National University of Computer and Emerging Sciences

Programming Fundamentals CS118

Lab Manual

Course Instructor	Mirza Mubasher Baig
Lab Instructor#01	Fraz Yousaf
Lab Instructor#02	Samia Akhter
Section	BDS-1A
Semester	FALL 2021

FAST School of Computing
FAST-NU, Lahore, Pakistan

National University of Computer and Emerging Sciences

	Lab No 10			
	Course Name	Programming Fundamentals	**Course Code**	CS188
	Program	BS(DS)	**Semester**	Fall 2021
	Duration	3 hours	**Total Points**	30+30+40
	Lab Date	17-Dec-2021	**Weight**	3%
	Section	BDS-1A	**Page(s)**	5

Topics Covered: Char Arrays

Use Visual Studio to write code for the following questions.

Submission Guidelines:

1. Save all .cpp files according to the following naming convention i.e.,
   {Section}_{RollNo}_{ProblemNo}.cpp
   For Problem#01: CX_21L-XXXX_P01.cpp
   For Problem#02: CX_21L-XXXX_P02.cpp
   For Problem#03: CX_21L-XXXX_P03.cpp

2. Now create a new folder according to the following naming convention i.e.,
   {Section}_{ROLLNO}_{LABNO}
   For students of C1: C1_21L-XXXX_L10
   For students of C2: C2_21L-XXXX_L10

3. Move all of your .cpp files to this newly created directory and compress it into a single .zip file.

4. Submit this compressed file on Google Classroom.

Even one-minute late submission would be considered as late and won't be accepted.

C-Strings

String is a collection of characters. There are two types of strings commonly used in C++ programming language:

- Strings that are objects of string class (The Standard C++ Library string class)
- C-strings (C-style Strings)

C-strings

In C programming, the collection of characters is stored in the form of arrays. This is also supported in C++ programming. Hence it's called C-strings.

C-strings are arrays of type `char` terminated with null character, that is, `\0` (ASCII value of null character is 0).

How to define a C-string?

```
char str[] = "C++";
```

In the above code, **`str` is a string and it holds 4 characters.**
Although, "`C++`" has 3 character, the null character `\0` is added to the end of the string automatically.

Alternative ways of defining a string

```
char str[4] = "C++";


char str[] = {'C','+','+','\0'};


char str[4] = {'C','+','+','\0'};
```

Like arrays, it is not necessary to use all the space allocated for the string. For example:

```
char str[100] = "C++";
```

Example 1: C++ String to read a word

C++ program to display a string entered by user.

```
#include <iostream>
using namespace std;

int main()
{
   char str[100];

   cout << "Enter a string: ";
   cin >> str;
   cout << "You entered: " << str << endl;

   cout << "\nEnter another string: ";
   cin >> str;
   cout << "You entered: "<<str<<endl;

   return 0;
}
```

Output

```
Enter a string: C++
You entered: C++

Enter another string: Programming is fun.
You entered: Programming
```

Notice that, in the second example only "Programming" is displayed instead of "Programming is fun".

This is because the extraction operator >> works as `scanf()` in C and considers a space " " has a terminating character.

Example 2: C++ String to read a line of text

C++ program to read and display an entire line entered by user.

```cpp
#include <iostream>
using namespace std;

int main()
{
   char str[100];
   cout << "Enter a string: ";
   cin.get(str, 100);

   cout << "You entered: " << str << endl;
   return 0;
}
```

Output

```
Enter a string: Programming is fun.
You entered: Programming is fun.
```

To read the text containing blank space, `cin.get` function can be used. This function takes two arguments.

First argument is the name of the string (address of first element of string) and second argument is the maximum size of the array.

In the above program, *str* is the name of the string and `100` is the maximum size of the array.

**********Enjoy Coding **********

Problem#01 **Marks (30)**

Write a function named ***countCharacter*** which takes three arguments i.e., a 1D char array named as ***Text***, an integer named as ***Length***, and a char named as ***Character***. Your function should return the number of times that Character occurs in the array.

You need to set these variables in ***main*** function and call ***countCharacter*** from there.

Sample Cases

Text ➔ {'H', 'E', 'L', 'L', O'}
Length ➔ 5
Character ➔
E

Output:
Occurrences of E are: 1

Text ➔ {‘H’, ’E’, ’L’, ’L’, O’}
Length ➔ 5
Character ➔
‘L’

Output:
Occurrences of L are: 2

Text ➔ “She sells sea shells on the sea shore”
Length ➔ 37
Character ➔ ‘s’

Output:
Occurrences of s are: 7

Problem#02 **Marks (30)**

Write a function named ***countAllLetters*** which takes two arguments i.e., a 1D char array named as ***Text***, an integer named as ***Length***. Your function should display the number of times each Letter is used in the Text.

You need to set these variables in ***main*** function and call ***countAllLetters*** from there.

Sample Cases

Text ➔ {'H', 'E', 'L', 'L', O'}
Length ➔ 5

Output:
Occurrences of E are: 1
Occurrences of H are: 1
Occurrences of L are: 2
Occurrences of O are: 1

Text ➔ "She Sells Sea Shells On The Sea Shore"
Length ➔ 37

Output:
Occurrences of A are: 2
Occurrences of E are: 7
Occurrences of H are: 4
Occurrences of L are: 4
Occurrences of N are: 1
Occurrences of O are: 2
Occurrences of R are: 1
Occurrences of S are: 8
Occurrences of T are: 1

Problem#03

Write a function to Remove all Characters including spaces in a String Except Alphabets.

Sample Cases

Enter a string: I2love !Programming

Output:
Filtered String: IloveProgramming

Problem#04

Your program should take a sentence as input in a character array which is null terminated. Your task is to pass this array to a function which reverses each word.

```
void reverseWords(char array[]);
```

Modify original array, print the result after the function call.

Sample Output:

```
Welcome to character arrays
Input taken

Output:
emocleW ot retcarahc syarra
```

Problem#05

Create a function that tweaks letters by one forward (+1) or backwards (-1) according to an array and store the result in an array named tweakedArray and print it in the function.

```
Example:
    tweakLetters("apple", {0, 1, -1, 0, -1}) ➞ "aqold"
    // "p" + 1 => "q"; "p" - 1 => "o"; "e" - 1 => "d"

    tweakLetters("many", {0, 0, 0, -1}) ➞ "manx"
    tweakLetters("rhino", {1, 1, 1, 1, 1}) ➞ "sijop"
```