Recursion

"In order to understand recursion, one must first understand recursion."

In computer science, recursion is a method used to solve a problem where the solution of the problem depends upon the solution to the smaller instances of the same problem.

The action in which a subroutine calls itself (directly or indirectly) is called recursion and the subroutine that depends on recursion in its life is called a recursive function.

To have a better understanding, let us first establish a relation between mathematical aspects that bind with recursion:

Just as we write a function which gives a squared of the input value as output in the following way in algebra $f(x) = x^2$.

In programming, we write the code as

```
public static int fun(int x)
{
    return x*x;
}
```

The above code is just an implementation of the algebraic function written before it.

"In programming language, the tool for expressing algebraic expressions is called a function."

Now the foundation of this uniformity is laid down by the ***Principles of Mathematical Induction.***

Taking an example for better comprehension, below is the well-known formula for summation of the first *n*-natural numbers and we prove it using **PMI**:

$$\sum_{i=1}^{n} i = \frac{n\,(n+1)}{2}$$

Step 1: We assume the formula is true for ***n = k***: …(1)

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

Step 2: We prove the formula holds true for ***n = k+1***: …(2)

$$\sum_{i=1}^{k+1} i = \frac{(k+1)\,(k+2)}{2}$$

Proof:

L.H.S = $\sum_{i=1}^{k+1} i = 1 + 2 + 3 + \ldots. + k + (k+1)$

Now , the sum of the first k terms can be written as (1)

So , equation can be written as

$$\sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k+1)$$

…(3)

$$\sum_{i=1}^{k+1} i = \frac{k(k+1)}{2} + (k+1)$$

$$\sum_{i=1}^{k+1} i = \frac{k(k+1)+2(k+1)}{2}$$

$$\sum_{i=1}^{k+1} i = \frac{(k+1)\,(k+2)}{2}$$

Hence, Proved that LHS = RHS and in turn,

The condition is true for ***n = k + 1***

Step 3: We prove it for ***n = 1*** so that it holds true for all natural numbers

$$\sum_{i=1}^{n} i = \frac{n\,(n+1)}{2}$$

Put n = 1 in the above equation in both L.H.S. and R.H.S.

LHS = 1

RHS = $\frac{1(1+1)}{2} = \frac{1(2)}{2} = 1$

Hence, the equation $\sum_{i=1}^{n} i = \frac{n\,(n+1)}{2}$ holds true for all natural numbers.

Now, it is only natural that one would wonder what is the context of this mathematical concept with recursion?

Let us take a sample problem:

```
Question:
-----------------------
1. You are given a positive number n.
2. You are required to print the counting from n to 1.
3. You are not to use any loops. Complete the body of
   the printDecreasing function to achieve it.
4. You have to write a recursive implementation.
```

Given Code:

```
import java.io.*;
import java.util.*;

public static void main(String args) throw Exception {
    //Write your code here
}

public void printDecreasing(int n)
{
    //complete the given function
}
```

Now we apply the *PMI* concept to the problem:

The function printDecreasing(n) is supposed to print the values from n to 1.

printDecreasing is ***expected*** to go about in a fashion resembling the following and printing the pattern:

n .. n-1 .. n-2 … … … 3 .. 2 .. 1. (Provided the given positive number is ***n***)

But, we have a ***faith*** the function printDecreasing(n-1) will execute a pattern like

n-1.. n-2.. … … 3 .. 2 .. 1.

At this moment, take a good look at the conditions that we have established.

Isn't it similar to the case of assuming n = k ***and then proving*** n = k+1 ***case ?***

We can write the printDecreasing() function for parameter n as a function of printDecreasing() with parameter n – 1

printDecreasing(n) = printDecreasing(n-1) + n

which is relatable with the ***equation(3)*** in the ***PMI*** *example*.

When we bring this down to programming implementation, the operation that we perform is that the function calls itself with a smaller parameter within its function definition and this is called **recursion** and the performing function is called a **recursive function.**

The above process of defining ***expectation & faith*** is called **High Level Thinking** of recursion.

It involves of **3 steps:**

1. Establish the expectation.
2. Establish the faith.
3. Link the above two (expectation and faith).

Now, writing the code till the process of **High Level Thinking**:

```
public void printDecreasing(int n) {
        System.out.println(n);
        printDecreasing(n-1);
}
```

But there is clear visible shortcoming to this approach:**There is no end to it!**

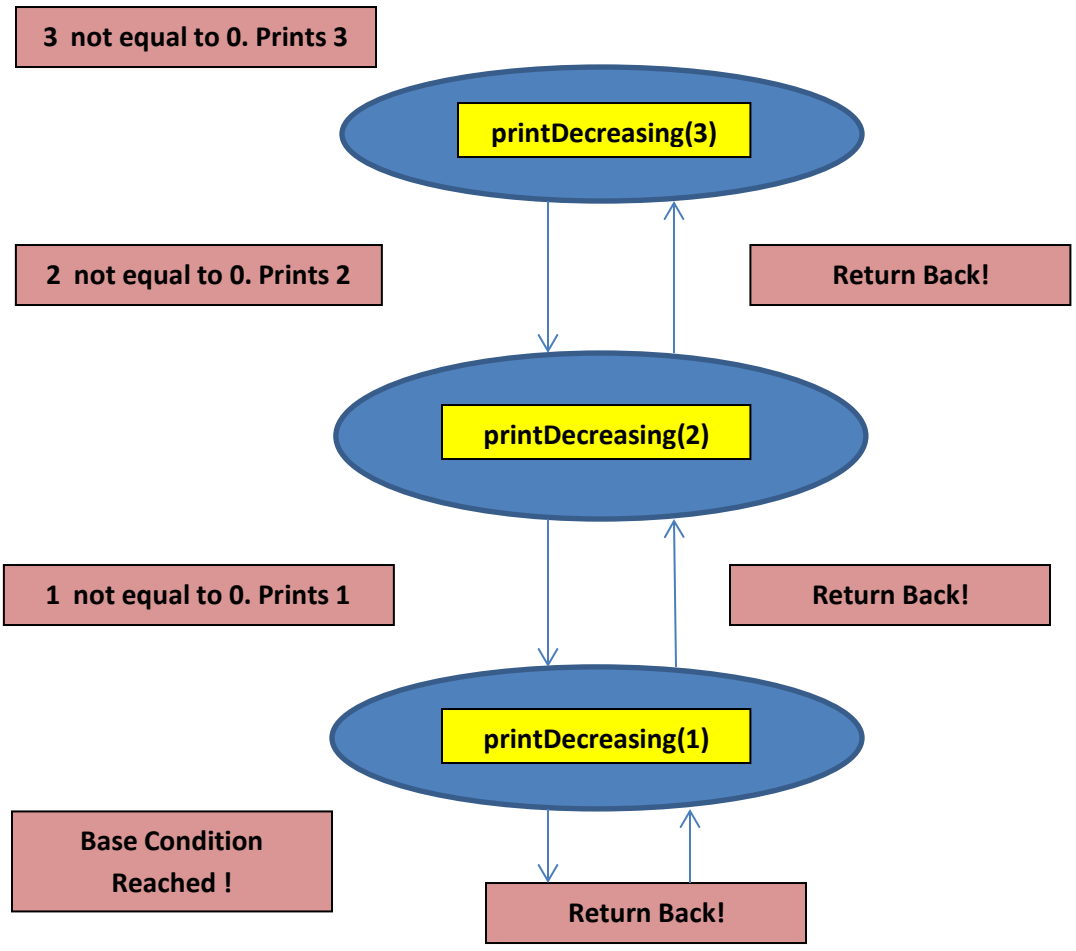
This is where we employ what is called the **Low Level Thinking** in recursive procedure programming:

We add a base case to it so the Call stack does not overflow.

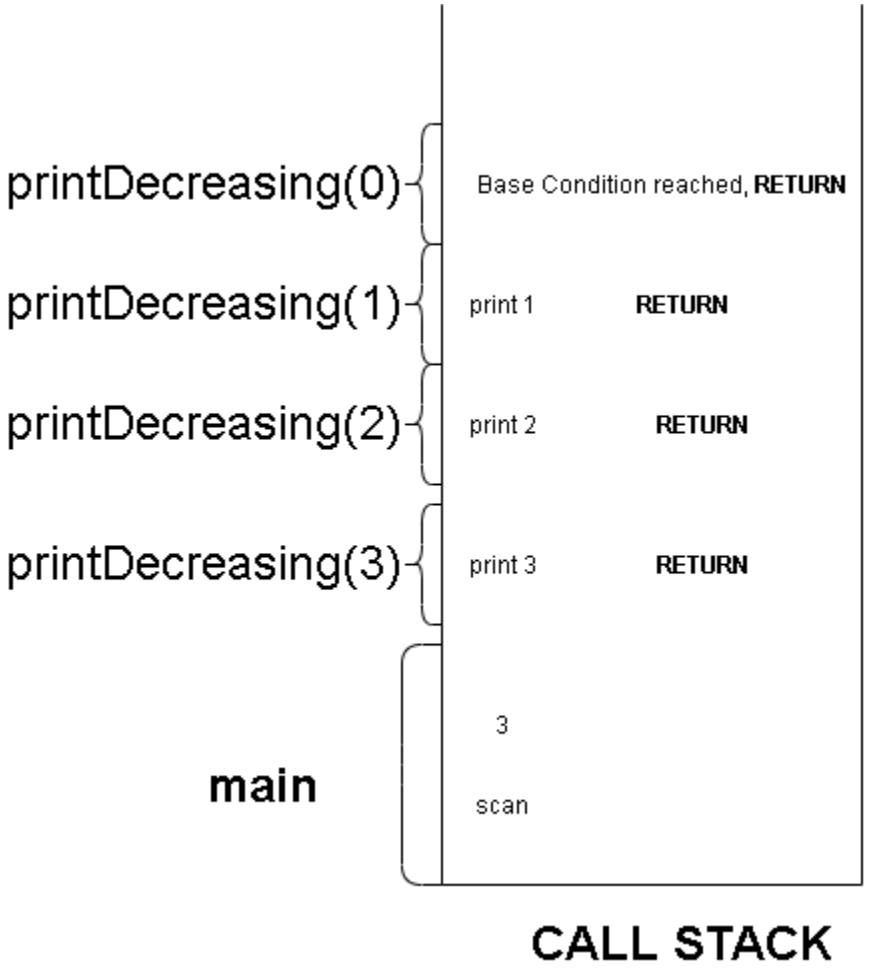
```
public void printDecreasing(int n) {
        if(n==0)    //base Case (LOW LEVEL THINKING)
        {
            return;     //Wipes the current executing function
                        //from the Call stack
        }
        System.out.println(n);
        printDecreasing(n-1);   //recursive call
                                //(HIGH LEVEL THINKING)
}
```

For example of working of this function, let us take **n = 3**

The operations that take place upon call invocation of **printDecreasing(3)** are:



The working of the call stack is as follows:



The function gives us the output

```
3
2
1
```

This is an example of a direct - tailed recursion.

Now what this statements means, we divide into two parts:

Direct and Indirect:

In direct recursion, the function calls itself for the purpose of recursion.

```
//DIRECT RECURSION
public void fun()
{
    fun();
}
```

In indirect recursion, the function may call another function which in turn again calls the first calling function.

```
//INDIRECT RECURSION
public void fun()
{
    gun();
}
public void gun()
{
    fun();
}
```

Tailed and non-tailed:

In tailed recursion, the base condition is first checked and the recursive call is the last thing executed by the function.

In non-tailed recursion, the base condition is checked later and the recursive call is the first thing executed by the function.

"***For understanding recursion, we must go one level deeper!***

Then?"