Strings: Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null

Declaration & Initialization of strings	*(String declaration, input and output)*
| Strings are declared in C in similar manner as arrays. Only difference is that, strings are of char type:<br><br>**char s[5];**<br><br>In C strings can be initialized in many ways:<br>char c[ ]="abcd";<br>   OR,<br>char c[5]="abcd";<br>   OR,<br>char c[ ]={'a','b','c','d','\0'};<br>   OR;<br>char c[5]={'a','b','c','d','\0'};<br><br><table><tr><td>a</td><td>b</td><td>c</td><td>d</td><td>\0</td></tr></table><br>When, compiler encounters strings, it appends null character at the end of string | ```c
#include<stdio.h>
int main()
{
    char str[10];
    char name[20];
    int i;

//Taking Inputs with Loop
    for(i=0; i<5; i++)
    {
        fflush(stdin);
        printf("Enter character:");
        scanf("%c",&str[i]);
    }
    printf("%s",str);

//Taking Inputs without loops
    printf("\nEnter string:");
    scanf("%s", name);
    printf("%s", name);
    fflush(stdin);

//Taking string with space in between using gets & puts
    printf("\nEnter string2:");
    gets(name);
    puts(name);

    return 0;
}
``` |

strcpy(s1, s2) - Copies string s2 into string s1.	**strcmp(s1, s2)-** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
strcat(s1, s2) - Concatenates string s2 onto the end of string s1.	**strchr(s1, ch)-** Returns a pointer to the first occurrence of character ch in string s1. **strstr(s1, s2)-** Returns a pointer to the first occurrence of string s2 in string s1.
strlen(s) - Returns the length of string s.	

Example: strlen(str), strcat(str1,str2) & strcpy(str1,str2) function in C	
| ```c
#include<stdio.h>
#include<string.h>

int main()
{
  char str1[10],str2[10],str3[20];
  int len;

  printf("Enter String 1:");
  gets(str1);
  printf("Enter String 2:");
  gets(str2);
``` | ```c
  len=strlen(str1);
  printf("The length of the string 1 is: %d\n", len);

  strcat(str1,str2);
  printf("%s\n",str1);

  strcpy(str3,str1);
  printf("%s",str3);

  return 0;
}
``` |

Task (10 marks)

1. Take two string inputs, calculate lengths of both (without using strlen()) and display the smaller one.

2. Write a program to compare two strings without using C library function.

    ```
    Enter first strings :abc
    Enter Second strings :abc
    Strings are equal
    ```

3. Declare two strings A and B of size 100 and 50, respectively. Then take user input of both strings. Concatenate (join) B at the end of A using loop. Display the concatenated string.

    > Enter first string: Bangla
    > Enter second string: desh
    > After joining, first string: Bangladesh

4. Check whether an input string is palindrome or not. A string is a palindrome if it remains the same after you reverse it. For example, "racecar", "level", "12321", "madam" etc.

    ```
    Enter a string: racecar
    It's a palindrome
    ```

Prepared by Ashik Mostafa Alvi