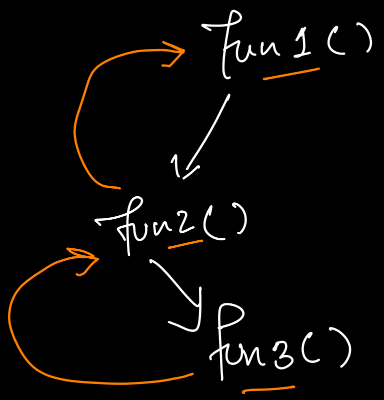
# Recursion:

```
void recur()
{
    = }   ↓
    recur();
}
```

'Infinite loop.



→ Base Condition (Solution) : Simplest problem"

→ Stop

→ Complex Problem ⇒ simplify → function

```
int factorial(int n)
{
    if (n == 0)  ✓ //Base Condition
        return 1;
                     simplify
                       ↓
    return n x factorial(n-1);
}
```

0! = 1

5! = 5 x 4!   ✓ 24 = 120

4! = 4 x 3!   6 = 24

3! = 3 x 2!   2 = 6

2! = 2 x 1!   1

1! = 1 x 0!   1

↑

n! = n x (n-1)!

↓

n x (n-1)!

n

Factorial(4)

return 4 x factorial(3)   6 = 24

return 3 x factorial(2)   2 = 6

return 2 x factorial(1)   1 = 2

return 1 x factorial(0)   1 = 1

```
                  2         1          5
void tableOf( int val, int start, int till)
{                6  5
    if (start > till) //Base Condition
→       return; ←

    cout<< val <<" * "<<start<<" = "<< val * start;
    cout<<endl;
    table of(val, start+1, till);
}
```

result

→ 2 X 1 = 2 ←

→ 2 X 2 = 4 ←

→ 2 X 3 = 6 ←

→ 2 X 4 = 8 ←

2 X 5 = 10 ←

↑

# Fibonacci Sequence:

Base Condition

| 0 | 1 | 1 | 2 | 3 | 5 | - - - - |
|---|---|---|---|---|---|---|
| fib(0) | fib(1) | fib(2) | fib(3) | fib(4) | fib(5) | |

fib(5)

fib(4) + fib(3)

$$fib(n) = fib(n-1) + fib(n-2)$$

```
int fib(int n)
{
  if(n<=1)
      return n;
  else
      return fib(n-1)+fib(n-2);
}
```

```
fib(4)            2        +          1 = 3
ret fib(3)+fib(2)          ---->  ret fib(1) + fib(0)
      1    +   1 = 2                   1   +   0  = 1
ret fib(2) + fib(1)
      1    +   0 = 1
ret fib(1) + fib(0)
```

Slow

fib(0) - 2 times

fib(1) - 3 times

fib(2) = 2 times