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 × 4!

$\Rightarrow$ fact(5) = 5 × 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 × fact(2)
fact(2) = 2 × fact(1)
fact(1) = 1 × fact(0)

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

∞

Step 1: Specify a base case where we return.

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

```cpp
1   #include<iostream>
2   using namespace std;
3
4   int factorial(int n) {
5
6       //base case
7       if(n == 0)
8           return 1;
9
10      return n * factorial(n-1);
11  }
12
13  int main() {
14
15      int n;
16      cin >> n;
17
18      int ans = factorial(n);
19
20      cout << ans << endl;
21
22      return 0;
```

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);
}
```
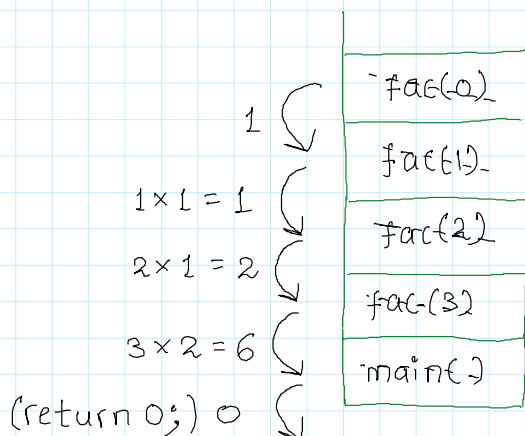
```
6
zsh: segmentation fault
lovebabbar@192 Lecture31: Recursion Day1 %
```
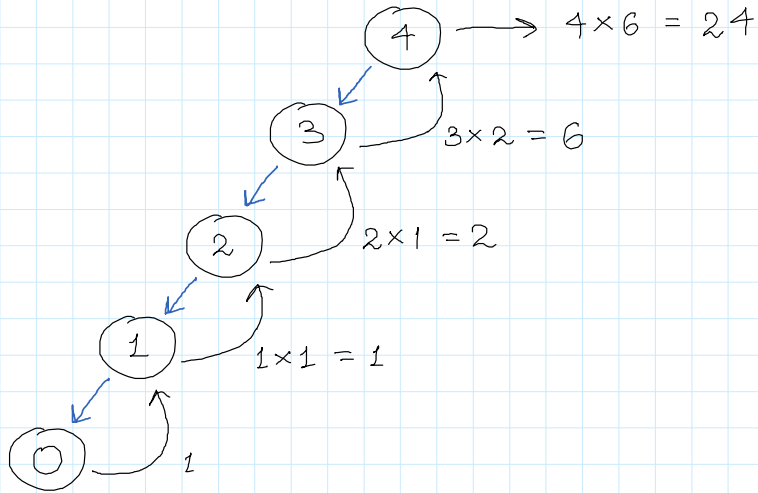
For fac(3), function call stack will look like this:

fac(0)
fac(1)
fac(2)
fac(3)
main()

1

$1 \times 1 = 1$

$2 \times 1 = 2$

$3 \times 2 = 6$

(return 0;) 0

Recursion Tree :

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

$$4 \quad \longrightarrow \quad 4 \times 6 = 24$$

$$3 \quad \quad 3 \times 2 = 6$$

$$2 \quad \quad 2 \times 1 = 2$$

$$1 \quad \quad 1 \times 1 = 1$$

$$0 \quad \quad 1$$

Final Structure of a Recursive Function :

Fun ( ) {

  Base Case

  Processing

  Recurrence Relation

}

Tail Recursion

Fun ( ) {

  Base Case

  Recurrence Relation

  Processing

}

Head Recursion

```cpp
3
4    int power(int n) {
5
6        //base case
7        if(n == 0)
8            return 1;
9
10       //recursive relation
11       int smallerProblem = power(n-1);
12       int biggerProblem = 2 * smallerProblem;
13
14       return biggerProblem;
15   }
```

```
ents/CodeHelp-DSA-Busted-Series/Lecture31: Recursion Day1/"po
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

```
4    int power(int n) {
5
6        //base case
7        if(n == 0)
8            return 1;
9
10       //recursive relation
11       return 2 * power(n-1);
12   }
13
```

```
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 % 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
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.

```
4    void print(int n) {
5        //base case
6        if(n == 0) {
7            return ;
8        }
9
10       cout << n << endl;
11
12       print(n-1);
13
14   }
```

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

(Tail Recursion) → n to 1.

```
4    void print(int n) {
5        //base case
6        if(n == 0) {
7            return ;
8        }
9
10       //Recursive relation
11       print(n-1);
12
13       cout << n << endl;
14   }
```

```
es/Lecture30: Macros, Gloab Variable etc./ counting
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)

Finish ← ③ ← returns after printing 2.
Returns after printing 3.
② ← returns after printing 1
① ← return base condition
⓪