SCC110: Software Development Term 1. C Programming.

Problem Set 7: Passwords

The Goal
String processing is critical to many computer applications. For example, we check a string entered by a user against a string stored in the computer whenever we check a login name or password.

Passwords are essential to the security of computer systems. A system's security is only as strong as the weakest password chosen by the systems users. We need so called '*strong passwords*', consisting of a *mix of symbols and digits* so that they're hard to guess (making it hard for attackers to work through each symbol in terms and guess your password!).

Your goal today is to take on the role of a computer security expert (CSI 110 ;-)) and to write a program that ensures the strength of a password entered by the user.

1.1 Password Exercise

Incrementally develop a solution that:

- Prompts the user to enter their password (*this should be synthetic, and not be a real password naturally!*), and checks it against a password hard coded into the program for validity. You should let the user know if they entered the password correctly or not.
- Extend your program so that it allows the user to change their password if they can enter the previous password correctly. The user should have to enter the *new password twice* to ensure that they have typed it correctly. *Note: in a real password program you would `hide' the user's keystokes. That is not necessary for this exercise.*
- Ensure that the password *is longer than 8 characters* to be valid.
- (*optional*) Validate that the password contains at least one number and at least one symbol to be valid.

You should think carefully about your algorithm and approach on paper first. Don't forget good coding practices, including choice of meaningful variable names, comments and indentation.

Hacker edition
- Extend your password program to allow additional validation (passwords must have at least two upper, two lower case letters, one or more numbers, one or more symbols). *See the manual page for `ctype.h' for assistance with this*.
- Output a `strength calculator' value that increases with the difficulty of guessing the password.
- (optional) Create a suggested password function, so the system provides a strong password to the user. Note that you are free to explore different designs for this, but note that memorability by the user is often in tension with security!

As before, you should be looking to make your code a) great to read, and b) solve the problem elegantly. Be sure to ask our opinion on your code in the labs.

As part of today's problem set you will get more familiar with how to do string processing in C. There are lots of resources to draw upon:

Resources
- There are nice examples in the book Section 2.3 and 3.5;

- we also recommend online tutorials such as https://beginnersbook.com/2014/01/c-strings-string-functions/
- Please also revisit this week's lecture covering arrays and strings.

You will need to use functions from `#include <string.h>` and possibly also `#include <ctype.h>`