Lab 9 [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(char str[]):` This function verifies that a string only contains letters (a-z, upper and lower case), numbers and spaces. This function accepts a pointer to a null terminated string and returns an integer. It returns 1 if all characters meet the above restraints, 0 otherwise.

`void Get_Valid(char str[])`: This function prompts the user to input a string from the user and stores it in str provided by the calling function. This function should scan for a string from the user (limited to the size of the array). Then, the function validates the string with the function above (`Validate_Alphanumberic`). This function accepts a pointer to a character array and an integer representing the size of the character array and returns nothing. If the sting 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. At the beginning of the character creation step, prompt the user for a user name and use the function above to retrieve and valid the name. Store it in a string along with the other player variables.

Add a new array for the inventory which will store item names associated with the other item information. Change the code where the item is acquired so that the item name is added as well. You may remove the old function that used to associate item IDs with item names now.

In the inventory, add a new option to sort the inventory items, when selected the sort function should be passed the names and an array of indices. After the sort is complete, the order of the IDs and the quantities will need to be updated to match the new order of the names.

SUBMISSION

Upload your solution according to your instructor’s guidelines.