Programming Assignment #4

Password Manager

CS 2308.003, Fall 2011
Instructor: Jill Seaman

Due: in class Thursday, 10/20/2011 (upload electronic copy by 10am)

Problem:

Write a C++ program that will manage passwords.

Your program will contain:
- A Class, PasswordManager, that will manage a single password.
- A main function, that will allow the user to test the PasswordManager class.

Your program should consist of the following files:
PasswordManager.h
PasswordManager.cpp
PasswordDriver.cpp

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

PasswordManager Class:

The PasswordManager class should have just one member variable: the encrypted password (a string).

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

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

> The XOR operator (^) takes two char arguments and returns a char.
> For every character in the input string, apply the XOR operator ^ with the character '2'. For example: `str[i] ^ '2';`

Store all the resulting chars in a string to be returned as the result of the function.

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

- it is at least 6 characters long
- it contains at least one uppercase letter
- it contains at least one lowercase letter
- it contains at least one digit

Otherwise it returns false.

The PasswordManager should have the following member functions that are accessible outside of the class:

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

getEncryptedPassword: returns the value of the encrypted password stored in the member variable.

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

validatePassword: takes a string (a password) and returns true if, once encrypted, it matches the encrypted string stored in the the instance variable.

Input/Output:

The main function should contain an instance of the PasswordManager class. It is called "the password manager" below.

Your main function should first try to input an encrypted password from the file "password.txt". If it finds one, it should set the encrypted password in the password manager. If there is no file (or it does not contain a string) then set the password in the password manager to "ABCabc123".

Then use the following menu to prompt the user to test the implementation:

```
Password Utilities:
A. Change Password
B. Validate Password
C. Quit
Enter your choice:
```

The menu should be processed in a loop, so the user may continue testing the password operations.

The Change Password option should ask the user to enter a new password, and explain the criteria for a valid password. The main function should ask the password manager to verify and change the password. Output a message indicating whether or not the password was changed.

The Validate Password option should ask the user to input the password. Then the main function should ask the password manager to validate the password, and then output whether or not the password was valid (matching the one stored by the password manager).

When the user selects C to quit, the program should save the encrypted password in the file "password.txt" (overwriting anything that was previously in the file).

NOTES:

This program DOES need to be done in a Linux/Unix environment. Create and use a makefile to compile the executable program. Modify (but don't copy and paste) the one in Lecture7.pdf. I recommend calling the executable file "password".

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.

DO NOT change the names of the functions or files.

DO NOT try to hide the password (with ****) as the user types in the password.

Follow the rest of the style guidelines described here:
http://www.cs.txstate.edu/~js236/styleguidelines.txt
(there is a link to these on the course webpage as well).

Logistics:

Since there are multiple files for this assignment, we want you to combine them into one file before submitting them. You should use the zip utility from the Linux/Unix command line:

```
[. . .]$zip assign4_Axxxxxxxx.zip PasswordDriver.cpp
PasswordManager.cpp PasswordManager.h makefile
```

This combines the 4 files into one zip file, assign4_Axxxxxxxx.zip. Then you should submit only assign4_Axxxxxxxxx.zip.

Name your file **assign4_xxxxxxxxx.zip** where xxxxxxxxx is your 9 character TX state ID number, the one that is on your ID card. It should look something like this: A04123456. If yours is just six digits, then add "A00" to the front.

There are **two** steps to the turn-in process:

1. Submit an **electronic copy** using the following upload link:

   http://www.cs.txstate.edu/~js236/homework
   (There is a link directly to this page on the course website).

   Click on CS1428.004, and log in with your Net ID and follow the directions to upload your file.

2. Submit a **printout** of the file at the beginning of class on the day it is due. Please print your name on the front page, and staple if there is more than one page.