Recursion - A function calling itself.

Thought Process

$Sum(N) = 1 + 2 + 3 + 4 + 5 + \ldots\ldots + N-1 + N$

15 = 10 + 5

$Sum(5) = Sum(4) + 5$

↓ 10 = 6 + 4

$Sum(4) = Sum(3) + 4$

↓ 6 = 3 + 3

$Sum(3) = Sum(2) + 3$

↓ 3 = 1 + 2

$Sum(2) = Sum(1) + 2$

1

$Sum(1) = Sum(0) + 1$ (0)

we are breaking a big problem into smaller achievable chunks to achieve our goal.

NOTE: 0 is not a natural number.

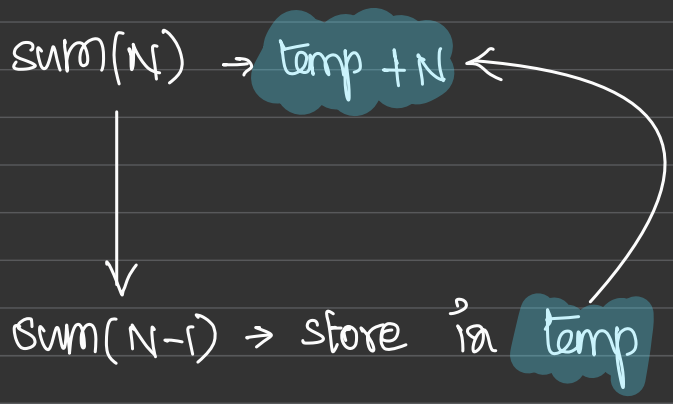
Q. Given N, find sum of numbers from (1...N) using recursion

Three magical steps to remember -

a. **Faith** - define what your function must do and have faith that it works.

The faith of the above problem is give any N and return sum of first N natural numbers.

b. Main logic - solve your problem with subproblem.
subproblem - smaller instance of the same problem.



c. Base case - solution to the smallest subproblem.
Here, smallest problem is N = 1. When N is 1 we return 1.

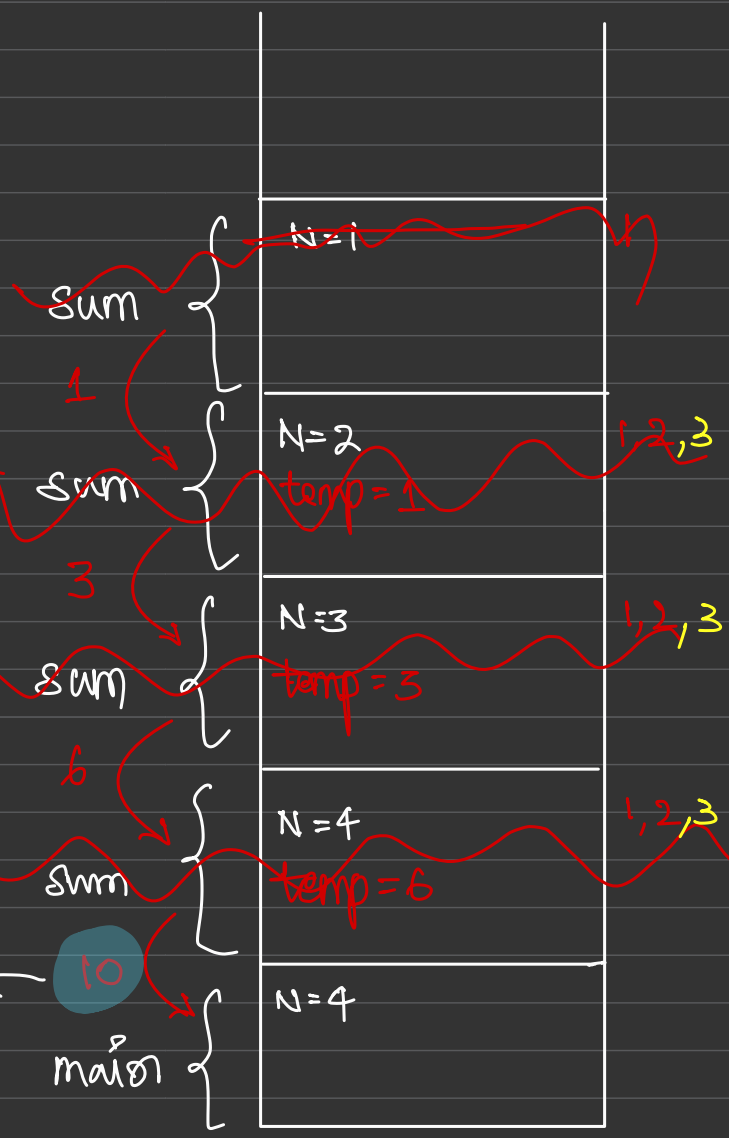
code:

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

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

d. Tracing - To know if the above code works we need to perform tracing every time you write a recursive function. AKA dry run.

```
int sum (int N) {
1. if (N==1) return 1;
2. int Temp = sum(N-1);
3. return temp+N;
}
```



for N=4, 10 will be printed in main().

Q. Find factorial of N

Ex:- N=3 → 3*2*1 = 6

N=4 → 4*3*2*1 = 24

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

a. **faith:** given N, find factorial and return.

b. **main logic:** solve with subproblem.

fact(N) ← temp*N

↓

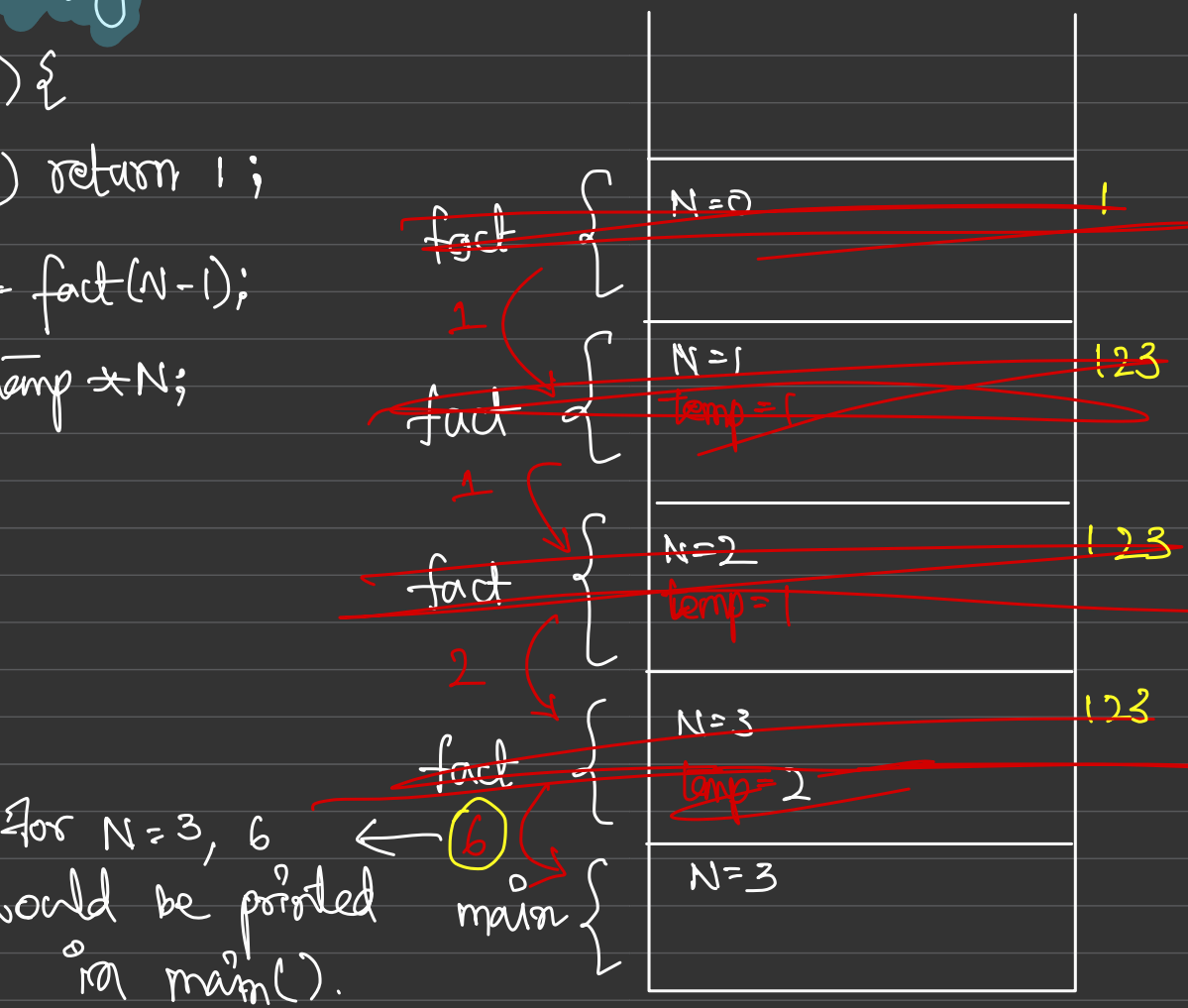
fact(N-1) = Temp

c. **base case:**

fact(0)=1

d. dry run / tracing

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



for N=3, 6 would be printed in main().

Q. Print n^{th} fibonacci number using recursion

Ex:- 0 1 1 2 3 5 8 13 21 34 55

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

a. Faith: given n, calculate & return n^{th} fibonacci number

b. main logic:

fib(N) => temp1 + temp2

fib(N-1) ↓ Temp1

fib(N-2) ↓ Temp2

c. Base case:

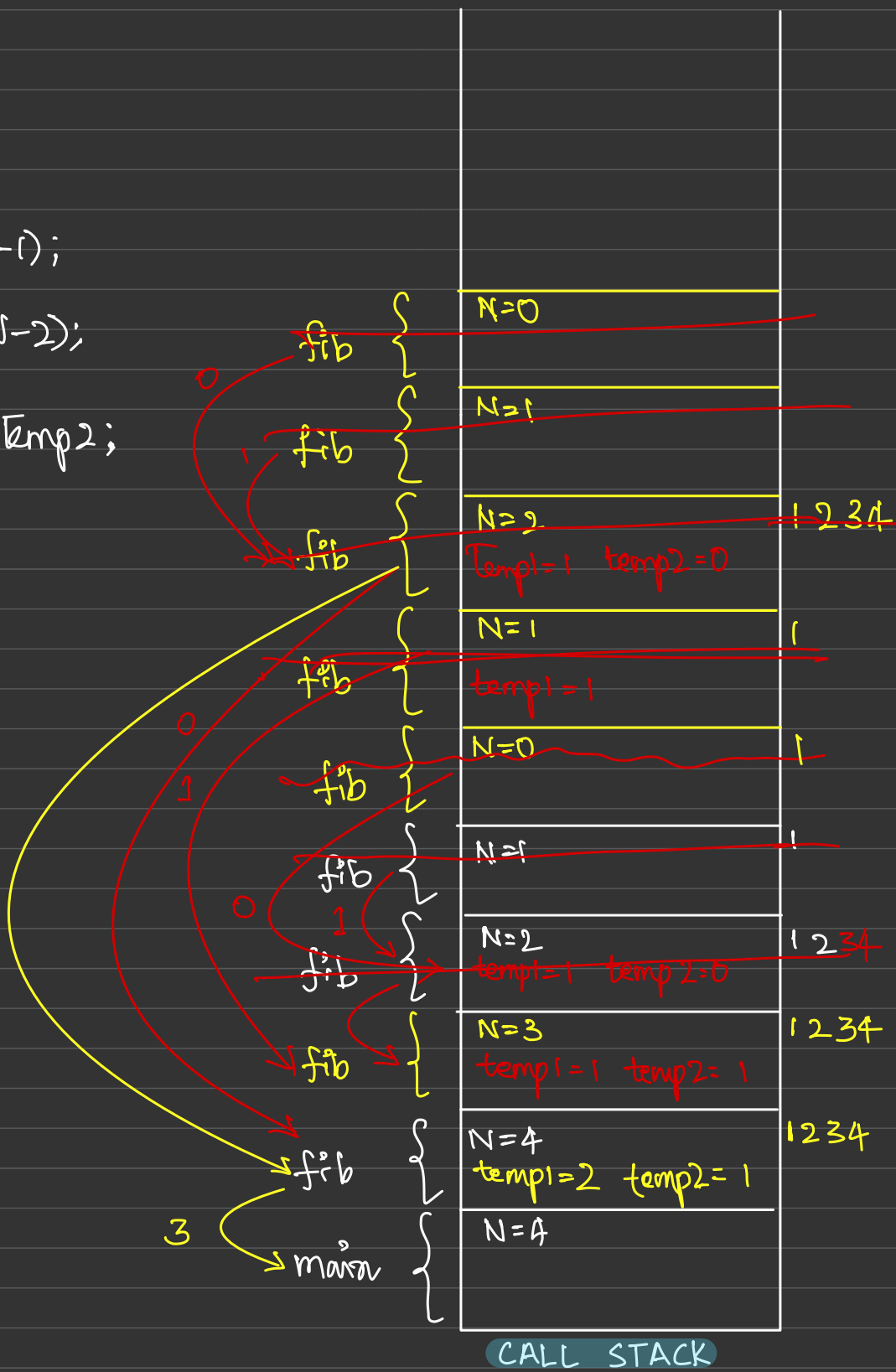
fib(0) = 0

fib(1) = 1

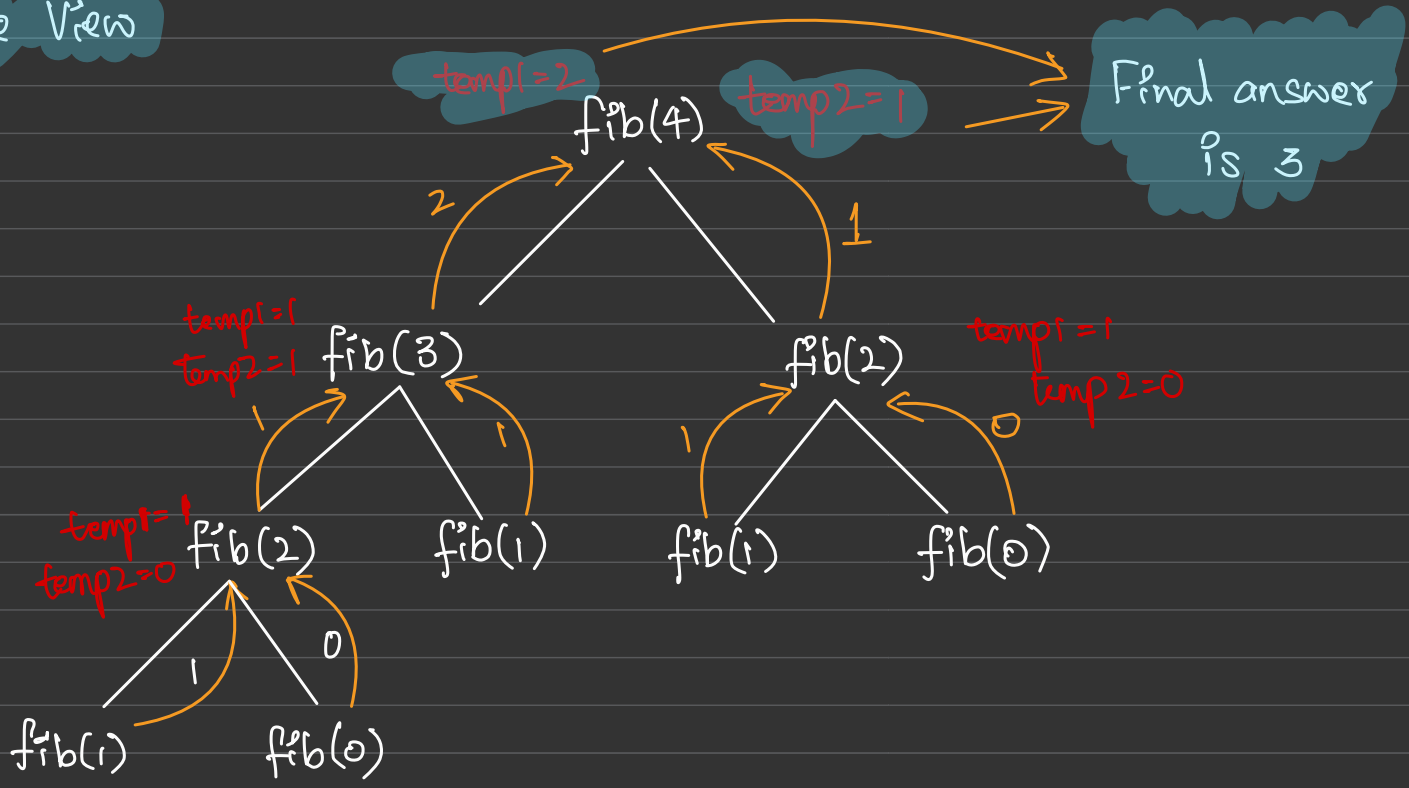
code

```
int fib(int N){
Line
1  {if(N==0 || N==1)
      return N;
2  {int temp1 = fib(N-1);
3  {int temp2 = fib(N-2);
4  {return temp1 + temp2;
   }
```

Tracing / Dry Run



Tree View



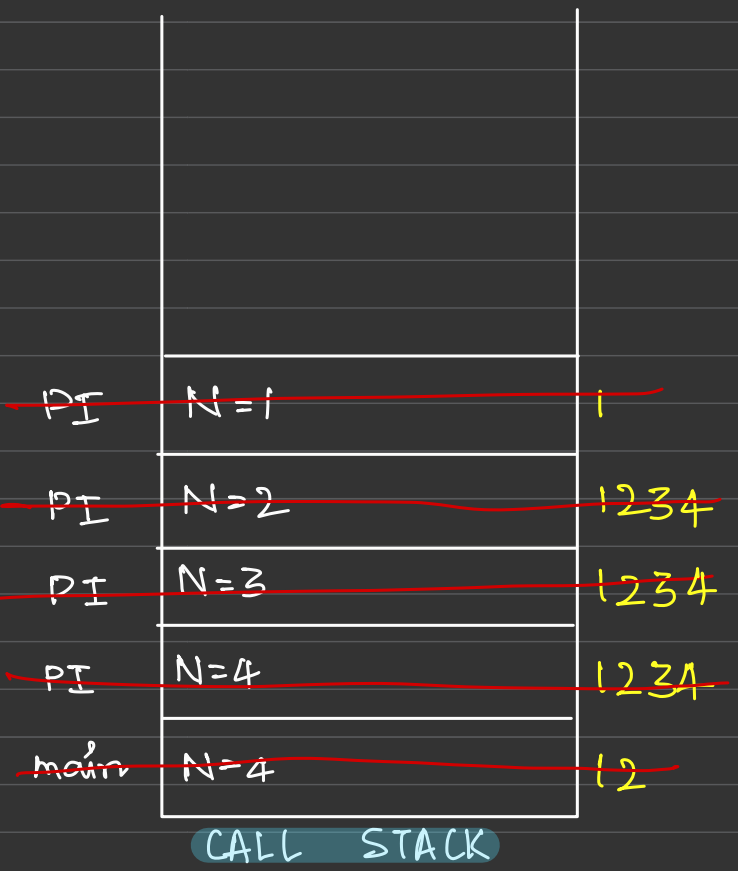
Q. Given N, print all the numbers from 1 to N using recursion.

a. faith - given N, print numbers from 1 to N.

b. main logic - if our function can print N numbers then it can definitely print N-1 numbers.

c. base case -
```
if(N==1) {
    system.out.println(1);
    return;
}
```

```
void printIncreasing(int N){
    if (N==1){                          line 1
        System.out.println(1);
        return;
    }
    printIncreasing(N-1);               2
    System.out.println(N);              3
    return;                             4
}
```



output: 1 2 3 4

Tree View: