CSE101-Lec#23

Strings

Outline

- Introduction to string
  - Declaration
  - Initialization
- Reading and writing strings
  - functions of the standard input/output library (stdio.h)
- Processing of strings.
  - String Manipulation Functions from the String Handling Library
  - Comparing strings
  - Determining the length of string
  - All string operations without inbuilt functions
  - Other programs related to strings

Fundamentals of strings

- Strings
  - Array of characters treated as a single unit called string:
    - Can include **letters**, **digits** and **special characters** (*, /, $)
  - String literal (string constant) - written in **double** quotes
    - “Lovely Professional University."

What is a String??

- String is a collection of characters terminated by null character
- Strings are arrays of characters
  - String is a **pointer** to first character (like array)
  - Value of string is the **address** of first character
- Each element of the string is stored in a **contiguous** memory locations.
- Terminated by a **null character**('\0') which is automatically inserted by the compiler to indicate the **end of string**.

String Definition

- They are defined as

```
char array_name[size];
```

e.g. char carname[30];

or char *carname;

- It defines an array name and reserves 30 bytes for storing characters and single character consumes 1 bytes each.
- Since the last byte is used for storing null character so total number of character specified by the user cannot exceed **29**.

String Initialization

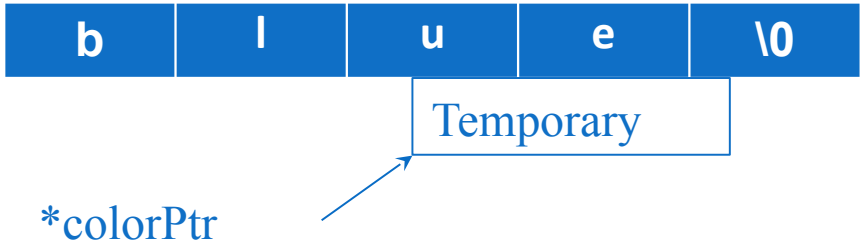
- String Initialization
  - Two ways:
  - Define as a character array or a variable of type char *

```
char color[] = "blue"; //char array
```

Or

```
char color[] = { 'b', 'l', 'u', 'e', '\0' };
char *colorPtr = "blue"; //pointer variable
```

  - Remember that strings represented as character arrays end with '\0'



String Input/Output

- Inputting strings
  - Use scanf.

    scanf("%s", word);
    - Copies input into word[]
    - **Do not need & (because a string is a pointer)**
  - Remember to leave last place in the array for '\0'.
  - Because array knows no bounds the string written beyond char array size will **overwrite** the data in memory.
- Displaying strings
  - Use printf.

    printf("%s", word);

Program to read and display string

```c
#include <stdio.h>
int main()
{
 char carname[20]; //string char array
 printf("Enter the name of your car: ");
 scanf("%s", carname);// to display the input
 printf("\nName of car is %s", carname);
} //end main
```

Output

```
Enter the name of your car: XUV500
Name of car is XUV500
```

How?

- The last program will print only a single word not the sentences with white spaces?
- That is if input is

  Lovely Professional University

- Output will be:

  Lovely

- So how to print:

  Lovely Professional University

use gets and puts

Standard I/O Library Functions

- List of functions in #include<stdio.h>
- Used for string input/output functions.

Function	Description
gets(char *s);	Inputs characters from the standard input into the array s until a **newline or end-of-file** character is encountered. A terminating null character is appended to the array.
puts(const char *s);	Prints the string s followed by a newline character.

```c
#include <stdio.h>
int main()
{
 char  name[100]; //string char array
 puts("\nEnter a string:");
 gets(name); //to input string with space
 printf("\nString is:")
 puts(name); //to output const string
}//end main
```

Program to print strings with white spaces using library functions

```
Enter a string:
Lovely Professional University
String is:
Lovely Professional University
```

Output

Drawback of gets():

gets() has been removed from c11. So it might give you a warning when implemented.

We see here that it doesn't bother about the size of the array.

So, there is a chance of Buffer_Overflow.

Alternative of gets():

To overcome the above limitation, we can use **fgets** as :

Syntax : char *fgets(char *str, int size, FILE *stream)

Example : fgets(str, 20, stdin); as here, 20 is MAX_LIMIT according to declaration.

```
#include <stdio.h>
#define MAX_LIMIT 20
int main()
{
char str[MAX_LIMIT];
fgets(str, MAX_LIMIT, stdin);
printf("%s", str);
return 0;
}
```

Multiple words input using scanf():

1)

```
#include <stdio.h>
int main()
{
char str[20];
scanf("%[^\n]%*c", str);
printf("%s", str);
return 0;
}
```

Here, **[]** is the scanset character. **^\n** tells to take input until newline doesn't get encountered. Then, with this **%*c**, it reads newline character and here used * indicates that this newline character is discarded.

2)

```
#include <stdio.h>
int main() {
  char str[100];
  scanf("%[^\n]s",str);
  printf("%s",str);
  return 0;
}
```

```c
#include <stdio.h>
int main()
{
 char  name[]="Hello"; //string char array
 int i=0;
 while(name[i]!='\0') //untill null character
  {
   printf("%c",name[i]);
   i++;
  }//end while
}//end main
```

Program to print strings character by character using loop.

```
Hello
```

Output

String Handling Library

- Functions defined in #include<string.h>
- String handling library provides **many** useful functions:
  - Manipulate string data(copy and concatenate)
  - Comparing strings
  - Determine string length

String Manipulation Functions(or Functions in string library)

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 );`	Copies the string `s2` into the character array `s1`. The value of `s1` is returned.
`char *strncpy( char *s1, const char *s2, size_t n );`	Copies at most `n` characters of the string `s2` into the character array `s1`. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 );`	Appends the string `s2` to `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n );`	Appends at most `n` characters of string `s2` to string `s1`. The first character of `s2` overwrites the terminating null character of `s1`. The value of `s1` is returned.
`int strcmp( const char *s1, const char *s2 );`	Compares the string `s1` with the string `s2`. The function returns a value of zero, less than zero or greater than zero if `s1` is equal to, less than or greater than `s2`, respectively.
`int strncmp( const char *s1, const char *s2, size_t n );`	Compares up to `n` characters of the string `s1` with the string `s2`. The function returns zero, less than zero or greater than zero if the `n`-character portion of `s1` is equal to, less than or greater than the corresponding `n`-character portion of `s2`, respectively.

More functions in string library

- strlen()-It is used to find the length of string without counting the null character
- strrev()-It is used to display the reverse of a string
- strlwr()-Converting a string from upper to lower case
- strupr()-Converting a string from lower to upper case

strcpy() and strncpy()

- **strcpy()** copies the entire **second** argument string into **first** argument.

```
strcpy( s1, s2);
```

- **strncpy()** copies the **first n** characters of **second** string argument into **first** string argument.

```
strncpy( s1, s2, 4);
```

  - A null character ('\0') is appended **explicitly** to first argument, because the call to strncpy in the program **does not** write a terminating null character.
  - The third argument is less than the string length of the second argument.

Examples

```
//strcpy() function
#include<stdio.h>
#include<string.h>
int main()
{
        //strcpy function
        char ori[20],dup[20];
        char *z;
        printf("\n Enter your name:");
        gets(ori);
        z=strcpy(dup,ori);
        printf("Original String is:
%s",ori);
        printf("\nDuplicate String is:
%s",dup);
        printf("\n Value of z is:%s",z);
        return 0;
}
```

```
//strncpy()
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[15],str2[15];
        int n;
        printf("\nEnter Source String:");
        gets(str1);
        printf("\nEnter Destination String:");
        gets(str2);
        printf("Enter number of characters to
copy in destination string:");
        scanf("%d",&n);
        strncpy(str2,str1,n);
        printf("Source string is:%s",str1);
        printf("\nDestination String is:
%s",str2);
        return 0;
}
```

strcat()

- Function **strcat appends its second argument string to its first argument string.**

strcat(s1, s2);

- The array used to store the first string should be large enough to store
  - the first string
  - the second string and
  - the terminating null character copied from the second string.

strncat()

- Function strncat appends a specified number of characters from the second string to the first string.

strncat(s1, s2, 6)

- A terminating null character is **automatically** appended to the result.

Examples

```
//strcat
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[20],str2[10];
        printf("\n Enter first string:");
        gets(str1);
        printf("\n Enter second string:");
        gets(str2);
        strcat(str1,str2);
        printf("\n String after concatenation:
%s",str1);
        return 0;
}
```

```
//strncat
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[20],str2[10];
        int n;
        printf("\n Enter first string:");
        gets(str1);
        printf("\n Enter second string:");
        gets(str2);
        printf("\n Enter number of
characters you want to combine:");
        scanf("%d",&n);
        strncat(str1,str2,n);
        printf("\n String after
concatenation:%s",str1);
        return 0;
}
```

Comparison Functions of the String Handling Library

- Comparing strings
  - Computer compares numeric ASCII codes of characters in string
  - **strcmp()** Compares its first string argument with its second string argument, character by character.
  - Function **strncmp()** does not compare characters following a null character in a string.

strcmp()

int strcmp(const char *s1, const char *s2);

- Compares string s1 to s2
- Returns
  - a negative number if s1 < s2,
  - zero if s1 == s2
  - a positive number if s1 > s2

strncmp()

int strncmp(const char *s1, const char *s2, `int` n);

- Compares up to n characters of string s1 to s2
  - a negative number if s1 < s2,
  - zero if s1 == s2
  - a positive number if s1 > s2

Examples

```
//strcmp
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[20],str2[10];
        int x;
        printf("\n Enter first string:");
        gets(str1);
        printf("\n Enter second string:");
        gets(str2);
        x=strcmp(str1,str2);
        if(x==0)
        {
                printf("\n Strings are equal");
        }
        else if(x>0)
        {
                printf("\n First string is greater
than second string(strings are not equal)");
        }
        else
        {
                printf("\n First string is less than
second string(strings are not equal)");
        }
        return 0;
}
```

```
// strncmp
#include<stdio.h>
#include<string.h>
int main()
{
        char str1[20],str2[10];
        int x,n;
        printf("\n Enter first string:");
        gets(str1);
        printf("\n Enter second string:");
        gets(str2);
        printf("\n Enter no. of characters to compare:");
        scanf("%d",&n);
        x=strncmp(str1,str2,n);
        if(x==0)
        {
                printf("\n Strings are equal");
        }
        else if(x>0)
        {
                printf("\n First string is greater than
second string(strings are not equal)");
        }
        else
        {
                printf("\n First string is less than
second string(strings are not equal)");
        }
        return 0;
}
```

stricmp()[Ignore case], stricmp will ignore the case

```
#include<stdio.h>
int main()
{
        char str1[20],str2[10];
        int x;
        printf("\n Enter first string:");
        gets(str1);
        printf("\n Enter second string:");
        gets(str2);
        x=stricmp(str1,str2);
        if(x==0)
        {
                printf("\n Strings are equal");
        }
        else if(x>0)
        {
                printf("\n First string is greater than second string(strings are not equal)");
        }
        else
        {
                printf("\n First string is less than second string(strings are not equal)");
        }
        return 0;
}
//consider str1(HELLO) and str2(hello) and if we apply stricmp on these strings, then 0 will be returned, as strings are equal
```

Determining the length of string

strlen()

- ***Function*** strlen in #include<string.h>
- Function **strlen()** takes a **string** as an argument and returns the **number** of characters in the string
  - the terminating null character is not included in the length

Example

```
#include<stdio.h>
#include<string.h>
int main()
{
char str[]="Hello";
printf("\n Length of the given string is:%d",strlen(str));
return 0;
}
```

strrev()-Example

```
#include<stdio.h>
#include<string.h>
int main()
{
        char s[100]="Hello";
        printf("%s",strrev(s));
        return 0;
}
```

strlwr(),strupr()-Examples

```
#include<stdio.h>
#include<string.h>
int main()
{
        char s[]="hello";
        strupr(s);
        puts(s);
        strlwr(s);
        puts(s);
        return 0;
}
```

All string operations without inbuilt functions

- Copying one string to another
- Finding length of a string
- Concatenation(or Combining) of two strings
- Comparing two strings
- Displaying reverse of a number
- Checking whether a given string is palindrome or not
- Converting all characters of a given string from lowercase to uppercase
- Converting all characters of a given string from uppercase to lowercase

WAP to copy one string to another without using strcpy()/or inbuilt function

```
#include<stdio.h>
int main() {
   char s1[100], s2[100];
   int i;
   printf("\nEnter the string :");
   gets(s1);//Hello
   i = 0;
   while (s1[i] != '\0') {
      s2[i] = s1[i];
      i++;
   }
   s2[i] = '\0';
   printf("\nCopied String is %s ", s2);
   return (0);
}
```

WAP to find the length of a string without using strlen()/ or inbuilt function

```
#include<stdio.h>
int main()
{
        char x[100];
        int i=0;
        printf("\n Enter String:");
        gets(x);
        while(x[i]!='\0')
    {
                i++;
          }
        printf("\n Length of the string is:%d",i);
        return 0;
}
```

WAP to concatenate(or combine) two strings without using strcat/ or inbuilt function

```
#include<stdio.h>
int main()
{
        char str1[100],str2[100],str3[200];
        int i=0,j=0;
        printf("\n Enter the first string:");
        gets(str1);
        printf("\n Enter the second string:");
        gets(str2);
        while(str1[i]!='\0')
        {
                str3[j]=str1[i];
                i++;
                j++;
        }
        i=0;
        while(str2[i]!='\0')
        {
                str3[j]=str2[i];
                i++;
                j++;
        }
        str3[j]='\0';
        printf("\n The concatenated string is:");
        puts(str3);
        return 0;
}
```

WAP to compare two strings without using strcmp()/ or inbuilt function

```
#include <stdio.h>
#include<string.h>
int main ()
{
 // declare variables
  char str1 [30], str2 [30];
  int i = 0, flag=0 ,length1, length2, length;
 // take two string input
 printf ("Enter string1:");
 gets (str1);
 printf ("\nEnter string2:");
 gets (str2);
 //length of both string
  length1 = strlen (str1);
  length2 = strlen (str2);
  if(length1>length2)
  length=length1;
  else
  length=length2;
```

•

```
while (i<length)
   {
    if( str1 [i] == str2 [i])
     {
      i++;
      continue;
     }
 if( str1 [i] < str2 [i])
     {
      flag = -1;
      break;
     }
    if( str1 [i] > str2 [i])
     {
      flag = 1;
      break;
     }
 }
 if (flag == 0)
   printf ("\nBoth strings are equal ");
 if(flag == -1)
   printf ("\nstring1 is less than string2 ");
 if( flag == 1)
   printf ("\nstring1 is greater than string2 ");
   return 0;
}
```

Dry running

Comparing two strings

```
str1 → First string      flag=0;
str2 → Second string     i=0
length1 = strlen(str1);
length2 = strlen(str2);
if( length1 > length2)
{
 length = length1);
}
else
{
 length= length2;
}
while( i<length )
 if (str1[i]==str2[i])
 {
   i++;
   continue;
 }
 if( str1[i]< str2[i])
 {
   flag=-1;
   break;
 }
 if( str1[i]>str2[i])
 {
   flag=1;
   break;
 }
if( flag==0)
printf("Equal");
if( flag==-1)
 printf("S1 is less than S2");
if( flag==1)
 printf("\n S1 is > than S2");
```

Consider

str1 → Have
str2 → Has

length1 = 4
length2 = 3

4 > 3

length = 4.

i=0, 0 < 4 (True)
H == H (True)
i=1
Continue (Next Iteration)
1<4 (True)
a==a (True)
i=2
Continue (Next Iteration)
2<4 (True)
v==s (False)
v<s (False)
v>s (True)
flag=1;
break; [Loop terminates]
1==1 (flag==1) True
S1 is > (greater than) S2

WAP to display the reverse of a given string without strrev()/ or inbuilt function

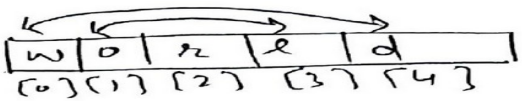
```
#include<stdio.h>
#include<string.h>
int main() {
  char str[100], temp;
  int i, j;
  printf("\nEnter the string :");
  gets(str);
  i = 0;
  j = strlen(str) - 1;
  while (i < j) {
    temp = str[i];
    str[i] = str[j];
    str[j] = temp;
    i++;
    j--;
  }
  printf("\nReverse string is :%s", str);
  return (0);
}
```

Dry running

Reverse of a given string
Without using strrev().

```
i = 0;                    [str → string]
j = strlen(str) - 1;
while (i < j)
{
    temp = str[i];
    str[i] = str[j];
    str[j] = temp;
    i++;
    j--;
}
printf("\n Reverse string is: %s", str);
```

str = "world"



i = 0;
j = 5 - 1 = 4

0 < 4 (True)

temp = w
w = d
d = w
} Swapping

i = 1
j = 3

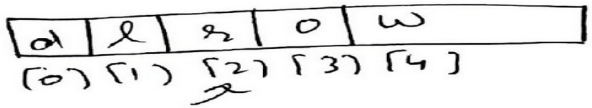
1 < 3 (True)

temp = o
o = l
l = o
} Swapping

i = 2
j = 2

2 < 2 (false)
Loop stops.

So, str:



Reverse

WAP to check whether the given string is palindrome or not(without using strrev())

```
#include<stdio.h>
#include<string.h>
int main() {
  char str[100], temp;
  char str1[100];
  int i, j;
  printf("\nEnter the string :");
  gets(str);
  i = 0;
  j = strlen(str) - 1;
  strcpy(str1,str);
  while (i < j)
  {
    temp = str[i];
    str[i] = str[j];
    str[j] = temp;
    i++;
    j--;
  }
  if(strcmp(str1,str)==0)
   {
        printf("\n Given String is Palindrome");
   }
   else
   {
        printf("\n Not a Palindrome");
   }
   return (0);
}
```

WAP to convert all characters of a given string into uppercase without using strupr()/or inbuilt function

```
#include<stdio.h>
#include<string.h>
int main()
{
char str1[10];
int i,len;
printf("Enter any string \t");
gets(str1);
len=strlen(str1);
for(i=0;i<len;i++)
        {
        if(str1[i]>='a' && str1[i]<='z')

        str1[i]=str1[i]-32;
        }
puts("string in upper is");
puts(str1);
return 0;
}
```

WAP to convert all characters of a given string into lowercase without using strlwr()/or inbuilt function

```
#include<stdio.h>
#include<string.h>
int main()
{
char str1[10];
int i,len;
printf("Enter any string \t");
gets(str1);
len=strlen(str1);
for(i=0;i<len;i++)
        {

        if(str1[i]>='A' && str1[i]<='Z')
        str1[i]=str1[i]+32;
        }
puts("string in lower is");
puts(str1);
return 0;
}
```

More programs on strings

WAP to sort the characters of a given string into ascending order

```
#include<stdio.h>
#include<string.h>
int main()
{
char s[10],t;
int n,i,j;
printf("\n Enter String:");
gets(s);
n=strlen(s);
for(i=0;i<n-1;i++)
{
for(j=0;j<n-i-1;j++)
{
if(s[j]>s[j+1])
{
   t=s[j];
   s[j]=s[j+1];
   s[j+1]=t;
}
}
}
printf("%s",s);
}
```

WAP to count vowels in a given string

```
#include<stdio.h>
int main()
{

        char x[100];
        int i=0,count=0;
        printf("\n Enter the string:");
        gets(x);
        while(x[i]!='\0')
        {
if(x[i]=='a'||x[i]=='e'||x[i]=='i'||x[i]=='o'||x[i]=='u'||x[i]=='A'||x[i]=='E'||x[i]=='I'||
x[i]=='O'||x[i]=='U')
          {
                count++;
           }
           i++;
        }
        printf("\n Number of vowels in the string are:%d",count);
        return 0;
}
```

WAP to traverse all characters of a given string using pointer to character

```
#include<stdio.h>
int main()
{
        char *g="C Programming";
        int length=0,i=0;
   while(*g!='\0')
        {
                printf("%c",*g);//Value at address
                g++;//Pointer is incremented by 1 after each iteration
                length++;//Variable for counting length
        }
        printf("\nLength of the string is:%d",length);
        return 0;
}
```

WAP to count total no. of characters and words in a given string

```
#include<stdio.h>
int main()
{
        char x[100];
        int i=0,length=0,c=0,w=1;
        printf("\n Enter String:");
        gets(x);
        while(x[i]!='\0')
    {
                if(x[i]==' ' && x[i+1]!=' ')
                 {
                    w++;
                 }
                 c++;
                 i++;
           }
        printf("\n Total number of characters are:%d, and no. of words are:%d",c,w);
        return 0;
}
```

WAP to demonstrate array of strings in C

```
#include<stdio.h>
int main()
{
        char names[5][10];
        int i,n;
        printf("\n Enter the number of students:");
        scanf("%d",&n);
        fflush(stdin);
        for(i=0;i<n;i++)
        {
                printf("\n Enter the name of student %d: ",i+1);
                gets(names[i]);
        }
        printf("\n Names of the students are:\n");
        for(i=0;i<n;i++)
        puts(names[i]);
        return 0;
}
```

WAP to traverse a string character by character

```
#include <stdio.h>
int main()
{
 char  name[]="Hello  World"; //string char array
 int i=0;
 while(name[i]!='\0') //untill null character
  {
   printf("%c\n", name[i]);
   i++;
  }//end while
}//
```

WAP to replace all spaces in a given string with '$'[Example for character replacement]

```
#include<stdio.h>
int main()
{
        char x[100];
        int i=0;
        printf("\n Enter the string:");
        gets(x);
        while(x[i]!='\0')
        {
    if(x[i]==' ')
    {
        x[i]='$';//Character replacement
    }
    i++;
   }
  printf("\n String after character replacement is:%s",x);
  return 0;
}
```

Output-1??

```
#include<stdio.h>
int main()
{
char str[]="Practice MCQ";
printf("\n%d",sizeof(str));
return 0;
}
```

A. 13
B. 12
C. 11
D. 1

Output-2??

```
#include<stdio.h>
int main()
{
char str[]="Program";
printf("%c",str[7]);
return 0;
}
```

A. m
B. Program
C. Compile time error
D. Nothing will be visible

Output-3??

```
#include<stdio.h>
int main()
{
char str1[]="Good";
char str2[5];
str2=str1;
printf("%s",str2);
return 0;
}
```

A. Good
B. Garbage value
C. Compile time error
D. Nothing will be visible

Output-4??

```
#include<stdio.h>
int main()
{
char str1[]="Good";
char *str2;
str2=str1;
puts(str2);
return 0;
}
```

A. Good
B. Garbage value
C. Compile time error
D. Nothing will be visible

Output-5??

```
#include<stdio.h>
#include<string.h>
int main()
{
char str[20]="Example";
printf("%d %d",sizeof(str),strlen(str));
return 0;
}
```

A. 7 7
B. 20 20
C. 20 7
D. 20 8

Output-6??

```
#include<stdio.h>
#include<string.h>
int main()
{
char str1[20]="Example";
char str2[30]="Exam";
if(strncmp(str1,str2,4))
printf("\nHello");
else
printf("\nWorld");
return 0;
}
```

A. Hello
B. World
C. Nothing will be printed
D. Compile time error

Output-7??

What will be the output of the program ?

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[20] = "Hello", str2[20] = " World";
   printf("%s\n", strcpy(str2, strcat(str1, str2)));
   return 0;
}
```

A. Hello
B. World
C. Hello World
D. WorldHello

Output-8??

In below program, what would you put in place of “?” to print “Quiz”?

```
#include <stdio.h>
int main()
{
  char arr[] = "HelloQuiz";
  printf("%s", ?);
  return 0;
}
```

A. arr

B. (arr+5)

C. (arr+4)

D. Not possible

Question-9

Which of the following C code snippet is not valid?

(A) char* p = “string1”; printf(“%c”, *++p);

(B) char q[] = “string1”; printf(“%c”, *++q);

(C) char* r = “string1”; printf(“%c”, r[1]);

(D) None of the above

Output-10 ??

```
#include<stdio.h>

int main()

{

    printf(8+"C Programming\n");

    return 0;

}
```

A. mming

B. ming

C. amming

D. gramming

Output-11 ??

```
//Assume unsigned integer takes 4 bytes
#include <stdio.h>
int main()
{
    char *str1 = "Hello";
    char str2[] = "Hello";
    printf("sizeof(str1) = %d, sizeof(str2) = %d",sizeof(str1), sizeof(str2));
    return 0;
}
```

A. sizeof(str1) = 8, sizeof(str2) = 6

B. sizeof(str1) = 4, sizeof(str2) = 6

C. sizeof(str1) = 4, sizeof(str2) = 4

D. sizeof(str1) = 6, sizeof(str2) = 4

Output-12 ??

Predict the output of the following program:

```
#include <stdio.h>
int main()
{
    char str[] = "%d %c", arr[] = "HelloWorld";
    printf(str, 0[arr], 2[arr + 3]);
    return 0;
}
```

A. 72 W

B. H W

C. W H

D. Compile-time error

Output-13??

```
#include <stdio.h>
int main()
{
   char *str="WORLD";
   while(*++str)
   {
      printf("%c",*str);
   }
   return 0;
}
```

A. ORLD

B. WORLD

C. RLD

D. Compile time error