Lab 10 [String Utilities]

In this workshop, we will continue our development of a text based game "Code Quest!". This workshop focuses on enhancing the game by adding strings for items and players.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will be able to

- work with null-terminated character strings
- write code that accepts string input
- write code that displays string output

PART 1:

SPECIFICATIONS

This program will require the following two functions:

`int Validate_Alphanumberic(const char str[]):` This function accepts an address of a C-style string and returns an integer. This function verifies that a string only contains letters (a-z, upper and lower case), numbers and spaces. The function returns 1 if the string is valid, and 0 otherwise.

`void Get_Valid(char str[])`: This function takes an address of a C-style string (**`str`**) as input. This function prompts the user to input a string and stores it in **`str`**. The function validates the string with the function above (**`Validate_Alphanumberic`**). If the string is invalid, an error message is printed and the input buffer is cleared. The retrieval step is repeated until a valid string is entered.

Once the above two functions are implemented, write a program that allows simple tests of the above functions. This program first allocates an array of 5 strings (char arrays) with 10 chars each. This allocation is done by calling **`Get_Valid`** function.

Then, the program defines an integer array of 5 integers initializing them from 0 to 4.

Finally, the program outputs the list of integers and the strings associated with them.

The output of a typical run-through of your program should look like this (user input highlighted in green):

```
Please enter items names
Item 1: Abc 123
Item 2: Abc_123
Invalid, try again: zaZ9001
Item 3: !@#$%^&*()
Invalid, try again: root!root
Invalid, try again: root root
Item 4: This one is really long
Item 5: LASTONE

0 - Abc 123
1 - zaZ9001
2 - root root
3 - This one i
4 - LASTONE
```

If your program's output exactly matches the output shown above, given the provided inputs then your lab is complete and ready to be submitted (read below).

PART 2-Bonus:

BUILDING THE GAME (OPTIONAL)

If you have completed all previous game labs, you may complete this section in order to merge your current lab with you previous labs.

Copy the functions (and prototypes) into your codeQuest source code file. Delete the testing code associated with this lab. In both `Struct Player` and `Struct Item`, add a C-style string representing the names of the player and the name of the item. At the beginning of both `loadPlayer` and `loadItem` functions, prompt the user for a player name or an item name. Use the function above to retrieve and validate the names. Store it in a string along with the other player/Item variables.

Fix `printData` Function from Lab9 to display the names of both Player and Item.

If you've followed all instructions up to this point, you should be done CodeQuest! Download a binary version of CodeQuest v1.0 (CodeQuestV1.exe) to help you ensure your game works correctly!

To download the binary version, you can also click on the link below:

https://scs.senecac.on.ca/~jp.hughes/?q=node/32

SUBMISSION

Upload your solution according to your instructor’s guidelines.