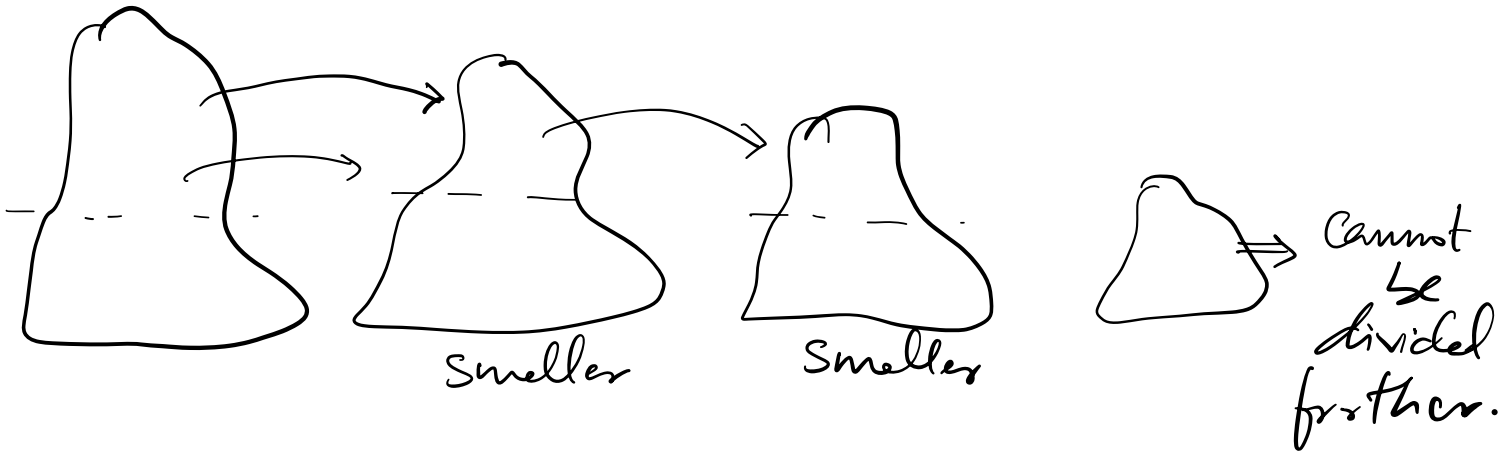
Recursion

(12th)
R1 ⇒ Sunday 11:59 PM
R2 ⇒ Monday (13th) ⇒ 21st Nov
(4 days) R3 ⇒ 22nd Nov ⇒ 5th Jan



Solving a problem using the solution of its subproblems
↳ smaller instance of the problem.

Given an integer N. find the sum of the first N natural no.'s

$Sum(N) = \underbrace{1 + 2 + 3 + 4 \ldots\ldots N-2 + (N-1)}_{} + N$

$Sum(N) = Sum(N-1) + N$

Problem ⇓ Subproblem ⇓

3 Steps to Write Recursive Code

1) **Assumption** : Decide what the function does.

`int Sum(N)` : Calculates & returns the correct sum of the first N natural numbers!

2) **Main Logic.** : Break down the problem into smaller subproblems to solve the assumption.

```
sum(N) = Sum(N-1) + N;
```

3) **Base Case** : The inputs (subproblems) for which we should stop recursion.

```
if (N == 1) { return 1; }
```

```
int sum (N) {
    if (N == 1) {
        return 1;
    }
    return sum(N-1) + N;
}
```

Base Case

Main Logic.

Function Call Tracing

Function calls are stored in a stack.

```
int add (int x, int y) {
    return x+y;
}
```

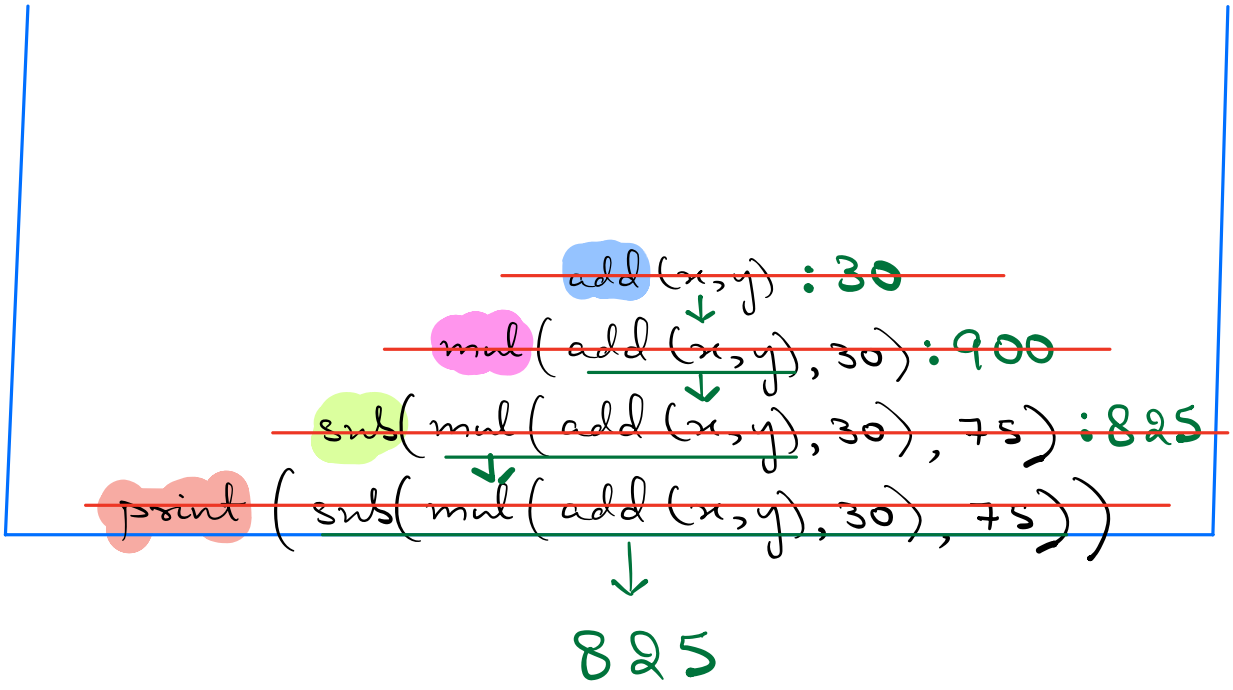
```
int mul (int x, int y) {
    return x*y;
}
```

```
int sub (int x, int y) {
    return (x-y);
}
```

```
void print (int x) {
    // Print the value of x;
}
```

```
int x = 10;
int y = 20;
print (sub(mul(add(x,y),30),75));
```

```
print ( sub( mul( add (x, y), 30), 75) );
```



Given a positive integer N.
Find the value of factorial of N (N!)

factorial of 5

$N! = \underbrace{1 \times 2 \times 3 \times 4 \ldots\ldots (N-1)}_{(N-1)!} \times N$

$N! = (N-1)! \times N$

$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120.$

factorial (N) = factorial (N-1) x N;

Recursive Code

1) Assumption : factorial (N) calculates & returns the correct value of (N!)

2) Main Logic :

factorial (N) = factorial (N-1) × N;

3) Base Case :

if (N == 0) { return 1; }

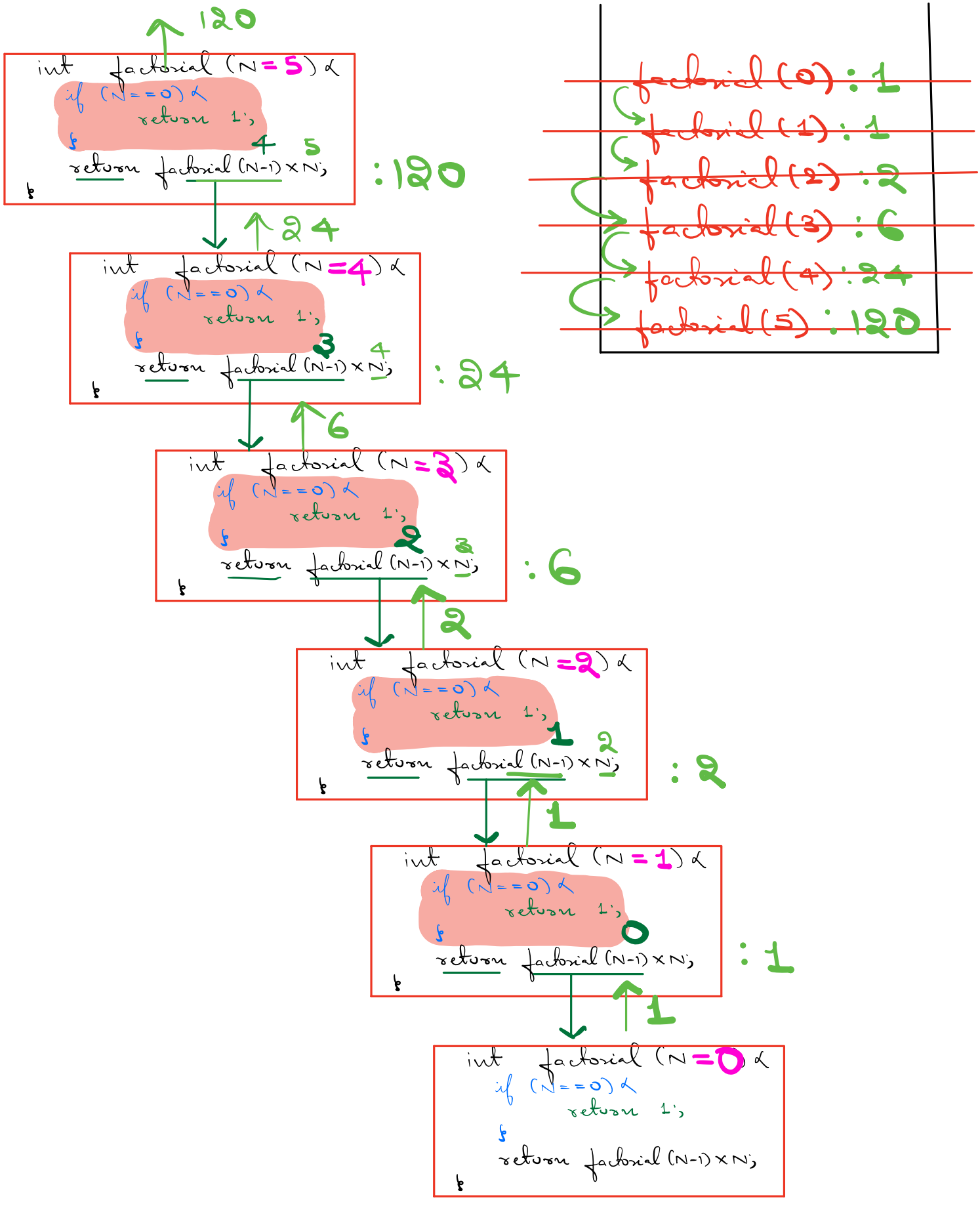
N → (N-1) → (N-2) → (N-3) ..... 2 → 1 → 0

0! = 1

Code

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

Dry Run (N = 5)


120
int factorial (N=5) {
if (N==0) {
return 1;
}
return factorial (N-1) x N;
4
5
:120
24
int factorial (N=4) {
if (N==0) {
return 1;
}
return factorial (N-1) x N;
3
4
:24
6
int factorial (N=3) {
if (N==0) {
return 1;
}
return factorial (N-1) x N;
2
3
:6
2
int factorial (N=2) {
if (N==0) {
return 1;
}
return factorial (N-1) x N;
1
2
:2
1
int factorial (N=1) {
if (N==0) {
return 1;
}
return factorial (N-1) x N;
0
:1
1
int factorial (N=0) {
if (N==0) {
return 1;
}
return factorial (N-1) x N;
factorial (0) : 1
factorial (1) : 1
factorial (2) : 2
factorial (3) : 6
factorial (4) : 24
factorial (5) : 120

Given a +ve integer N. Print the Nth term of the fibonacci sequence.

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

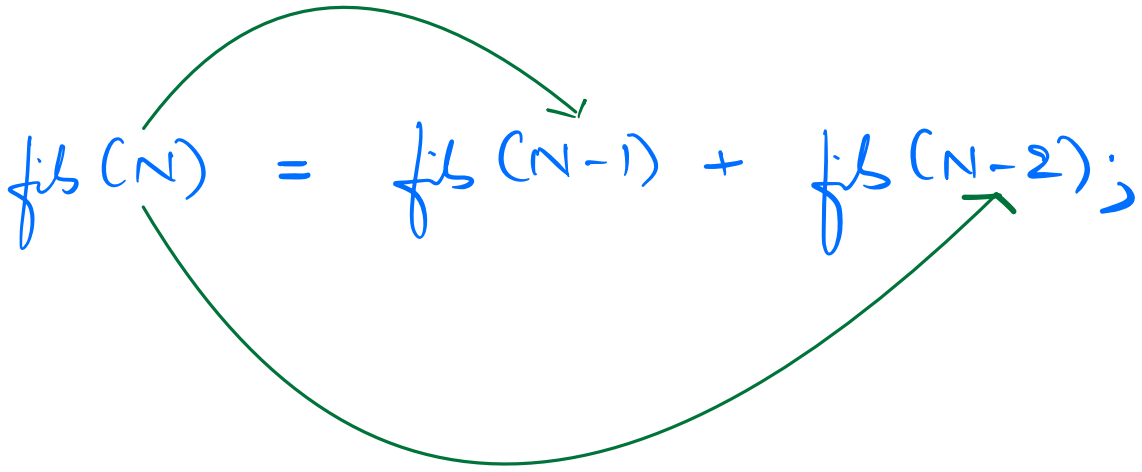
N	0	1	2	3	4	5	6	7	8	9	10
	0	1	1	2	3	5	8	13	21	34	55

1) Assumption: The function fib(N) calculates & returns the correct value of the Nth fibonacci term.

2) Main Logic :

```
fib(N) = fib(N-1) + fib(N-2);
```

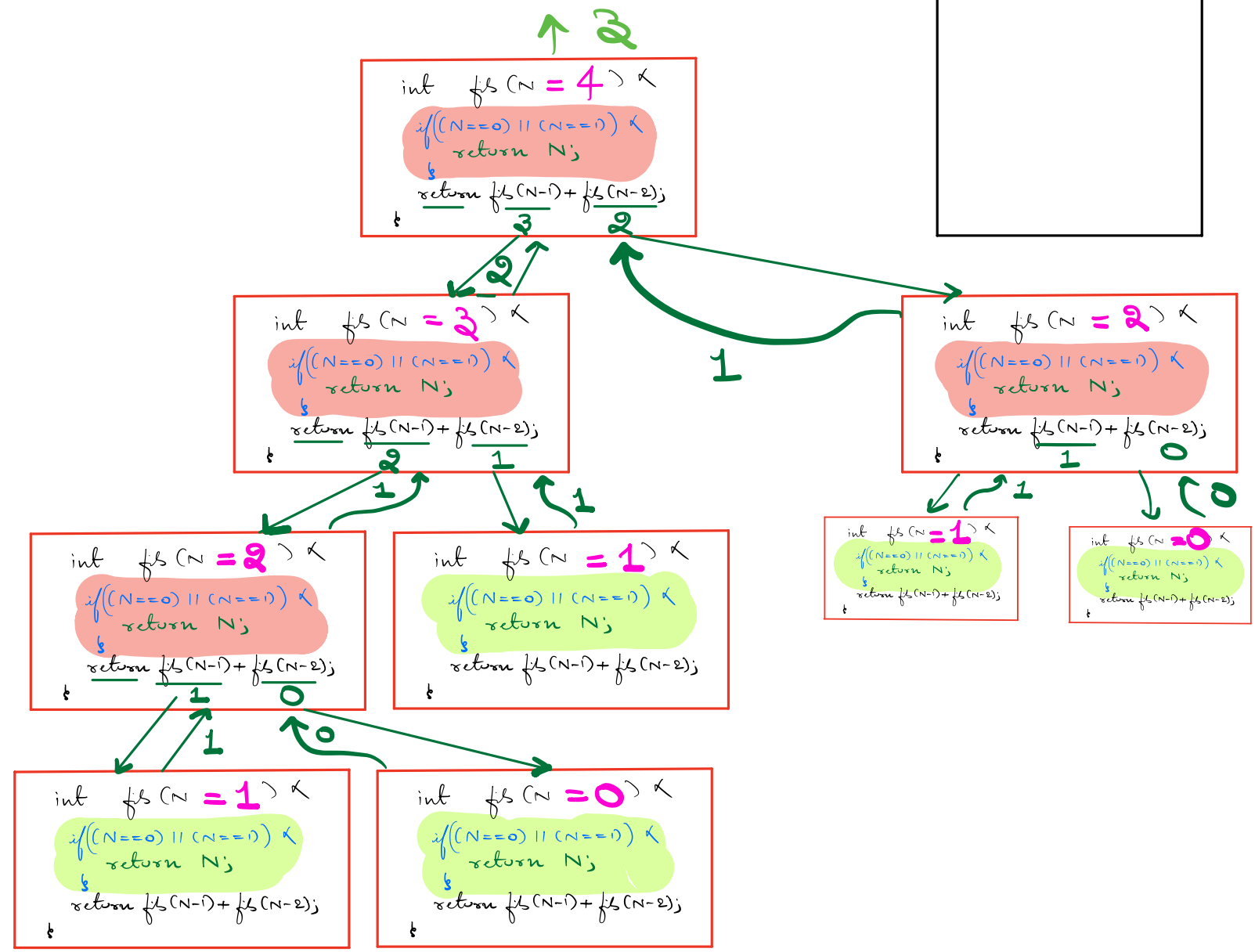
3) Base Case : Smallest value of I/P for which the transition goes out of bounds (becomes invalid)



N	N-1	N-2
0 (Base Case) (0)	-1 (Not valid)	-2 (Not valid)
1 (Base Case) (1)	0 (valid)	-1 (Not valid)
2 (Not a Base Case)	1 (valid)	0 (valid)

Code

```
int fib(N) {
    if((N==0) || (N==1)) {
        return N;
    }
    return fib(N-1) + fib(N-2);
}
```



Given two positive no.s A & N
Calculate A^N.

Eg : A = 2, N = 3 $\Rightarrow 2^3 = 8$ = Ans

A^N = A x A x A ..... A x A x A (N times)

$A^N = (A^{N-1}) \times A$;

Recursive Solution

1) Assumption: int power(A, N): Calculates & returns the correct value of A^N.

2) Main Logic:

power(A, N) = power(A, N-1) × A

3) Base Case:

$A^N \rightarrow A^{N-1} \rightarrow A^{N-2} \ldots\ldots A^2 \rightarrow A^1 \rightarrow A^0$

⇓

if (N==0) { return 1; }

Code

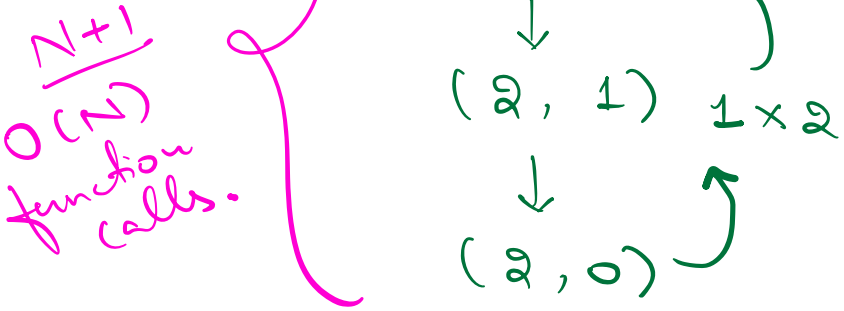
```
int power (A, N) {
    if (N==0) { return 1; }

    return power (A, N-1) × A;
}
```

↑ 8

(2, 3) ↗ 4 × 2

↓

(2, 2) ↗ 2 × 2



$2^{10} = 2^9 \times 2$

$A^N = A^{N-1} \times A$

$2^{10} = 2^5 \times 2^5$

$A^N = A^{N/2} \times A^{N/2}$ (Even)

$2^{11} = 2 \times 2^{10} = 2 \times (2^5) \times (2^5)$

$A^N = A \times A^{N/2} \times A^{N/2}$ (Odd)

Code

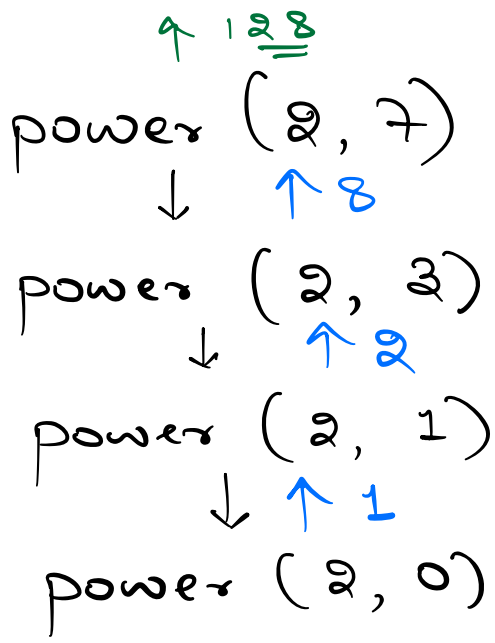
```
int power (A, N) {
    if (N == 0) { return 1; }
    if (N % 2 == 0) {
        return power(A, N/2) * power(A, N/2);
    }
    else {
        return A * power(A, N/2) * power(A, N/2);
    }
```

}

- Optimisation

```
int power (A, N) {
    if (N==0) { return 1; }
    int p = power (A, N/2);
    if ( N % 2 == 0) {
        return pxp;
    }
    else {
        return Axpxp;
    }
}
```

Dry Run ($2^7 = 128$)



T.C. of Recursion using Recurrance Relation Ship

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

$T(0) = O(1)$

Recursive rel^n = factorial (N) = factorial (N-1) * N

⇓ ⇓

Recurrance rel^n ⇒ $T(N) = T(N-1) + 1$

$$T(N) = T(N-1) + 1 \quad \text{— RR}$$

put $N = N-1$ in RR

$$T(N-1) = T(N-2) + 1$$

put $T(N-1)$ in RR

$$T(N) = T(N-2)+1+1$$
$$= T(N-2) + 2$$

$$T(N) = T(N-2) + 2 \quad \text{— (1)}$$

put $N = N-2$ in RR

$$T(N-2) = T(N-3) + 1$$

put $T(N-1)$ in eq (1)

$$T(N) = T(N-3)+1+2$$
$$= T(N-3) + 3$$

$$T(N) = T(N-3) + 3$$

⋮ K times

$$T(N) = T(N-K) + K \rightarrow \text{GE}$$

$$\Downarrow$$

$$N-K = 0$$

$(K = N)$ After N steps

put $N = K$ in GE

$$T(N) = T(0) + N$$

1

$$T(N) = (N+1)$$

$$T.C. = O(N)$$

Alternate Way

$$\text{T.C. of Recursive Soln} = \left(\begin{matrix}\text{Total No. of}\\ \text{Recursive}\\ \text{Calls}\end{matrix}\right) \times \left(\begin{matrix}\text{Time per}\\ \text{Recursive}\\ \text{Call}\end{matrix}\right)$$

$$O(N) \times O(1)$$

$$T.C. = O(N)$$

T.C. of Power functions.

(1)

```
int power (A, N) {
    if (N==0) { return 1; }

    return power (A, N-1) xA;
}
```

$$T(N) = T(N-1) + 1$$

$$T.C. = O(N)$$

(2)

```
int power (A, N) {
    if (N == 0) { return 1; }
    if (N % 2 == 0) {
        return power(A, N/2) * power(A, N/2);
    }
    else {
        return A * power(A, N/2) * power(A, N/2);
    }
}
```

$T(1) = 1$

$$T(N) = 2 \times T(N/2) + 1 \quad — (1)$$

$$T(N/2) = 2 \times T(N/4) + 1$$

$$T(N) = 2\left(2 \times T\left(\frac{N}{4}\right) + 1\right) + 1$$

$$T(N) = 2^2 T\left(\frac{N}{2^2}\right) + 2^1 + 2^0 \quad — (2)$$

$$T\left(\frac{N}{2^2}\right) = 2 \times T\left(\frac{N}{2^3}\right) + 1$$

$$T(N) = 2^2\left(2 \times T\left(\frac{N}{2^3}\right) + 1\right) + 2^1 + 2^0$$

$$T(N) = 2^3 \times T\left(\frac{N}{2^3}\right) + 2^2 + 2^1 + 2^0$$

⋮ K times

$$T(N) = 2^k \times T\left(\frac{N}{2^k}\right) + 2^{k-1} + 2^{k-2} \ldots\ldots 2^1 + 2^0$$

$$T(N) = \underline{2^k} \times T\left(\frac{N}{2^k}\right) + \left(2^k - 1\right)$$

$$\frac{N}{2^k} = 1$$

$$\Rightarrow N = 2^k$$

$$K = \log_2 N$$

$$T(N) = N \times (1) + N - 1$$

$$= 2N - 1$$

$$T.C. = O(N)$$

(3)

```
int power (A, N) {
    if (N==0) { return 1; }
    int p = power (A, N/2);
    if (N%2 == 0) {
        return pxp;
    }
    else {
        return Axpxp;
    }
}
```

$$T.C. = O(\log_2 N) \times O(1)$$

$$= O(\log_2 N)$$

$$T(N) = T\left(\frac{N}{2}\right) + O(1)$$

H.W. to Solve

Space Complexity

The maximum space occupied by the stack during the entire execution of the recursive solution.

H.W. figure out the S.C. of each question we have solved.

Diwali 10th ⟶ 17th

⇓

25 people

3 question in Advanced DSA

Only for Advanced people

T.C. AT

A1
A2
A3
BM1
BM2
R1

$6 \times 4 = 24$ ques^n

30 mins

19/24

(47 %)