# Recursion

→ Process in which function calls itself. That function is called recursive function.

## 1) Mathematical representation :

→ Let function $f(x)$ be recursive function. Then;

$$f(x) = \begin{cases} 1 ; & x = 1 \\ x + f(x-1) ; & x \geq 1 \end{cases}$$

→ Always include base case condition at the start of function.

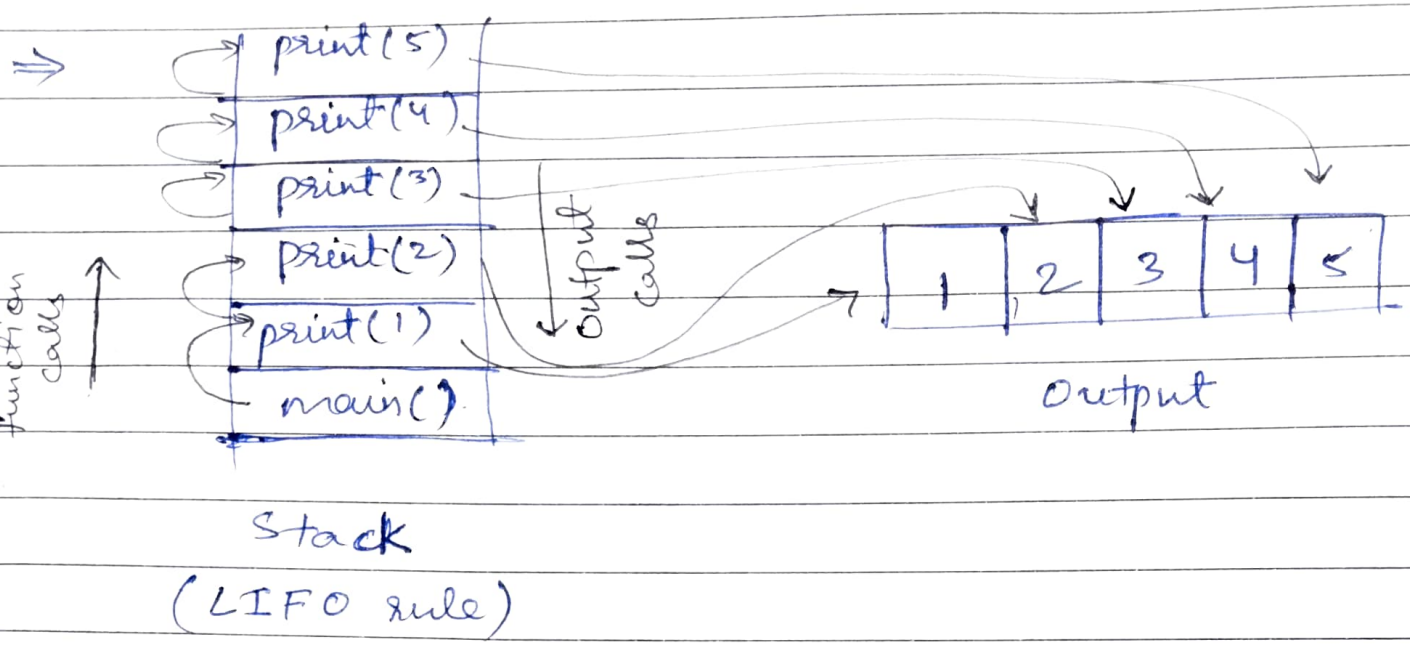
## 2) Internal working :

→ Let's print 1 to 5, without multiple calling of functions or o/p statements.

⇒

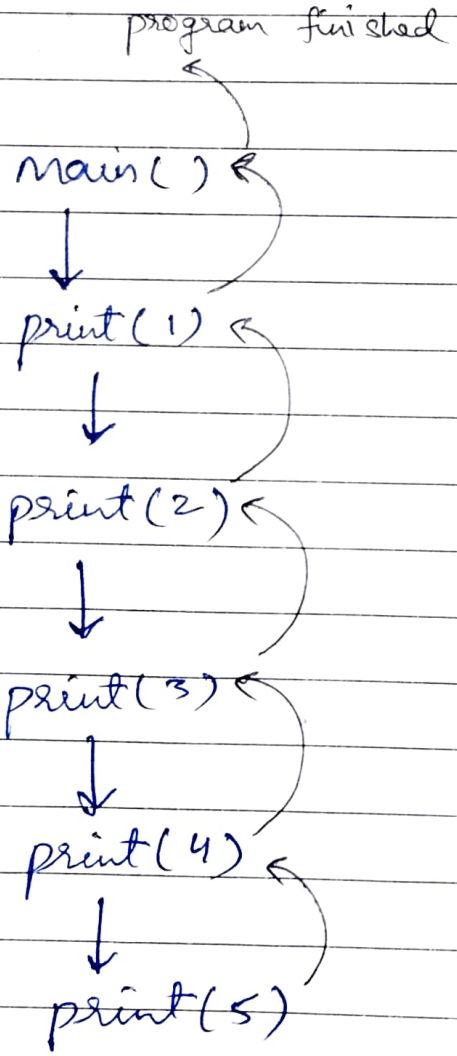
```
void print (int n)
{
  if(n==5)
  {
    cout<<n;
    return;
  }
  cout<<n;
  print(n+1);
}
```

```
void main()
{
  cout << print(1);
}
```




Stack
(LIFO rule)

3) Recursive Tree :



4) Stackoverflow condition :

→ If the base case is never reached, then stack overflow occurs.

Ex:
```
int fact(int n)
{
  if(n==50)
    return 1;
  else
    return n* fact(n-1);
}
```

→ So, for n=10, base case is never reached.

5) Tail recursion :

→ Recursive function is tail recursive when a recursive call is last thing executed by function.

→ It has an advantage in terms of stack usage.

6) Recurrence relations :

→ For factorial : $f(n) = n * f(n-1)$

→ For fibonacci : $f(n) = f(n-1) + f(n-2)$

Date
Page no.
zaneBlitz

7) Steps to solve problems :

→ Identify if you can break problem into smaller ones.

→ Try to form recurrence relation.

→ Draw recursive tree

→ Understand functions through tree by forming left & right branches.

→ Observe what values are returned at each step.