the concatenation between password and salt needs to be in the format of (password||salt). For example, with the aforementioned input, the concatenation result will be 0599054. 0 should not be omitted.

```
def computeMD5hash(my_string):
    m = hashlib.md5()
    m.update(my_string.encode('utf-8'))
    return m.hexdigest()'
```

Requirement for the designed system:

1) The designed verification system should be able to correctly verify the example shown above. When the input is correct, the system will output a String "The input password and salt matches the hash value in the database". Otherwise, the output should be "The input password and salt does not match the hash value in the database".

2. Implementation of the Cracker System

To reduce the complexity for cracking the password and salt, the passwords are randomly set in the range of [0000, 1000], while the salt is randomly set in the range of [000,100] for each UID. One easy idea to implement a cracker system is to brute-forcely try all possible combinations of password and salt for one UID. As the Hash.txt and UID.txt files are given, students are requested to implement a cracker system which could find the correct password and salt for a specific UID.

Requirement for the designed system:

1) For a specific UID, the cracker system can output the correct password and salt value. For example, when input the UID as 001, the output should be "password: 0599; salt: 054".

Demo and Report:

1) Each student is required to go to either TA or instructor to demo both systems. The TA or instructor will ask the students to run one or two specific UID(s) to check the corresponding password and salt.
2) The report should firstly describe how these two systems are designed; secondly, the report should also include the set of passwords and salts for **ten** different UIDs.
3) For undergraduate students, the verification and cracker systems can be designed separately. For graduate students, the cracker system should include the function of verification system.