Programming Assignment #4

Password Manager

CS 2308.255 + CS5301 Spring 2020
Instructor: Jill Seaman

Due: Monday, 3/23/2020: upload electronic copy by 11:59pm!

Problem: Write a C++ program that will simulate a Change Password Utility.

Your program will contain:

- One class, PasswordManager, that will manage a single password for a given username.
- A main function that will allow the user to change their password.

Your program should consist of the following files:

PasswordManager.h
PasswordManager.cpp
PasswordDriver.cpp (containing the main function)

You should also have a **makefile** that can be used to build the executable program.

PasswordManager Class:

The PasswordManager class should have two member variables, which will store the username and the encrypted password (both strings). Do **not** store the password (in the class or in a text file) unencrypted!

The PasswordManager class should have the following two internal member functions (not accessible outside of the class):

encrypt: this takes a password (a string) and returns the encrypted form of the password. Note: there is no decrypt function (there is no need to decrypt passwords). We will use the following VERY simple encryption algorithm (a Caesar Cipher):

For every character in the input string, add 45 to the ascii value of the character. The encrypted character's ascii value must stay in the range of printable, non-whitespace characters: 33 to 126. This can be enforced using this formula:

```
ascii value of encrypted char =
((ascii value of ch - 33) + 45) % 94 + 33
```

Store all the resulting chars in a string to be returned as the result of the function (hint: use string.push_back(char)).

meetsCriteria: this takes a string (a password) and returns true if it meets the following criteria:

- it has at least 15 characters, but not more than 127 characters.
- it contains at least one uppercase letter, one lowercase letter and one symbol (punctuation). However, '*' and '%' are not allowed

If the criteria are not met, it returns false.

The PasswordManager should have the following member functions that are accessible from outside of the class (from the driver):

setUsername: (a setter function) takes a string and stores it in the proper member variable.

getUsername: (a getter function) returns the value of the proper member variable.

setEncryptedPassword: (a setter function) takes a string (an already encrypted password) and stores it in the proper member variable.

getEncryptedPassword: (a getter function) returns the value of the encrypted password stored in the proper member variable.

setNewPassword: takes a string (a proposed, unencrypted, password). If it meets the criteria in meetsCriteria, it encrypts the password and stores it in the member variable and returns true. Otherwise returns false.

authenticate: takes a string (a password) and returns true if, once encrypted, it matches the encrypted string stored in the the member variable. Else returns false.

Input/Output:

The main function should create an array of 3 instances of the PasswordManager class.

Your main function will use a file "passwords.txt" to store the usernames and **encrypted** passwords in between executions of the program. When your main function starts, it should try to open the file "passwords.txt". If the file exists, you should assume it contains 3 users names with encrypted passwords, and the program should use these to set the usernames and encrypted passwords in the password manager array. Note: close the input file here! Do not open the file for output until after you have closed it for input. If the file doesn't exist, exit with an error message.

Your program should explain the criteria for new passwords (see meetsCriteria above) and then ask the user to enter their netID, old password, and new password:

```
Please enter your username: j_s108
Please enter your old password: AAAAAbbbbb.....
Please enter your new password: HelloKitty!!!!!
```

Your program should give one of the following responses:
```
    NetID is invalid, password not changed.
    Old password is incorrect.
    New Password does not meet criteria.
    Password has been changed for netID: j_s108
```

If any of the input data is invalid or causes an error message, do NOT ask the user to re-enter the data.

After outputting one of the above responses to the screen, the program should output the three **encrypted** passwords to the file "passwords.txt", one per line (overwriting anything that was previously in the file), then exit.

You should create a starter passwords.txt file with sample usernames and passwords: (note that **`nnnnn11111[[[[[`** is the encrypted versions of **AAAAAbbbbb.....).**

```
j_s108  nnnnn11111[[[[[
fgh123  nnnnn11111[[[[[
ert789  nnnnn11111[[[[[
```

NOTES:

- This program must be done in a **Linux or Unix** environment, using a command line compiler like g++. Do not use codeblocks, eclipse, or Xcode!
- Usernames and passwords do not contain any whitespace. When a user enters a password, assume a whitespace character indicates the end of the password.
- Do NOT change the names of the functions! Use the exact same function names, and do not change the case (uppercase/lowercase). DO NOT change the input order of the three values.
- Create and use a **makefile** to compile the executable program. Modify the one from the TimeDemo (on the class website).
- Put the Class declaration in the header file, the implementation of the class member functions in PasswordManager.cpp and the main function in PasswordDriver.cpp. Put a header comment at the top of each file.
- ALL of the input and output must be done by the driver. **The password manager class should not do ANY input/output**, not to the screen OR the file!
- Useful functions: isupper(char), islower(char), ispunct(char), string.length()

Logistics:

Submit 4 files only: PasswordDriver.cpp PasswordManager.cpp PasswordManager.h and makefile

Zip them into a single zip file for submission using zip:

```
[...]$zip assign4_xxxxx.zip PasswordDriver.cpp
PasswordManager.cpp PasswordManager.h makefile
```

Submit an **electronic copy** using the Assignments tool on the Canvas website for this class (canvas.txstate.edu).

See the assignment policy on the course website (cs.txstate.edu/~js236/cs2308) for more details, including late deadlines and penalties.