RECURSION

Recursion is defined as the function that calls itself repeatedly until condition is reached. But while using **recursion**, programmers need to be careful to define an exit condition from the function; otherwise it will go into an infinite loop.

Syntax:

```
Function1()
{
    Function1();
}
```

Example:

Calculating the factorial of a number

Fact (n)= n*fact(n-1);

6! = 6*fact(n);

6! = 6 *5*fact(4)

6! = 6 * 5 * 4 *fact(3)

6! = 6 * 5 * 4 * 3 *fact(2)

6! =6 *5 * 4 * 3 *2 * fact(1)

6! = 6 *5 * 4 * 3 *2 * 1

6!=120

Advantage of recursion

- Recursion makes program elegant and cleaner.
- All algorithms can be defined recursively which makes it easier to visualize and prove.
- Reduce unnecessary calling of function
- Easy to solve complex problems

Direct Recursion:

A function is directly recursive if it calls itself.

```
A( )
{
    ….
    A( ); // call to itself
```

```
        ….
}
```

Indirect Recursion:

Function calls another function, which in turn calls the original function.

```
A( )
{
        …
        B( );
        …
}
B( )
{
        …
        A( );// function B calls A
        …
}
```

Linear Recursion - It makes only one recursive call.

Binary Recursion - It calls itself twice.

N-ary recursion - It calls itself n times.

Program 1 : Find factorial using recursion

```
#include<stdio.h>
#include<conio.h>
int fact(int);
void main()
{
        int n, Result;
        printf("\n Enter any number:");
        scanf("%d", &n);
        Result = fact(n);
        printf ("Factorial value = %d", Result);
        getch();
}
int fact (int x)
{
        if (x == 0)
                return 1;
        else
                return x *  fact(x – 1);
}
```

Output:

```
Enter any number: 4
Factorial value = 24
```

Program 2 : Find the GCD of Two Positive Integer Numbers

```
#include<stdio.h>
#include<conio.h>
int gcd (int a, int b)
void main ()
{
        int a, b;
        printf("\n Enter the two numbers:");
        scanf ("%d%d", &a, &b);
```