CSCI 566

Grad Assignment 3
(10 points)

Overview

For this assignment, you are writing a program that verifies that user passwords meet the following criteria:

-The password must be at least 6 characters long

-The password must contain at least one uppercase letter

-The password must contain at least one lower case letter

-The password must contain at least one digit

Write a program that reads lines containing passwords from the keyboard. The program should determine if the password entered is valid or not. It will read multiple lines (one character at a time) reporting *on each line* until the user enters a line beginning with '!'

This program is not long or complex. But there are several new ideas and features to use. There are a lot of details, hints, and requirements in this assignment. So read it carefully - there is some new information in the assignment itself which is not in the lecture notes. Read it several times before you begin work on the program; make sure you clearly understand what is explained and required here.

When you think you are done, read the assignment again carefully and compare it to what is required to be sure you have followed instructions correctly. (Hint: always do this, for all assignments.)

Input

As described above, the program will process lines of text, one line one character at a time. To do this, the program will use a new input library function:

cin.get(ch); - this function (technically, a "method") reads a single character from standard input and *stores it into the char variable* passed as an argument. Up to now, we have said that a function cannot directly alter the value of any of its arguments. That was actually a lie. :-) We will soon learn how it can be done. But for now, we will just use this feature.

cin >> ch; does not work well in this program, because it skips over newline chars and spaces.

Processing

So the program will read characters - one at a time - from the keyboard. Since the program will quit when the first char in a line is a '!' - but will output a line whenever a newline character is encountered, the following logic can be used (NOTE: *ch* is a char variable):

```
char ch;

//Get the first character

cin.get( ch );

//While there is data to be processed
```

```
while( ch is not a '!' )
  {
  //While it is not the end of a line

  while( ch is not a newline character )    //'\n' or the
ASCII value of 10
    {
    //process the ch character. This should include echoing
the password incrementing the appropriate individual line
counters

    //Get the next character
    }

  //Display the valid or invalid ID message

  //Reset the alphabetic and digit counters for an
individual line

  //Get the next character
  }
```

The above logic is a loop nested inside of a loop. The outer loop processes each line, while the inner loop processes each character in a line. There is a behavior of *cin.get()* that needs to be understood: it will not "wake up" until a newline <enter> is typed. In other words, as the characters are typed, they will appear on the screen, but the actual processing of the character will not begin until the line is terminated by pressing <enter>.

The program should "echo" - that is, repeat - the inputted line, character by character, on the screen. As the program reads each character, it will display it on the screen via *cout*. See the Sample Output below. Understand that you will type a line, and see the characters you type, one at a time as you type.

Hint: write the program first without any counters: just make sure you can input and "echo" lines as described above, and also be able to terminate the program with a '!'

As mentioned earlier, for each character that is read (with the exception of newlines and the '!' that quits the program), the program will increment one or more of several counters (using function calls to determine whether to increment each one or not).

Finally, after all the counters have been incremented for all of the characters on the current line, display a message (Valid or Invalid) and then reset the counters for an individual line to 0 to prepare to process the next line.

The Functions

Write and use the following 3 functions in the program.

int isDigit(char ch)

This function will determine if the character ch is a digit '0' through '9'. If the character is a digit, return 1. Otherwise, return 0.

DO NOT use the isdigit function from the <cctype> library. You are writing your own version of the function.

int isUpper(char ch)

This function will determine if the character ch is an uppercase 'A' through 'Z' character. If the character is uppercase, return 1. Otherwise, return 0.

DO NOT use the isupper function from the <cctype> library. You are writing your own version of the function.

int isLower(char ch)

This function will determine if the character ch is a lowercase 'a' through 'z' character. If the character is lowercase, return 1. Otherwise, return 0.

DO NOT use the islower function from the <cctype> library. You are writing your own version of the function.

Processing Requirements

1. As always, complete program documentation is required for this program, that includes documentation boxes for ***EACH*** function. This will be the last reminder in the program write-ups.

2. Hand in a copy of the source code (CPP file) using Blackboard.

Sample Output

```
Enter password, ! at the beginning of the line to end
bC1234
bC1234
invalid password, missing uppercase letter

Enter password, ! at the beginning of the line to end
ABC1234
ABC1234
invalid password, missing lowercase letter

Enter password, ! at the beginning of the line to end
aBCdef
aBCdef
invalid password, missing digit

Enter password, ! at the beginning of the line to end
123ZzYx
123ZzYx
valid password

Enter password, ! at the beginning of the line to end
```

```
aBc12
aBc12
invalid password, too few characters

Enter password, ! at the beginning of the line to end
abcdef
abcdef
invalid password, missing uppercase letter
invalid password, missing digit

Enter password, ! at the beginning of the line to end
!

Number of valid passwords:    1
Number of invalid passwords:  5
```