CS 132 – PROGRAMMING ASSIGNMENT #2

OVERVIEW

This activity is designed to have you work with strings as a data type. This includes both a traditional CStrings as well as the more modern string ADT.

INSTRUCTIONS

In this program you will create an interface for a program that requires a password to enter it. For security you will implement a (bad) form of encryption to keep people from looking at a list of password.

You will open up the `PasswordEncrypter.cpp` file as your starting point. It contains the framework of a program, but with two methods that you will need to add to / finish to make the program work.

Note that you may/probably will want to create some additional helper functions to assist you in your code.

Your primary goal will be to implement two functions: `addToFile(string fname)` and `checkFile(string fname)`.

ADDTOFILE

The `addToFile(string fname)` method will ask the user for a word/phrase that they want to add to the allowed list of acceptable passwords. It will check the phrase for acceptable use based upon the rules described below, and if acceptable will then encrypt the word and then add it to the password file.

If the word/phrase is not acceptable, then the method will not add it to the password file, and it will return the user to the main program.

This method will require a string parameter that is the name of the file in question, and is the name of the file that should be opened. In the provided file, it is given for you.

CHECKFILE

The `checkFile(string fname)` method will do the opposite. It will ask the user for a password and then convert the possible password into its encrypted form. It will then compare it to every possible password in the file to see if it is one of the possible password. If it is, it will respond in the affirmative, and if not it will notify the user that the password was unacceptable.

PASSWORD RULES

In order for a phrase to be a valid password, it must follow the following rules. You should check these before encrypting the data.

- It must be at least 10 characters long.
- It must contain at least 8 letters
- It must contain at least 4 numbers
- It must contain at least 1 valid symbol
    - The ONLY valid symbols are
        - .
            - The period
        - ?
            - The Question Mark
        - ,
            - The Comma
        - ' '
            - The Space
- For reasons that will be clear in the next section, we do not care about capitalization.

DATA ENCRYPTION

For this assignment, we are going to use a **horrible** version of encryption, but one that will allow us to practice.

After you have checked the password for acceptance, you will then translate the string into Morse code using the symbols.

- `'*'` for dot
- `'-'` for dash
  - *(The symbol between the zero and the equals sign NOT the underscore)*
  - *Be careful here, get the right symbol.*
- A space `' '` between letters.
- A slash `'/'` between words.

So, for example `Hello World 1234` would translate to

```
**** * *-** *-** --- / *-- --- *-* *-** -** / *---- **--- ***-- ****-
```

Note that Morse code does not care about capitalization, so you will want to convert your password into all caps or lower case to help you with your work.

BEFORE SUBMISSION

Before submitting your assignment, include a comment at the beginning of your program with some basic information and a description of the program in your own words. For example:

```
// Suzy Student, CS132, Autumn 2049, Section XX
// Programming Assignment #2, 06/07/49
//
// This program's behavior is ...
```

Make sure to comment appropriately and document all your major points. There is a large amount of provided code that is uncommented. **This is deliberate**. Part of this assignment in addition to your own code is to go back and document the provided code to tell other programmer what it is supposed to do.

PROGRAM NOTES AND HINTS

Pick one method and get it working first.

While eventually you will need to work with files, you may just want to print to the screen until it is working.

A sample file has been provided to get you started. The sample passwords are `"Hello World 1234"` and `"password..5678"`.

- The final page of this assignment includes all the Letters and their Morse code versions. You may find it helpful to cut and paste it instead of typing everything in by hand.
- I highly recommend making a method that takes in a char, and returns a string with the morse code attached.

```
string charToMorse(char x);
```

- **If** you make the above method, it will have a large number of if statements, that is normal.
- In order to open a file and append a new line to it, check out :

```
outfile.open("filename", std::ios_base::app);
```

- You will probably want to use the `getline()` method to get entire lines from the password file.
- However when you use getline, you have the possibility of a line have unnecessary spaces at the beginning or end of a line. In computer science it is common to remove these extraneous spaces by using a `trim()` command after getting the line, but before manipulating the line.
    - Unfortunately C++ does not have a built in trim command. So here is one you can use if you so desire it:

```
void makeTrim(string &x)
{
   const char* ws = " \t\n\r\f\v";      //List of space Characters
   x.erase(0, x.find_first_not_of(ws)); //prefixing spaces
   x.erase(x.find_last_not_of(ws)+1);   //surfixing spaces
}
```

MORSE CODE CUT AND PASTE.

A	*-
B	-***
C	-*-*
D	-**
E	*
F	**-*
G	--*
H	****
I	**
J	*---
K	-*-
L	*-**
M	--
N	-*
O	---
P	*--*
Q	--*-
R	*-*
S	***
T	-
U	**-
V	***-
W	*--
X	-**-
Y	-*--
Z	--**
0	-----
1	*----
2	**---
3	***--
4	****-
5	*****
6	-****
7	--***
8	---**
9	----*
,	--**--
.	*-*-*-
?	**--**