LECTURE 12

1. Types of Functions:

The user defined functions may be classified in the following three ways based on the formal arguments passed and the usage of the return statement, and based on that, there are three of user defined functions:

1. A function is invoked without passing any formal argument from the calling portion of a program and also the function does not return back any value to the called function.
2. A function is invoked with formal arguments from the calling portion of a program but the function does not return back any value to the calling portion.
3. A function is invoked with formal arguments from the calling portion of a program which return back a value to the calling environment.

Here are two programs that find the square of the number using and not using a return statement.

Example 1:

```
# include <iostream.h>
Void main()
{
Void square (int);
Int max;
Cout<<"Enter a value for n ?\n";
Cin>>max;
For (int i=0;i<=max-1;++i)
Square (i)
}
Void square(int n)
{
Float value;
Value=n*n;
Cout<<"i="<<n<<"square="<<value<<endl;
}
```

```
# include <iostream.h>
Void main()
{
float square (float);
float I,max,value;
max=1.5;
i=-1.5;
while (i<=max) {
value=square(i);
Cout<<"i="<<i<<"square="<<value<<endl;
I=i+0.5;
}
}
Float square(float n)
{
Float value;
Value=n*n;
Return (value);
}
```

Example 2:

write C++ program, using function to find the sumation of the given series: Sum=x-(x^3)/3!+(x^5)/5!- …(x^n)/n!

```
#include <iostream.h>
Void main(void)
{
Long int fact (int);
Float power(float,int);
Float sum,temp,x,pow;
Int sign,I,n;
Longint factorial;

Cout<<"Enter a value for n?"<<endl;
Cin>>n;
Cout<<"Enter a value for x ?"<<endl;
Cin>>x;
I=3;  sum=x;   sign=1;
While (i<=n) {
Factval=fact(i);
Pow=power(x,i);
Sign=(-1)*sign;
Temp=sign*pow/factval;
Sum=sum+temp;
I=i+2;
}
Cout<<"sum of series ="<<sum;
}


Long int fact (int max)
{
Long intvalue;
Value=1;
For(int i=1;i<=max;++i)
Value=value*I;
Return (value);
}

Float power (float x, int n)
{
Float value2;
Value2=1;
For(int j=1;j<=n;++j)
Value2=value2*x;
Return(value2);
}
```

2. Actual and Formal Arguments:

The arguments may be classified under two groups, actual and formal arguments:

(a) **Actual arguments:** An actual argument is a variable or an expression contained in a function call that replaces the formal parameter which is a part of the function declaration. Sometimes, a function may be called by a portion of a program with some parameters and these parameters are known as the actual arguments.

(b) **Formal arguments:** are the parameters present in a function definition which may also be called as dummy arguments or the parametric variables. When the function is invoked, the formal parameters are replaced by the actual parameters. Formal arguments may be declared by the same name or by different names in calling a portion of the program or in a called function but the data types should b e the same in both blocks

For example:

```
# include <iostream.h>
Void main()
{
Int x,y;
Void output (int x, int y);              // function declaration
__
__
Output (x,y);                    // x and y are actual arguments
}
Void output (int a, int b)               // forma or dummy arguments
{
                  // body of function
 }
```

3. Local and Global variables:

The variables in general bay be classified as local or global variables.

(a) **Local variables:** Identifiers, variables and functions in a block are said to belong to a particular block or function and these identifiers are known

as the local parameters or variables. Local variables are defined inside a function block or a compound statement. For example,

```
Void func (int l, int j)
{
Int k,m;                    // local variables
……                          // bodyofthe function
}
```

Local variables are referred only the particular part of a block or a function. Same variable name may be given to different parts of a function or a block and each variable will be treated as a different entity.

(b) Global variables: these are variables defined outside the main function block. These variables are referred by the same data type and by the same name through out the program in both the calling portion of the program and in the function block.

Example 3:

 A program to find the sum of the given two numbers using the global variables.

```
#include <iostream.h>
Int x;
Int y=5;

void main( )
{
        X=10;
        Void sum(void);
        Sum();
 }
Void sum(void)
{
        Int sum;
        Sum=x+y;
        Cout<<"x="<<x<<endl;
        Cout<<"y="<<y<<endl;
        Cout<<"sum="<<sum<<endl;
}
```

4. Recursive Functions:

A function which calls itself directly or indirectly again and again is known as the recursive function. Recursive functions are very useful while constructing the data structures like linked lists, double linked lists, and trees. There is a distinct difference between normal and recursive functions. A normal function will be invoked by the main function whenever the function name is used, where as the recursive function will be invoked by itself directly or indirectly as long as the given condition is satisfied. Forexample,

```
# include <iostrem.h>
Void main(void)
{
Void func1();  //function declaration
_____
_____
Func1();  //function calling
}
Void func1()  //function definition
{
____
____
Func1();  //function calls recursively
}
```

Example 4:

 A program to find the sum of the given non negative integer numbers using a recursive function sum=1+2+3+4+…+n

```
#include <iostream.h>
Void main(void)
{
Int sum(int);
Int n,temp;
Cout<<"Enter any integer number"<<endl;
Cin>>n;
Temp=sum(n);
Cout<<"value="<<n<<"and its sum="<<temp;
}
Int sum(int n)  //recursive function
{
Int sum(int);   //local function declaration
```

```
Int value=0;
If (n==0)
Return (value);
Else
Value=n+sum(n-1);
Return (value);
}
```

The output of the above program:

Enter any integer number

Value = 11 and its sum=66

The following illustrations will be helpful to understand the recursive function

For value 1	For value 2	For value 3
=1+sum(1-1) =1+0 =1	=2+sum(2-1) =2+1+sum(1-1) =3	=3+sum(3-1) =3+sum(2-1) =3+2+1+sum(1-1) =6

Example 5:

 A program to find the factorial (n!) of the given number using the recursive function.Its the product of all integers from 1 to n (n is non negative) (so n!=1 if n=0 and n!=n(n-1) if n>0)

```
#include <iostream.h>
Void main(void)
{
Long int fact (long int);
Int x,n;
Cout<<"Enter any integer number"<<endl;
Cin>>n;
X=fact(n);
Cout<<"value="<<n<<"and its factorial=";
Cout<<x<<endl;
}
Long int fact (long int n)  //recursive function
{
Long int fact(long int);  //local function declaration
Int value =1;
If (n==1)
Return(value)
Else
{value=n*fact(n-1);
Return(value);
} }
```

The output of the above program:

Enter any integer number

Value = 5 and its factorial=120

The following illustrations will be helpful to understand the recursive function

For value 1 **=1*fact(1-1)** **=1**	**For value 2** **=2*fact(2-1)** **=2*1** **=2**	**For value 3** **=3*fact(3-1)** **=3*2*fact(2-1)** **3*2*1** **=6**

WORK SHEET (5)

Functions

Q1: Write a C++ program, using function, to counts uppercase letter in a 20 letters entered by the user in the main program.

Q2: Write a C++ program, using function, that reads two integers (feet and inches) representing distance, then converts this distance to meter.
Note: 1 foot = 12 inch
1 inch = 2.54 Cm

i.e.:
Input: feet: 8 or inches: 9

Q3: Write a C++ program, using function, which reads an integer value (T) representing time in seconds, and converts it to equivalent hours (hr), minutes (mn), and seconds (sec), in the following form:
hr : mn : sec

i.e.:
Input: 4000
Output: 1 : 6 : 40

Q4: Write a C++ program, using function, to see if a number is an integer (odd or even) or not an integer.

Q5: Write a C++ program, using function, to represent the permutation of n.

Q6: Write a C++ program, using function, to inputs a student's average and returns 4 if student's average is 90-100, 3 if the average is 80-89, 2 if the average is 70-79, 1 if the average is 60-69, and 0 if the average is lower than 60.

Q7: The Fibonacci Series is: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... It begins with the terms 0 and 1and has the property that each succeeding term is the sum of the two preceding terms. Write a C++ program, using function, to calculate the nth Fibonacci number.

Q8: Write a C++ program, using function, to calculate the factorial of an integer entered by the user at the main program.

Q9: Write a C++ program, using function, to evaluate the following equation:

$$z = \frac{x! - y!}{(x - y)!}$$

Q10: Write a C++ program, using function, to test the year if it's a leap or not.

Note: *use* *y % 4 == 0 && y % 100 != 0 :: y % 400 ==0*

Q11: Write a C++ program, using function, to find x^y.

Note: *use* ***pow*** *instruction with* ***math.h*** *library.*

Q12: Write C++ program, using function, to inverse an integer number:
For example: 765432 → 234567

Q13: Write C++ program, using function, to find the summation of student's marks, and it's average, assume the student have 8 marks.

Q14: Write C++ program, using function, to convert any char. From capital to small or from small to capital.

Q15: Write C++ program using recursive function to find the power of n numbers.