Recursion - 1

→ function calling itself

→ Problem is broken down into smaller sub-problems & using the solution of smaller subproblems, main problem is solved.

Question

find sum of first N natural no. using recursion.

$sum(N) = 1 + 2 + 3 + \dots + N-1 + N$

$sum(5) = \underbrace{\boxed{1 + 2 + 3 + 4}}_{sum(4)} + 5$

$sum(5) = sum(4) + 5$

$$\boxed{sum(N) = sum(N-1) + N}$$

↳ $sum(N-2) + N-1$

⋮

Steps to write recursive code

1. Define the problem / function

   sum(N) = 1 + 2 + 3 + ... + N

2. Define recursive relation

   sum(N) = sum(N-1) + N

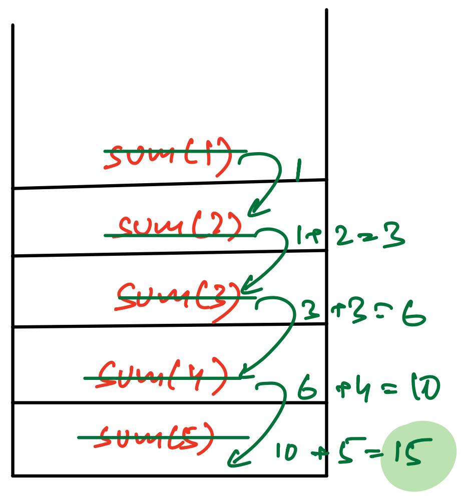
3. Define base case //simplest version of the problem

   sum(1) = 1

Code

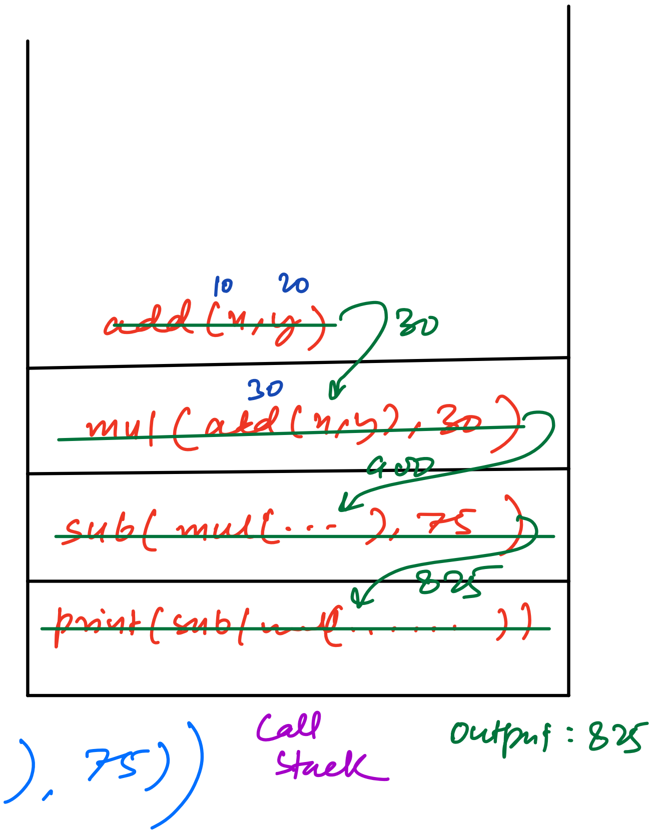
```
int sum(N) {
    //Base case
    if(N==1) return 1

    return sum(N-1) + N

}
```

sum(5)



function call tracing

```
int add( int x , int y){
    return x+y
}
int mul( int x, int y){
    return x* y
}
int sub( int x , int y){
    return x-y
}
main() {
    x=10, y=20
    print( sub( mul( add (x,y), 30), 75))
}
```



Question

Given a +ive integer N, find factorial of N.

N=3 fact(3) = 1×2×3 = 6

N=5 fact(5) = 1×2×3×4×5 = 120

Step 1

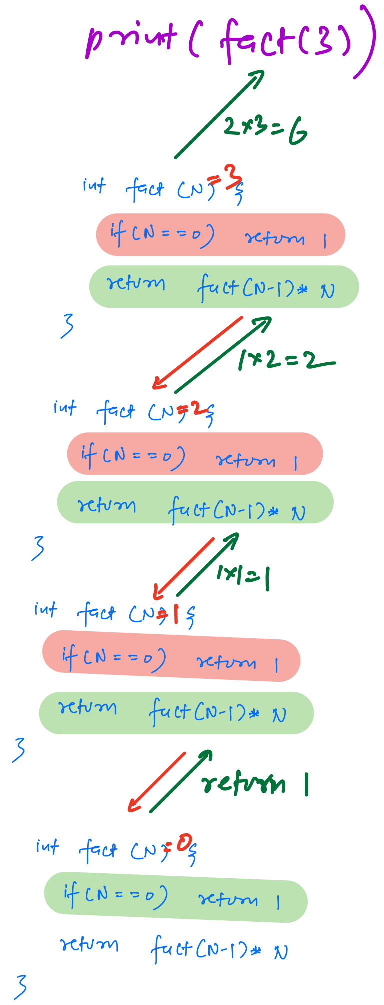
fact(N) = 1×2×3 × . . . . ×N

Step 2

fact(N) = fact(N-1) * N

Step 3

fact(0) = 1



```
int fact (N=3) {
    if (N==0) return 1
    return fact(N-1)* N
}
int fact (N=2) {
    if (N==0) return 1
    return fact(N-1)* N
}
int fact (N=1) {
    if (N==0) return 1
    return fact(N-1)* N
}
int fact (N=0) {
    if (N==0) return 1
    return fact(N-1)* N
}
```

code

```
int fact (N) {
    if (N==0) return 1
    return fact(N-1)* N
}
```

Question

Given N, print numbers from 1 to N in increasing order.

Inc(5) = 1 2 3 4 5

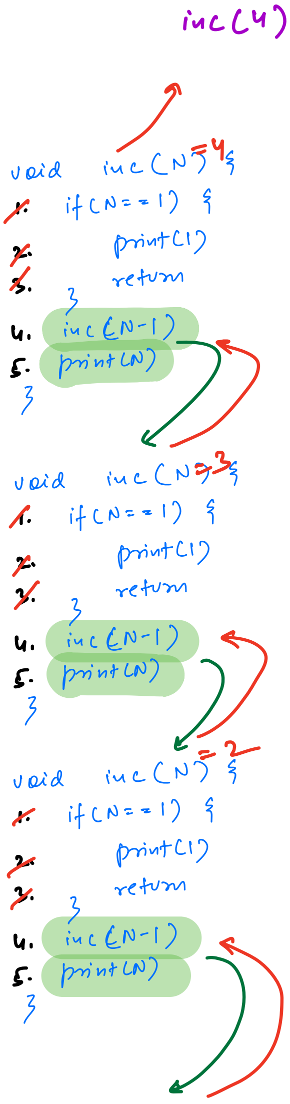
Inc(5) = Inc(4) 5

Inc(N) = Inc(N-1) N

Inc(1) = 1

Code

```
void inc(N) {
    if(N==1) {
        print(1)
        return
    }
    inc(N-1)
    print(N)
}
```



```
void inc(N=4){
~~1.~~ if(N==1) {
~~2.~~     print(1)
~~3.~~     return
    }
4.  inc(N-1)
5.  print(N)
}

void inc(N=3){
~~1.~~ if(N==1) {
~~2.~~     print(1)
~~3.~~     return
    }
4.  inc(N-1)
5.  print(N)
}

void inc(N=2) {
~~1.~~ if(N==1) {
~~2.~~     print(1)
~~3.~~     return
    }
4.  inc(N-1)
5.  print(N)
}
```

```
void   inc(N){
    if(N==1) {
        print(1)
        return
    }
    inc(N-1)
    print(N)
}
```

output: 1 2 3 4

Question

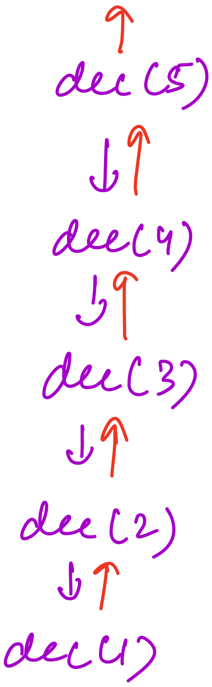
Given N, print numbers from 1 to N in decreasing order.

dec(5) = 5 4 3 2 1

dec(N) = N dec(N-1)

Code

```
void  dec(N) {
        if(N==1) {
            print(1)
            return
        }
        print(N)
        dec(N-1)
}
```



output: 5 4 3 2 1

Time Complexity

O(no. of function calls * TC per function call)

Space Complexity

O(max depth of recursion tree/stack + space per function call)

for factorial

```
int fact(N) {
    if(N==0) return 1
    return fact(N-1)*N
}
```

fact(5)
↓
fact(4)
↓
fact(3)
↓
fact(2)
↓
fact(1)
↓
fact(0)

function calls = N+1

time per call = O(1)

TC = O(N)

max stack size = N

SC = O(N)

for increasing(N) /
decreasing(N)

TC = O(N)   SC = O(N)

Question

Given a +ive number N, find Nth fibonacci no.

N :	0	1	2	3	4	5	6	7
fib(N) =	0	1	1	2	3	5	8	13
						↑ fib(5)	↑ fib(6)	

fib(7) = fib(6) + fib(5)

1. fib(N) → Nth fibonacci no.

2. fib(N) = fib(N-1) + fib(N-2)

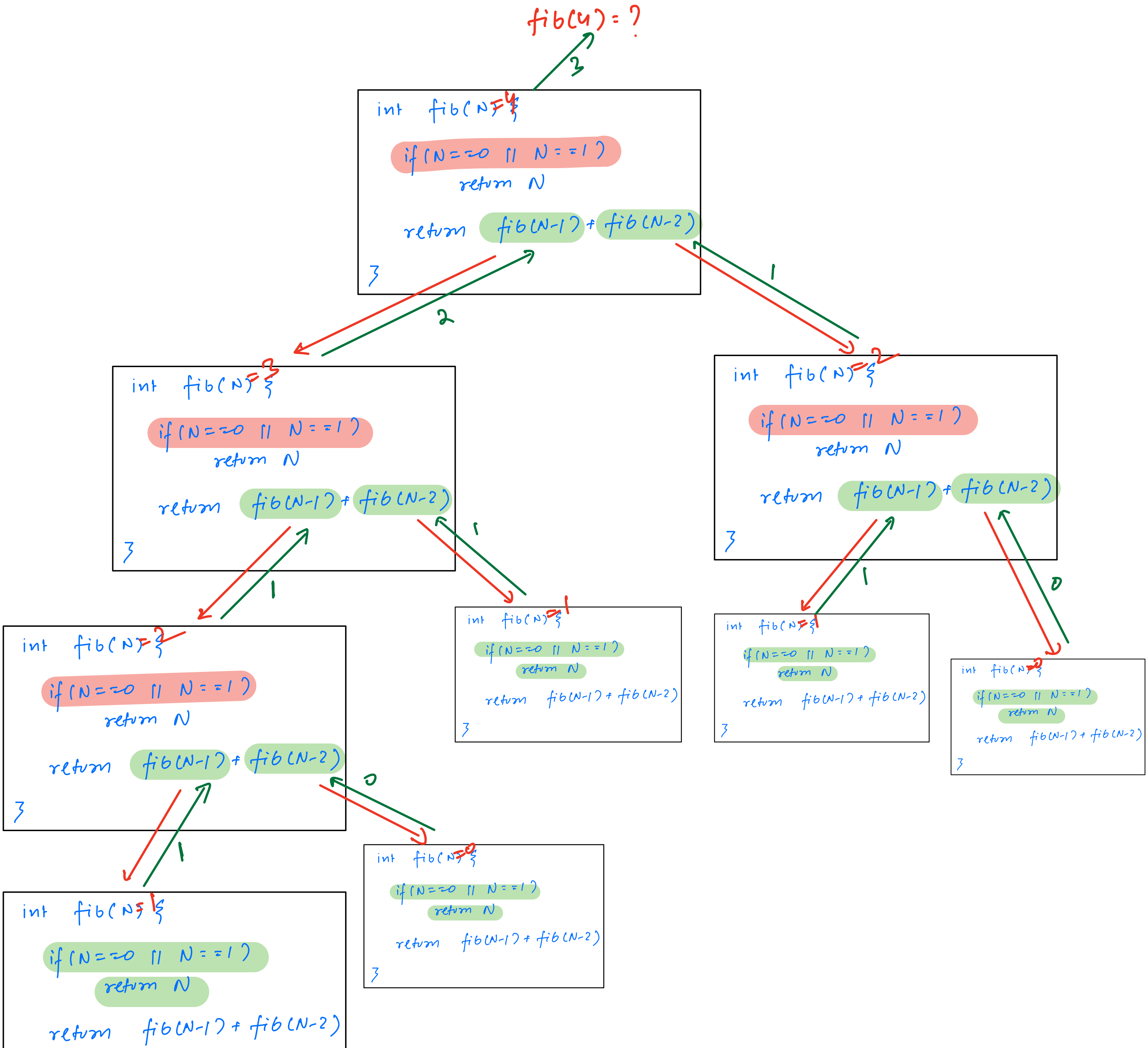
3. fib(0) = 0
   fib(1) = 1

$fib(1) = fib(0) + fib\cancel{(-1)}$

$fib(2) = fib(1) + fib(0)$

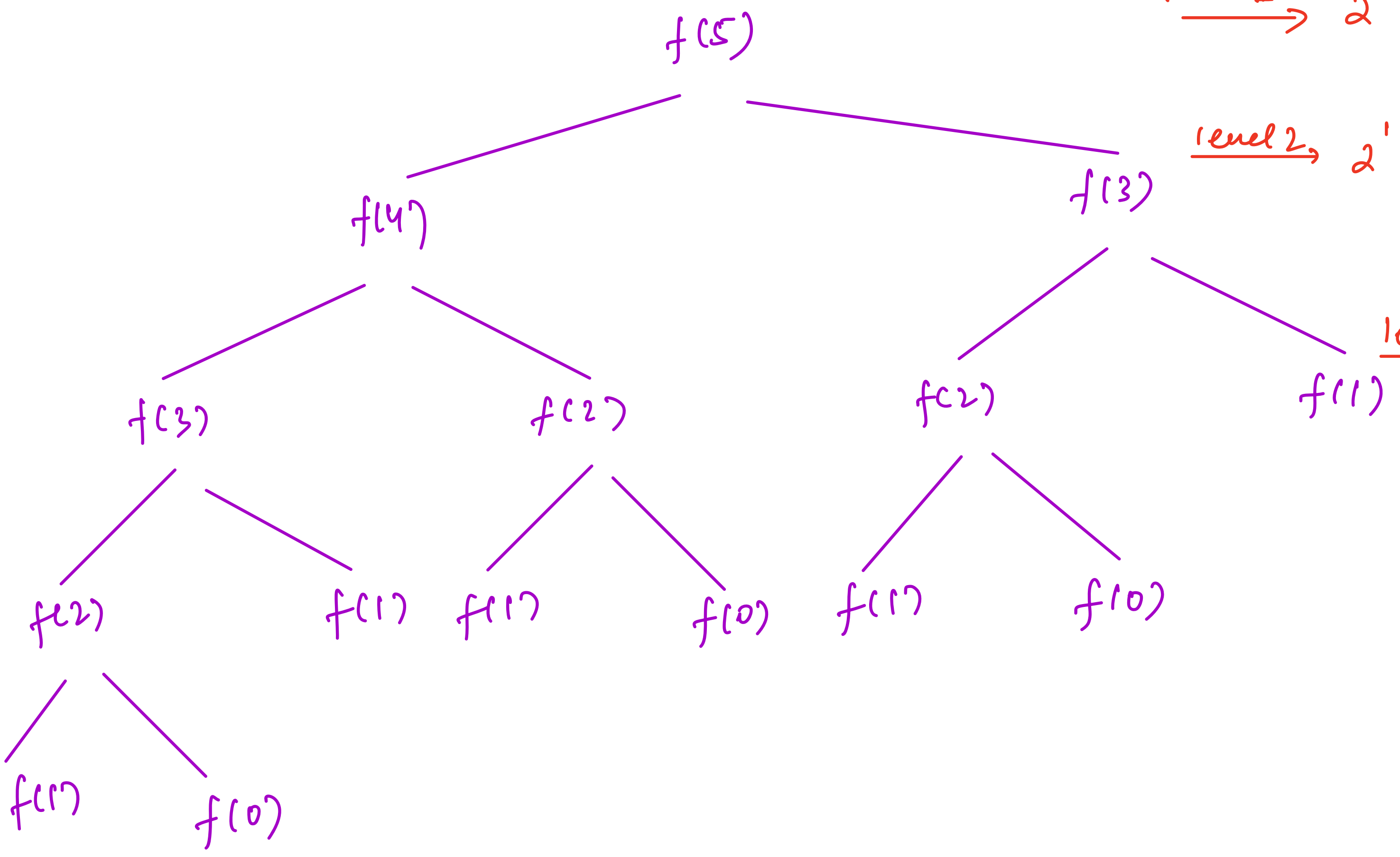
Code

```
int fib(N) {

    if(N==0 || N==1)
        return N

    return fib(N-1) + fib(N-2)

}
```



3

Time complexity



level 1 $\longrightarrow$ 2^0 calls

level 2 $\longrightarrow$ 2^1 calls

level3 $\longrightarrow$ 2^2

there is $\sim 2^x$ calls at each level

total levels = N

total f^n calls $= 2^0 + 2^1 + 2^2 + \; . \; . \; . \; . \; + 2^{N-1}$

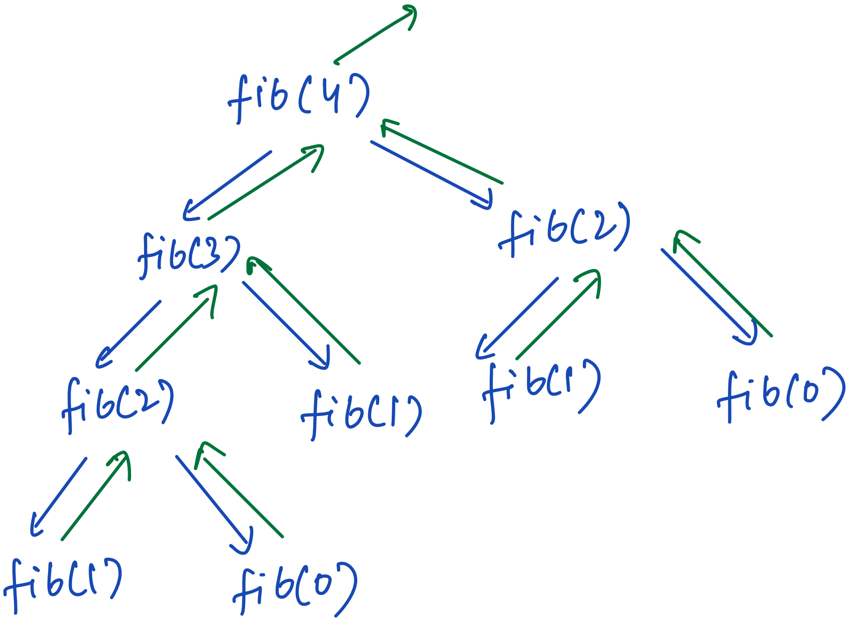
$$= 2^0 \left(\frac{2^N - 1}{2 - 1} \right)$$

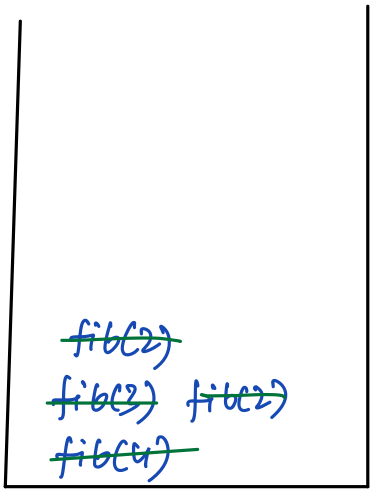
$$= 2^N - 1$$

$\left[\text{sum} = a \left(\frac{r^n - 1}{r - 1} \right) \right.$

$$\boxed{TC = O(2^N)}$$

Space complexity





max size = 3

$SC = O(N)$