Recursion

It is the mechanism in which a function calls itself to solve a problem efficiently.

In recursion a problem is dependent upon a smaller or similar problem just like itself.

Example: - 2^4 means $2 * 2^3$

And 2^3 means $2 * 2^2$

And 2^2 means $2 * 2^1$

And 2^1 means $2 * 2^0$

And 2^0 means 1 which is the **base case** of our problem.

So, in shorter form this form is converted as **Num=Num*Num$^{(power-1)}$**

This converted form also known as **Recursive relation.**

Three main components of Recursion: -

Base case: - Base case is the condition which tells you that where you want to stop int the recursion.

Recursive relation: - Recursive relation or recurrence relation is the formula which is mandatory in recursion and it is basically a formula which is used by the function to perform operation and obtain the exact result.

Processing part: - This part includes the statement printing and LOC which is important to the function or related to your result.

LOC stands for lines of code.

Types of Recursion: -

There are two types of Recursion present:

(i) Direct Recursion
(ii) Indirect recursion
(iii) Head Recursion

(iv) Tail recursion

Direct Recursion: - It is the simple recursion in which a single function is calling itself.

Syntax: - Return Type function_main (args list) {

```
        Statements;
        function_main (args list);
    }
```

Indirect Recursion: - In this type of recursion a function calls another function and this function calls the previous function.

Example: - Function_1 call the Function_2 and in the next step Function_2 calls the Function_1 this shows the perfect example of Indirect Recursion.

Syntax: - Return Type function_one (args list) {

```
        Statements;
        Function_two (args list);
}
Return Type Function_two (args list) {
    Statements;
    Function_one (args list);
}
```

Head Recursion: - In this type of recursion the recursive relation is at the top of the processing part.

Example: - Return Type function (args list) {

```
    Base case;
  Recursive relation;
    Processing part;
```

}

Tail Recursion: - In this type of recursion the recursive relation is at the top of the processing part.

Example: - Return Type function (args list){

Base case;

Processing Part;

Recursive relation;

}

Note: - When you create a recursion it is mandatory to create a base case and a recurrence relation. And if base case is not int the recursive function then Segmentation Fault occur.

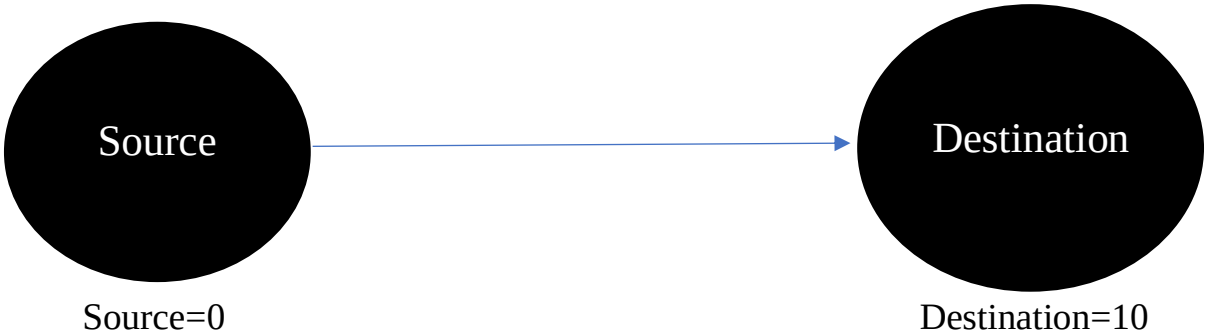
How Recursion works: -

Step 1 → The base case will be checked.

Step 2 →After step 1 the recurrence relation called.

Step 3 →Repeat Step 1 and 2 till the base case fails.

Best Example of Recursion: -



Suppose you are at source and wants to reach your destination and at a single time you can move only one single step means when source value is at 0 that means you have to move 10 steps to reach your destination and when source value is 5 then you have to move 5 steps to reach your destination. This example can easily code by the recursion.

```
Void reachdest (int src, int dest) {
// This is the base case of the problem
    If(src==dest) {
        Cout<<" Arrived at destination" <<endl;
    }
// This is the recurrence relation
    Reachdest (src+=1, dest);
}
```

Note: - In recursion break the problem into smaller ones and solve any one smaller problem, rest of the problems will be solved automatically.

Some Programs which can be solved using recursion are listed below:

1. Printing contiguous sequence of numbers using recursion.
2. Fibonacci series using recursion.
3. Factorial of a given number.
4. Convert digits into words like 12 => one two.
5. Reach distance.
6. Power of a number problem.

7. Nth stairs problem. (In this problem we have to find out the number of ways a person reach the Nth stairs by climbing one stairs or two stairs)