Password Keeper

CCPROG2 Machine Problem

Create the application Password Keeper in C. Password Keeper is an application to store users' passwords in a password protected encrypted file.

When Password Keeper is run, a menu is presented where the user can either Create a New Account or Login.

Create New Account

To create a new account, a user must create a unique username and password. If the username is unique, the user then inputs a filename, and if it is also unique, the file is created, and this is where that user's other passwords for other accounts will be stored. The username, password and filename are added to a common, encrypted file containing all usernames, passwords and filenames of the users of your Password Keeper, however each user's passwords for other accounts are kept in their own personal files, separate from other users' files. A username (and the filename) is unique if its encrypted version is not on the file of usernames and passwords of the users of your Password Keeper.

The user also creates their value for a "key." You can give this value a more relatable name, such as secret number. The key is not stored in any file.

After a user creates an account, they are sent to the main menu.

Login

The user logs in by inputting their username and password, which will then be encrypted by the program, and if it matches a pair in the encrypted file containing usernames and passwords of your Password Keeper, it will ask the user to input the value of their "key." This key will be used to decrypt their personal file, as well as to encrypt added data.

If the inputted key is correct, the program can correctly display the account names, usernames and corresponding passwords of that user. (If the key is incorrect, what will be displayed will essentially be meaningless or garbage.)

A logged in user will also be able to choose from the following **main menu**:

1. Display all passwords
2. Add a password
3. Change a password
4. Delete a password
5. Change your password for Password Keeper
6. Logout and Exit Password Keeper

Display all Passwords

Displays all the account names, usernames and passwords of the currently logged in user decrypted from that user's personal file. If the inputted key was incorrect, it should be decrypted incorrectly and in effect, garbage text will be displayed instead of correct data.

Add a Password

A user can create a unique account name, and then its corresponding username and password. They are encrypted (using the key) and added to the user's personal file. The account name is unique if its keyed, encrypted version does not exist in the user's personal file.

Change a Password

A user can change a password by typing the account name, and after encrypting it, if it is in the personal file, the password can be changed by allowing the user to input the old password (that partners with the specified account name), and if, when encrypted, it matches the one on file, it can be changed by allowing the user to input the new password and encrypting and saving it to the file.

Delete a Password

A user can delete a password by typing the account name, and if it is in the personal file, the account name and its username and password are deleted.

Change your Password for Password Keeper

A user can change their password for Password Keeper by inputting their previous password and if it matches their encrypted account password on the file, then the user inputs their new password, which will then be encrypted and stored as replacement to their former password.

Logout and Exit Password Keeper

Ends the program.

Encryption

Encryption converts values such that they cannot be retrieved in their original form without the encryption algorithm. An encryption algorithm is described below, but you are allowed to use your own encryption method if you get the approval of your teacher first. To get the approval, the encryption algorithm must be more complicated than the one described below. 1:1 encryption or Caesar cipher will not be allowed (encryption where a symbol is always encrypted to another symbol but that symbol is always encrypted only to that other symbol). The default encryption algorithm is discussed below.

Default Encryption Algorithm

Add the key to an integer value equivalent to the character. The sum will be stored on the file, and then will be added to an integer value equivalent to the next character. Store the new sum next to the previous sum in the file. Add this new sum again to the integer value of the next character, and store it beside the previous sum in the file, and keep on doing this a defined number of times depending on the key. After the defined number of times, reset the sum and start the process again.

For example, if the integer equivalent you are using for each letter is their sequence number in the English alphabet, and the key you are using is 24, and the number of times you keep adding the previous sum is (key % 10) [therefore 4 times] before resetting

to encrypt the word Google

G = (7 + 24) = 31 (7^{th} letter of English alphabet + the key)

O = (15 + 31) = 46 (15^{th} letter of English alphabet + previous sum)

O = (15 + 46) = 61 (15^{th} letter of English alphabet + previous sum)

G = (7 + 61) = 68 (7^{th} letter of English alphabet + previous sum)

L = (12 + 24) = 36 (12^{th} letter of English alphabet + the key, the process was restarted after 4 times because 24 % 10 = 4)

E = (5 + 36) = 41 (5^{th} letter of English alphabet + previous sum)

Thus, using the algorithm, to store the word GOOGLE, what will be stored will be

31 46 61 68 36 41

To decrypt, reverse the process.

This is still a very easy encryption algorithm to decrypt, so if you would like to use a more complicated one, you may propose it. Ones that can't easily be hacked by rainbow tables are good.

Two Kinds of Keys

As discussed earlier, each user chooses their own key when they create their account. Keys are not stored in any files. If a key is wrong, the user just cannot read their data since it is garbage.

For the common encrypted file containing all the usernames and passwords of the users of your Password Keeper, you, as programmer, can choose its key, as the second key.

Adding Salt

If you are using the default encryption algorithm, or an algorithm where it makes sense to use it, you can add another value called "salt" to the key or algorithm. Instead of a random value, for Password Keeper, the salt can be computed from data that is already available, for example the salt could be the length of the username. So in the example above, the user-chosen key is 24, and the username is CCPROG2, the length of the username (7) will be added to the user's chosen key (24) so the encrypted value will be 31 plus the integer value of the character being encrypted. You can come up with a better method to generate the salt, but the value of the key should not be constant for all users. You can also use the salt in other ways, instead of simply just adding it to the key.

Other Notes

This is an individual project.

Your solution must use structures.

If you use an existing encryption method or code, for example a predefined function, it must only be on top of or in addition to another encryption method, such as the default encryption method, that you code yourself.

If your encryption algorithm is 1:1 or Caesar cipher or is worse than what is discussed here, at best, your grade will only be barely passing, assuming everything else is correct.

What is discussed here is the minimum requirement. Maximum bonus of 10 points can be given for additional, real security measures, such as the use of a much better encryption algorithm, or a more advantageous use of the salt. You may propose additional security measures to your teacher. You should not discuss these additional, security measures to anyone else other than your teacher.

Note that you should not actually use this software to store your passwords unless your encryption and security measures are enough.