Introduction of Recursion:

Recursion is the process which comes into existence when a function calls a copy of itself to work on a smaller problem.

Any function which calls itself is called 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 code is shorter than iterative code however it is difficult to understand.

Recursion cannot be applied to all the problem, 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.

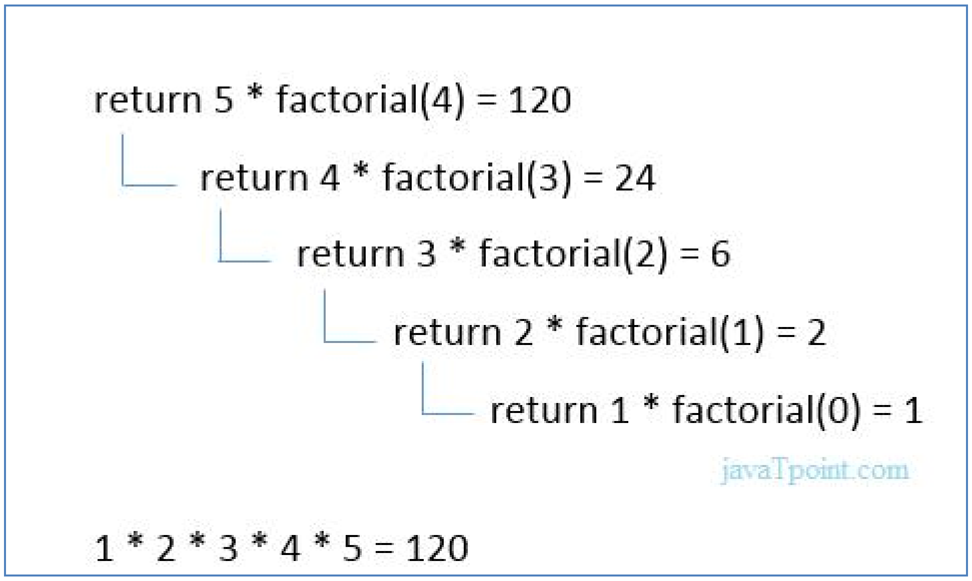
Generally, iterative solutions are more efficient than recursion since function call is always overhead.

Any problem that can be solved recursively, can also be solved iteratively. However, some problems are best suited to be solved by the recursion, for example, tower of Hanoi, Fibonacci series, factorial finding, etc.

Example:

```
#include <stdio.h>
int fact (int);
int main()
{
   int n,f;
   printf("Enter the number whose factorial you want to calculate?");
   scanf("%d",&n);
   f = fact(n);
   printf("factorial = %d",f);
}
int fact(int n)
{
   if (n==0)
   {
      return 0;
   }
   else if ( n == 1)
   {
      return 1;
   }
        else
        {
           return n*fact(n-1);
        }
     }
```

We can understand the above program of the recursive method call by the figure given below:


return 5 * factorial(4) = 120
return 4 * factorial(3) = 24
return 3 * factorial(2) = 6
return 2 * factorial(1) = 2
return 1 * factorial(0) = 1
javaTpoint.com
1 * 2 * 3 * 4 * 5 = 120

Recursive Function

A recursive function performs the tasks by dividing it into the subtasks.

There is a termination condition defined in the function which is satisfied by some specific subtask.

After this, the recursion stops and the final result is returned from the function.

The case at which the function doesn't recur is called the base case whereas the instances where the function keeps calling itself to perform a subtask, is called the recursive case.

All the recursive functions can be written using this format.

Pseudocode for writing any recursive function is given below.

```
if (test_for_base)
{
   return some_value;
}
else if (test_for_another_base)
{
   return some_another_value;
}
else
{
   // Statements;
   recursive call;
}
```

Example: Fibonacci Series

```
#include<stdio.h>
int fibonacci(int);
void main ()
{
   int n,f;
   printf("Enter the value of n?");
   scanf("%d",&n);
   f = fibonacci(n);
   printf("%d",f);
}
int fibonacci (int n)
{
   if (n==0)
   {
   return 0;
```

```
    }
    else if (n == 1)
    {
        return 1;
    }
    else
    {
        return fibonacci(n-1)+fibonacci(n-2);
    }
}
```