Recursion

Any function which calls itself is called a recursive function, and such function calls are called recursive calls. Recursion involves several numbers of recursive calls. However, it is important to impose a termination condition of recursion.

Recursion cannot be applied to all the problems, but it is more useful for the tasks that can be defined in terms of similar subtasks. For Example, recursion may be applied to sorting, searching, and traversal problems.Generally, iterative solutions are more efficient than recursion since function call is always overhead.

//Factorial of a number using recursion

```
#include<stdio.h>
int factorial(int number);
int main()
{
   int num;
   printf("Enter the number you want factorial of\n");
   scanf("%d",&num);
   printf("The factorial of %d is %d\n",num,factorial(num));

   return 0;                                          //fact(n)=n*fact(n-1)
}
int factorial(int number)                             //   Suppose number=5
{                                                     //1.    5 * fact(4)
   if(number==0 || number ==1)                        //2.     5 * 4 * fact(3)
   {                                                  //3.    5 * 4 *3* fact(2)
      return 1;   //basecondition                     //4.     5 * 4 * 3 * 2 * fact(1) ->since fact(1)=1
                                                      //Therefore 120
   }
   else
   {
      return ( number * factorial (number-1));//recursive call condition
   }
}
```

//Fibonacci series

```c
#include<stdio.h> // include stdio.h library
int fibonacci(int num);

int main(void)
{
   int terms;

   printf("Enter terms: ");
   scanf("%d", &terms);

   for(int n = 0; n < terms; n++)
   {
      printf("%d ", fibonacci(n));
   }

   return 0;
}

int fibonacci(int num)
{
                                                        //suppose terms=5
   //base condition                                     //fibonacci(0)--> 0
   if(num == 0 || num == 1)                            //fibonacci(1)-->1
   {                                                  //fibonacci(2)->fibonacci(2-1)+fibonacci(2-2)
      return num; //base condition                     //          =fibonacci(1)+fibonacci(0)
                                                       //          =1 +0 =1
   }                                                  //fibonacci(3)->fibonacci(3-1)+fibonacci(3-2)
                                                       //          =fibonacci(2)+fibonacci(1)
   else                                               //          =1+1 =2
   {                                                  //fibonacci(4)->fibonacci(4-1)+fibonacci(4-2)
      // recursive call                               //          =fibonacci(3)+fibonacci(2)
      return fibonacci(num-1) + fibonacci(num-2);      //        =2+1 =3
   }

}
```