



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 $2^n = \underbrace{2 \times 2 \times 2 \times \dots \times 2}_{n \text{ times}}$

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

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

`fun(m);`

Then $\text{fun}(n) = 2 * \text{fun}(n-1);$ ← Recurrence Relation

Eg: Find factorial. ($n!$)

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

$$\Rightarrow \text{fact}(5) = 5 \times \text{fact}(4)$$

$$\Rightarrow \text{fact}(n) = n * \text{fact}(n-1)$$

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

$$\text{fact}(3) = 3 \times \text{fact}(2)$$

$$\text{fact}(2) = 2 \times \text{fact}(1)$$

$$\text{fact}(1) = 1 \times \text{fact}(0)$$

$$\begin{aligned}
 \text{fact}(0) &= 0 \times \text{fact}(-1) \\
 \text{fact}(-1) &= -1 \times \text{fact}(-2) \\
 &\vdots \\
 &\vdots \\
 &\infty
 \end{aligned}$$

Step 1: Specify a base case where we return.

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

```

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).

```

int factorial(int n) {
    //base case
    return n * factorial(n-1);
}

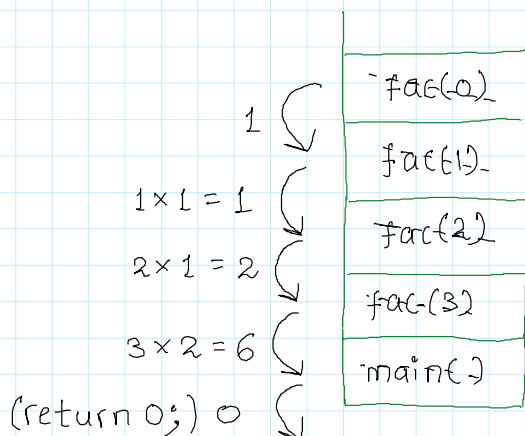
```

```

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

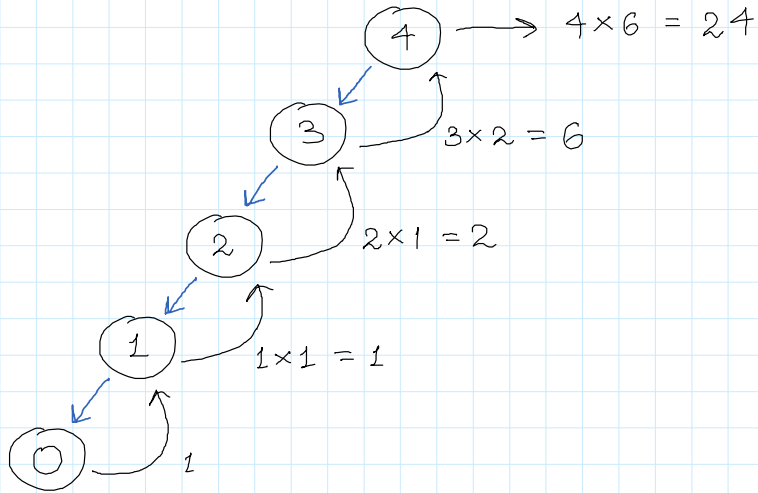
```

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



Recursion Tree :

Let's make a recursion tree for $\text{fac}(4)$ 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

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

```
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 %
```

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/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 $n-1$.

```

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

```

(Tail Recursion) \rightarrow 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, Global Variable etc./ counting
5
1
2
3
4
5
lovebabbar@192 Lecture30: Macros, Global Variable etc. %

```

(Head Recursion) \rightarrow 1 to n.

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

