What is Recursion ?

→ When a function calls itself directly or indirectly, the process is called recursion and this call is called a recursive call.

Eg:
```
int fun(int n){
    fun(n);
}
```

If the solution of a problem depends on a smaller problem (subproblem) of the same type, then we will use recursion.

Eg: Find 2^n.

We can say that $2^n = \underbrace{2 \times 2 \times 2 \times \ldots \times 2}_{n \text{ times}}$

$\Rightarrow 2^n = 2 \times 2^{n-1}$

If we make a function that gives us 2^m when it's called like :

```
fun(m);
```

Then `fun(n) = 2 * fun(n-1);` ← Recurrence Relation

Eg: Find factorial (n!)

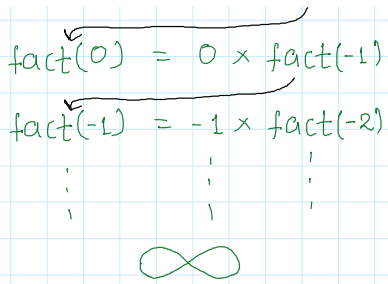
We know that $5! = 5 \times 4!$

$\Rightarrow fact(5) = 5 \times fact(4)$

$\Rightarrow fact(n) = n * fact(n-1)$

Given that we stop at a condition, example $0! = 1$, this is called base condition. Without this, our function will go bonkers !

$fact(3) = 3 \times fact(2)$
$fact(2) = 2 \times fact(1)$
$fact(1) = 1 \times fact(0)$



Step 1: Specify a base case where we return.

Step 2: Solve for the bigger problem using the smaller problem.

```cpp
#include<iostream>
using namespace std;

int factorial(int n) {

    //base case
    if(n == 0)
        return 1;

    return n * factorial(n-1);
}

int main() {

    int n;
    cin >> n;

    int ans = factorial(n);

    cout << ans << endl;

    return 0;
```

```
Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/" && g++ factroial.cpp -o factroial && "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"factroial
5
120
lovebabbar@192 Lecture31: Recursion Day1 % cd "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/" && g++ factroial.cpp -o factroial && "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"factroial
4
24
lovebabbar@192 Lecture31: Recursion Day1 % cd "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/" && g++ factroial.cpp -o factroial && "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"factroial
6
720
lovebabbar@192 Lecture31: Recursion Day1 %
```

Without a base case, we will get 'Segmentation Fault' which means you have exhausted your function call stack (the memory where function calls are accumulated when called).

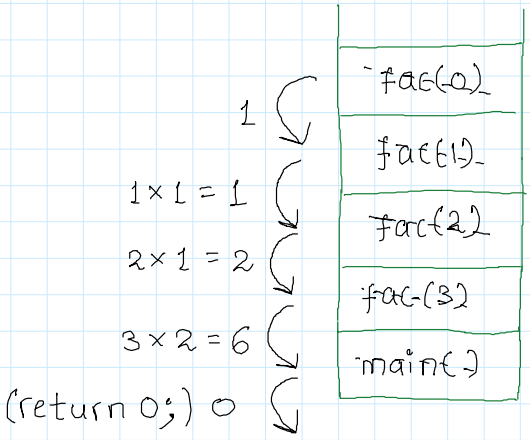
```cpp
int factorial(int n) {

    //base case


    return n * factorial(n-1);
}
```

```
6
zsh: segmentation fault
lovebabbar@192 Lecture31: Recursion Day1 %
```

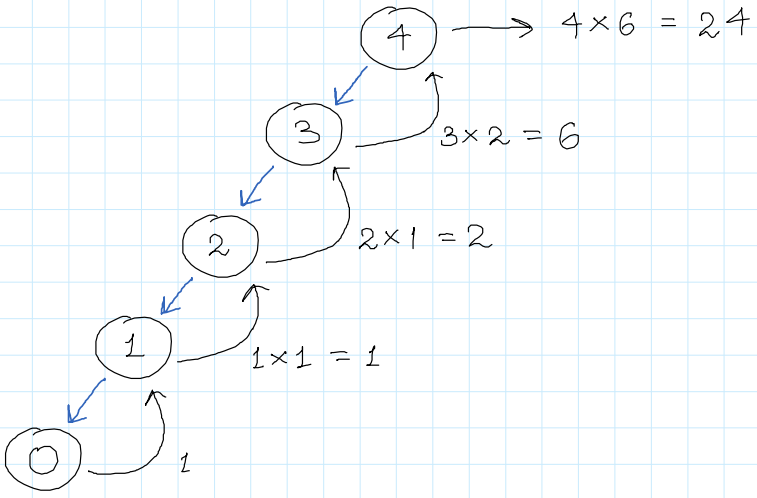
For fac(3), function call stack will look like this:



If we don't create a base case then, it will continue going further to fact(-1), fact(-2), fact(-3) and so on until our memory gets exhausted. This is called segmentation fault.

Recursion Tree :

Let's make a recursion tree for fac(4) where n = 4.



Final Structure of a Recursive Function :

```
Fun ( ) {
    Base Case
    Processing
    Recurrence Relation
}
```

Tail Recursion

```
Fun ( ) {
    Base Case
    Recurrence Relation
    Processing
}
```

Head Recursion

```cpp
int power(int n) {

    //base case
    if(n == 0)
        return 1;

    //recursive relation
    int smallerProblem = power(n-1);
    int biggerProblem = 2 * smallerProblem;

    return biggerProblem;
}
```

```
wer
10
1024
lovebabbar@192 Lecture31: Recursion Day1 % cd "/Users/lovebab
bar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion
 Day1/" && g++ power.cpp -o power && "/Users/lovebabbar/Docum
ents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"po
wer
5
32
lovebabbar@192 Lecture31: Recursion Day1 %
```

OR

```cpp
int power(int n) {

    //base case
    if(n == 0)
        return 1;

    //recursive relation
    return 2 * power(n-1);
}
```

```
10
1024
lovebabbar@192 Lecture31: Recursion Day1 % cd "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/" && g++ power.cpp -o power && "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"power
5
32
lovebabbar@192 Lecture31: Recursion Day1 % cd "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/" && g++ power.cpp -o power && "/Users/lovebabbar/Documents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"power
6
64
lovebabbar@192 Lecture31: Recursion Day1 %
```

Example: Print n to 1.

Approach: Base case is when n = 0, else we will print the number and then call for n-1.

```cpp
void print(int n) {
    //base case
    if(n == 0) {
        return ;
    }

    cout << n << endl;

    print(n-1);
}
```

```
5
5
4
3
2
1
lovebabbar@192 Lecture30: Macros, Gloab Variable etc. %
```

(Tail Recursion) → n to 1.

```cpp
void print(int n) {
    //base case
    if(n == 0) {
        return ;
    }

    //Recursive relation
    print(n-1);

    cout << n << endl;
}
```

```
5
1
2
3
4
5
lovebabbar@192 Lecture30: Macros, Gloab Variable etc. %
```

(Head Recursion) → 1 to n.

Recursion tree for 1 to n. (n = 3)