The 2FA program allows for Prompting user for different requests: ‘1’ for creating new user, ‘2’ for login, ‘3’ for updating password and ‘4’ for deleting user account.

2. Creating Users: If we choose to create a user, the program should ask the user-id, password, salt and initial token IT. The salt will be same for a user account unless the user changes the password or deletes it and then create it again. If a user already exists, the program should display it and should exit (this should be considered as a failed attempt at creating user). If a user does not exist, then only user account should be created by 2FA method. When you create a user, you should update the /etc/shadow and /etc/passwd file for the user. A home directory for that user should be created and there should be an entry of home directory in the passwd file. Remember, no two users can have the same user-id in the passwd file. If a user is created, your code should print “User: <user-id of the user> created”.

3. Login: If a user enters “2” for login, the user should be prompted for the user-id first, and if the user does exist with that user-id then he/she should be prompted for the password, current token, and next token. At this point, the complete login process described in 2FA method should be executed. You code should handle possible errors, like what if the password is wrong or token is wrong, etc.

4. Password Update: Your login program should be able to update the password of a user. This deals with the situation when a user's password is compromised, or certain amount of time has elapsed since the password was created. Its code should ask the user for user-id and password, new password, new salt, current token, and next token. It should do necessary error handling. This function is a combination of creating and logging in a user account.

5. Deleting a user: It is simple as it sounds, the code should ask a user for its user-id, password and current token. If correct values are supplied for all of these, all entries and home directory for the deleted user-id should be cleaned.

6. Error Handling: Your code should be able to handle errors and should print a confirmation message after completing any of the above-mentioned functions. For example, if it just deleted a user U1, it should print U1 deleted or an interactive message to let the user know whether the task has successfully completed.

7. Evaluation: For evaluation, we will give you some test cases, and you should submit screenshots of the initial shadow file before each test case and the final shadow file after finishing all test cases. Do not alter shadow or any other files when executing test cases. As one test case can be dependent on other, if you delete some entries after test case 1, this may lead to a different output than the expected for test case 2.