COMP 53: Arrays and Vectors Lab, part 3

Instructions: In this lab, we are going to review library functions for strings and characters, along with two-dimensional arrays.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your source code files list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **38 points** in aggregate. The details are given in the following.

1 `main.cpp`

In `main.cpp` do the following step by step:

1. Include `iostream`, `cstring`, and `cctype` libraries.
2. Globally define character array `title1[]` of length 30, and initialize it with
   `"Data Structures In C++"` ***(2 points)***.
3. Globally define character array `title2[]` of length 30, and initialize it with
   `"Data  Structures  In C++"`, where there are three spaces between `Data` and `Structures`. Moreover, there are two tabs and one space between `Structures` and `In` ***(2 points)***.
4. Globally define two dimensional array of characters `password[][]` with four rows and 30 columns. Initialize it with `"monkey"`, `"MonnkeeY"`, `"M8nnkeeY"`, and `"M8nnkeeY!"`. For initialization use braces ***(3 points)***.
5. Define the following functions.
   (a) Define function `void splittedPrint(...)` that receives an array of characters. It prints the words which are separated by white-spaces (tabs and spaces) in separate lines ***(4 points)***. *Hint*: Use proper functions from `cctype` library to identify white-spaces. Check the sample output below for examples.
   (b) Define function `void stringFlipCase(...)` that receives an array of characters. It prints the input array of characters where the uppercase alphabetic characters are flipped to lowercase, and lowercase alphabetic characters are flipped to uppercase ***(4 points)***. *Hint*: Use proper functions from `cctype` library to identify lowercase/uppercase characters and to flip them. Check the sample output below for examples.
   (c) Define function `void stringTrimNonAlphanumeric(...)` that receives an array of characters. It prints the input array of characters where all non-alphanumeric characters are removed ***(3 points)***. *Hint*: Use proper functions from `cctype` library to identify alphanumeric characters. Check the sample output below for example.
   (d) Define function `bool isValidPassword(...)` that receives an array of characters (that represents a potential valid password). This function returns `true` if the input is a valid password. Otherwise, it returns `false`. Here are the rules that define a valid password:
      (i) Passwords must at least be of length 8 ***(2 points)***. *Hint*: Use proper function from `cstring` library to study string length.
      (ii) Passwords must at least have one lowercase alphabetic character ***(2 points)***.
      (iii) Passwords must at least have one uppercase alphabetic character ***(2 points)***.
      (iv) Passwords must at least have one digit ***(2 points)***. *Hint*: Use proper function from `cctype` library to study whether a character is digit.

(v) Passwords must at least have one special character ***(2 points)***. Special characters are `!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

Hint: Use proper function from `cctype` library to study whether a character is a special one.

Your function must report all of the reasons for rejecting the input. It must also report if an input is acceptable ***(3 points)***. Check the sample output below for examples.

In `main()` function do the following step by step, using the functions defined above:

(I) Print out the splitted `title1` using the function defined above ***(1 points)***.

(II) Print out the splitted `title2` using the function defined above ***(1 points)***.

(III) Print out `title1`, where the cases are flipped (using the function defined above) ***(1 points)***.

(IV) Print out `title1`, where non-alphanumeric characters are removed (using the function defined above) ***(1 points)***.

(V) Print out `title2`, where non-alphanumeric characters are removed (using the function defined above) ***(1 points)***.

(VI) Within a `for` loop check the validity of the passwords given in the array `password[][]` (using the function defined above) ***(2 points)***.

The output of the program may look like the following:

```
Splitting title1:
Data
Structures
In
C++

Splitting title2:
Data
Structures
In
C++

Flipping the case in title1: dATA sTRUCTURES iN c++
Trimming non-alphanumeric characters in title1: DataStructuresInC
Trimming non-alphanumeric characters in title2: DataStructuresInC

monkey Not Accepted: Passwords must be at least 8 characters long.
monkey Not Accepted: Passwords must at least include one uppercase alphabetic character
monkey Not Accepted: Passwords must at least include one digit.
monkey Not Accepted: Passwords must at least include one special character:
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

MonnkeeY Not Accepted: Passwords must at least include one digit.
MonnkeeY Not Accepted: Passwords must at least include one special character:
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

M8nnkeeY Not Accepted: Passwords must at least include one special character:
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

M8nnkeeY! Accepted.
```