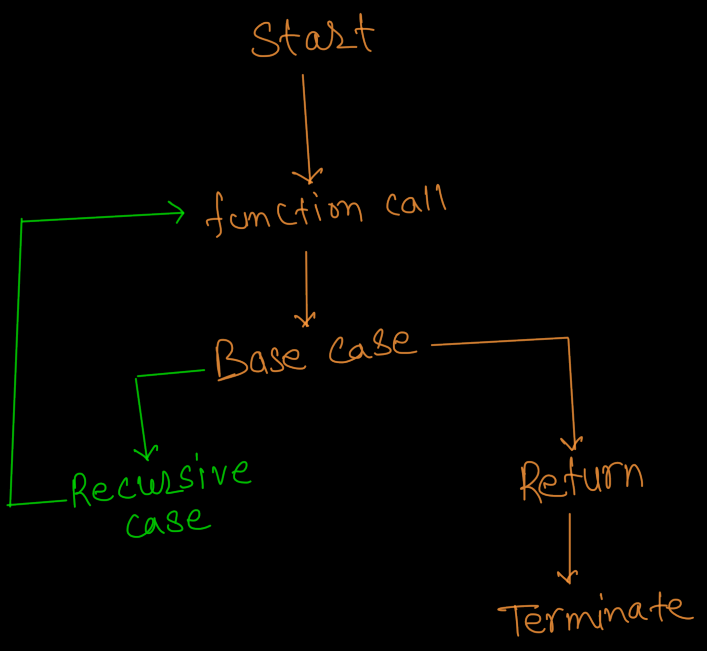
Recursion

→ Describing an action interm of itself.

Why : Task that are composed of similar substack.
- Shorter code (may be easier)
- Nested loops can be avoided

```
function recursion() {
  // base case
  if (anyBaseCase()){
    return;
  }
  // recursive case
  recursion();
```



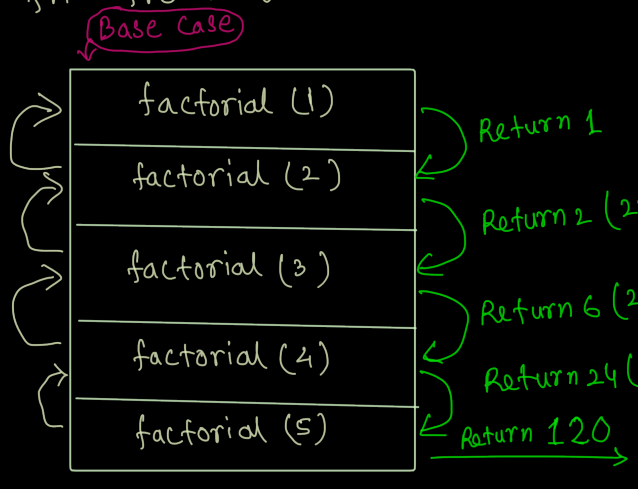
* Memory in Recursion *

→ A recursive fn call itself.

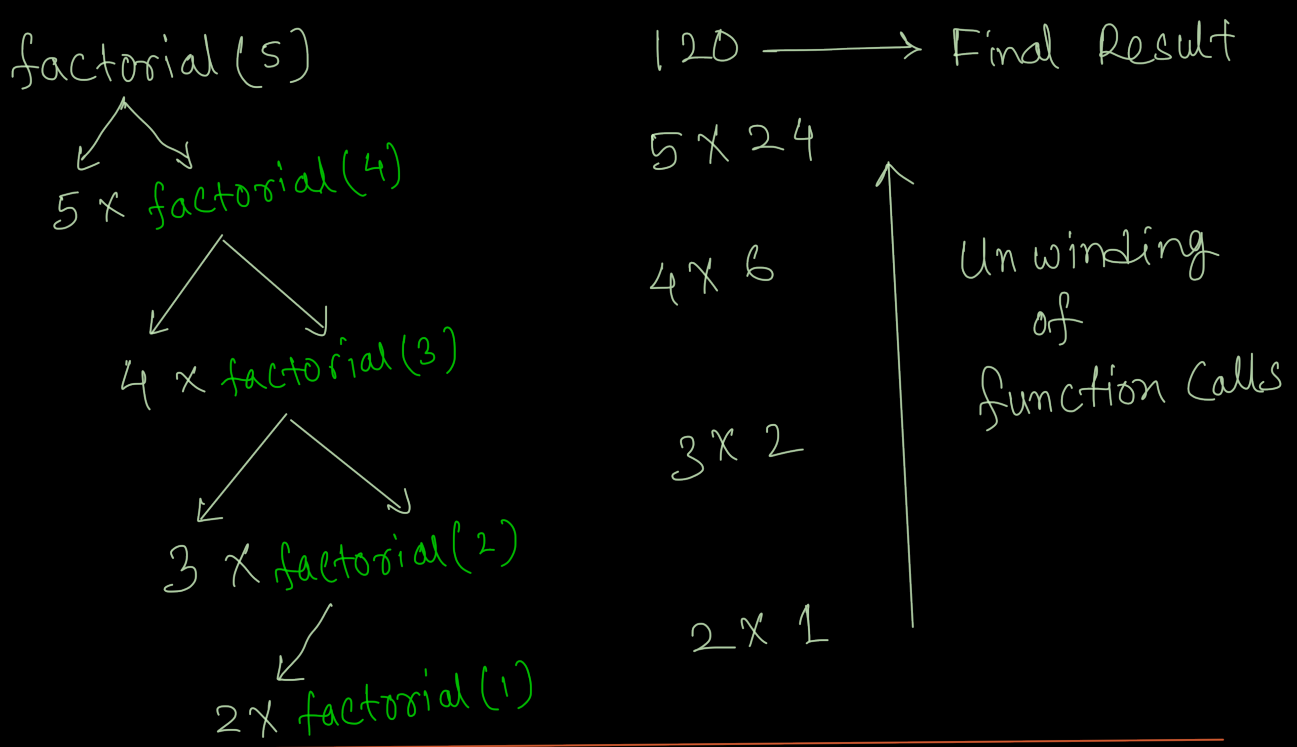
↳ The memory for a called fn is allocated on top of memory allocated for called fn.

→ Each fn call gets different copy of local variables.

→ When base case is reached, child fn returns value to the fn from which it was called, and the process continues.




```
function factorial(n){
  if(n===0){
    return 1;
  }
  return n * factorial(n-1);
}
console.log(factorial(5));
```



Direct Recursion

→ when fn call itself

```
1  function printNumber(min, max) {
2      if (min > max) {
3          return;
4      }
5      printNumber(min + 1, max);
6      console.log(min);
7  }
8
9  printNumber(1, 5);
10
```

Print NO. → 5, 4, 3, 2, 1

```
function printNumber(min, max) {
    if (min > max) {
        return;
    }
    console.log(min);
    printNumber(min + 1, max);
}

printNumber(1, 5);
```

Print NO. → 1, 2, 3, 4, 5

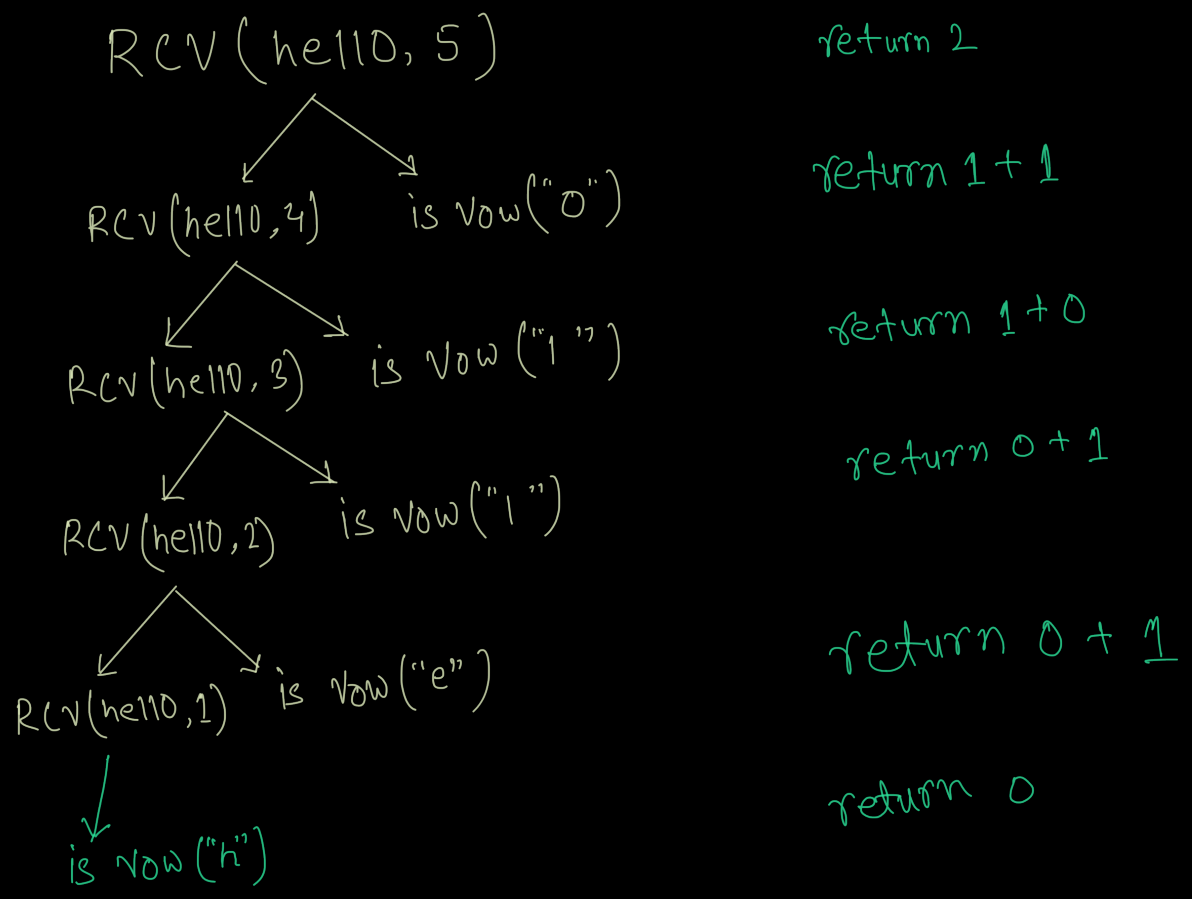
PN (6,5) → Return
PN (5,5) → Print 5
PN (4,5) → Print 4
PN (3,5) → Print 3
PN (2,5) → Print 2
PN (1,5) → Print 1

Indirect Recursion

```
function printNumber(min, max) {
    if (min > max) {
        return;
    }
    console.log(min);
    min = min + 1;

    logic(min, max);
}

function logic(min, max) {
    if (min > max) {
        return;
    }
    printNumber(min, max);
}

printNumber(1, 5);
```

→ Logic is extended

→ Logic is calling PrintNumber again & again.



Fibonacci

0, 1

0, 1, 1

0, 1, 1

0, 1, 1, 2

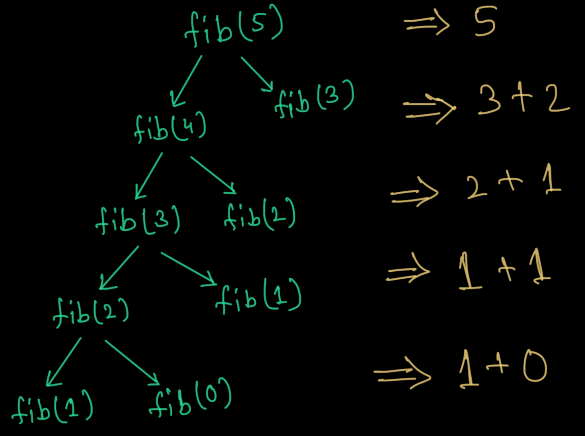
0, 1, 1, 2, 3

0, 1, 1, 2, 3, 5

0, 1, 1, 2, 3, 5, 8

$F_n = F_{n-2} + F_{n-1}$

```
n0 = 0
n1 = 1
temp = n0 + n1
⇓
n0 = n1
n1 = temp
```



Power of a number

P(2,4) ⇒ 16 → 2×2×2×2

2 × P(2,3) ⇒ 8×2

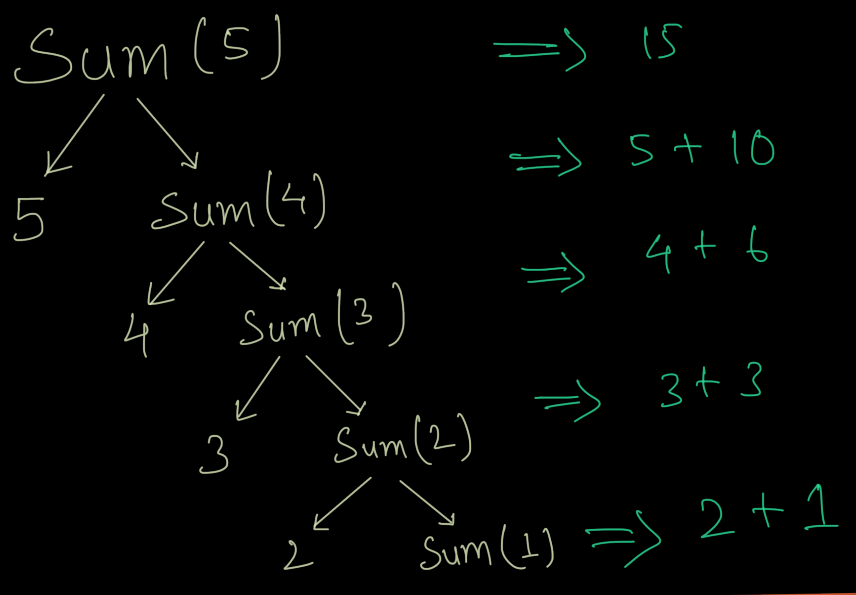
2 × P(2,2) ⇒ 4×2

2 × P(2,1) ⇒ 2×2

2 × P(2,0) ⇒ 2×1

Exponent == 0
↳ Returns 1

Sum of 1 to n numbers



Greatest Common Divisor

42 → 1, 2, 3, 6, 7, 14, 21, 42

56 → 1, 2, 4, 7, 8, 14, 28, 56

If m == n

$$GCD = \begin{cases} m \\ gcd(m-n, n) \text{ if } m > n \\ gcd(m, n-m) \text{ if } m < n \end{cases}$$

Remove Adjacent Duplicate