Today's agenda

- Recursion
- How to write Recursive code.

→ Google

- Graphs / Trees / DP
  - recursion is must.

Recursion:

↳ function calling itself.

* function call

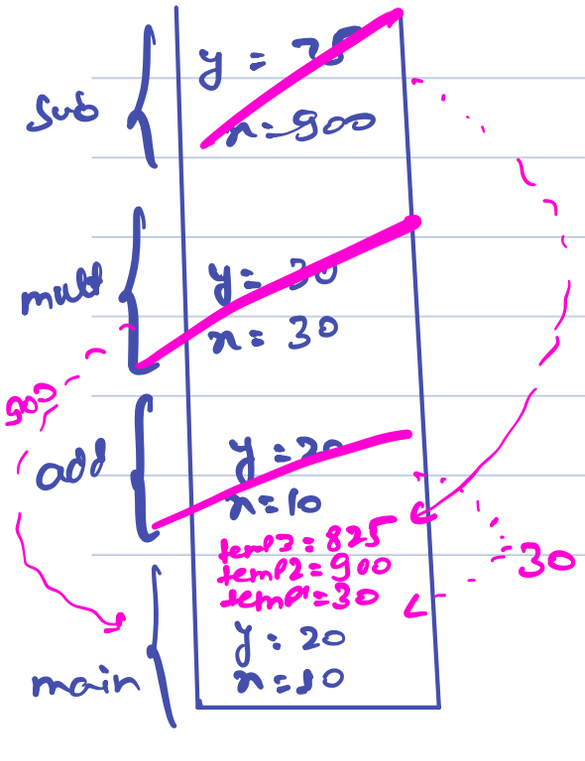
```
main ( ) {
    int x = 10;
    int y = 20;
    int temp1 = add(x, y);
    int temp2 = mult(temp1, 30);
    int temp3 = sub(temp2, 75);

    → s.o.p(temp3);
        ↳ 825
}
```

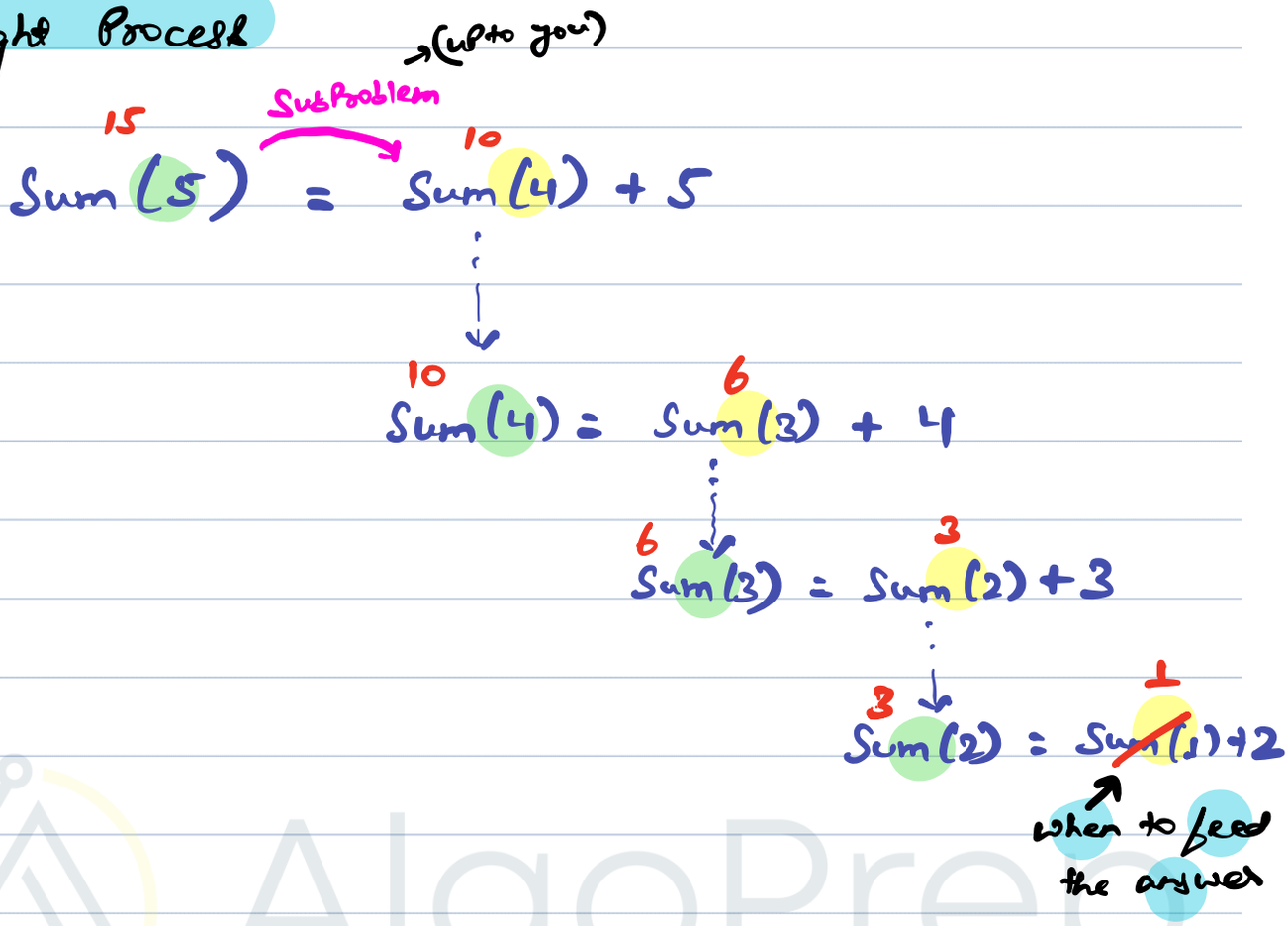
```
int add (int x, int y){
    return x+y;
}
```

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

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



// Thought Process



Q) Given N, find Sum of no.s from (1.....N), using recursion

Three magical steps of recursion.

Faith: define what your function is going to solve.
Have faith that function does it.

main logic: Figure out subProblem of your problem.

base case: Solution to smallest subProblem. (when to feed the answer)


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

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

Faith: Given N, Calculate and return Sum of first N natural no.

main logic: Sum (N) → temp+N

Sum (N-1) → temp

base case: Sum (1) = 1

T.C of 1 function: O(1)

No. of function: N

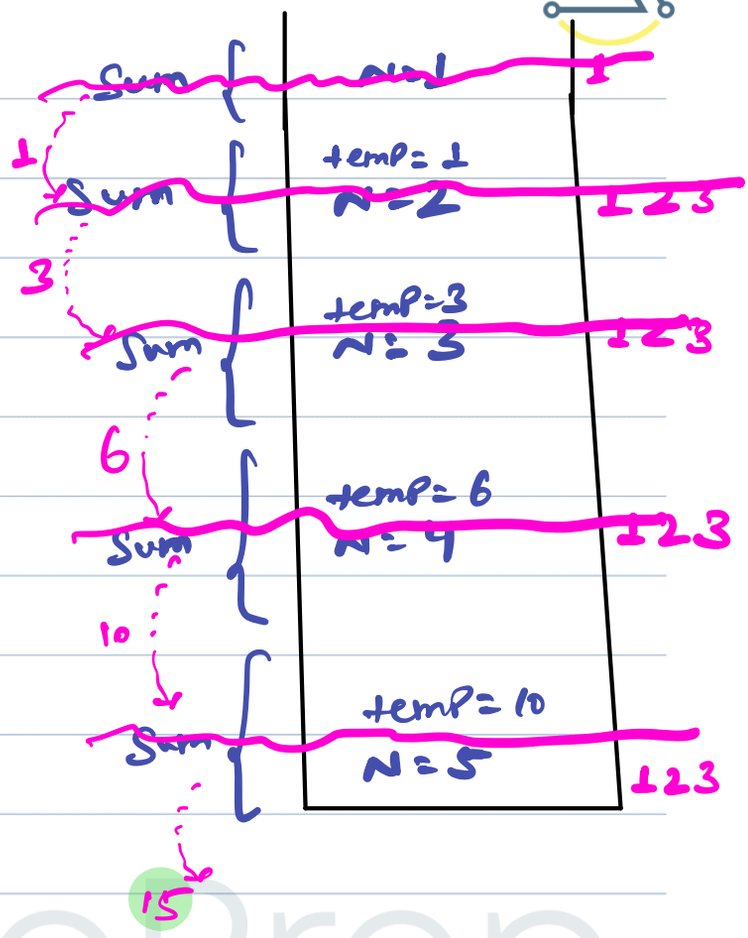
overall: O(1) * N = O(N)

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

  2 int temp = Sum (N-1);

  3 return temp + N;

}
```



Sum(5) → 10+5 = 15

↑ 10

Sum(4) 6+4=10

↑

Sum(3) → 3+3=6

↑

Sum(2) → 1+2=3

↑ 1

Sum(1)

Q) find factorial of N.

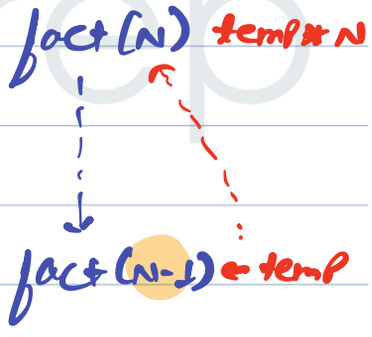
Ex: N=3 → 3*2*1 = 6

N=4 → 4*3*2*1 = 24

```
int fact (int n) {
    if (N==2) { return 2; }
    int temp = fact (N-1);
    return temp*N;
}
```

Faith: Given N, Calculate and return factorial of first N natural no.

main logic:



T.C of 1 function: O(1)

No. of function: N

overall: O(1)*N = O(N)

Base case: fact(2) = 2

```
int fact (int n) {
  1  if (n==2) { return 2; }
  2  int temp = fact (n-1);
  3  return temp * n;
}
```

Break till 10:35 Pm

Q) Print N^{th} fibonacci number, with recursion.

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

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

↳ fib(0) = 0

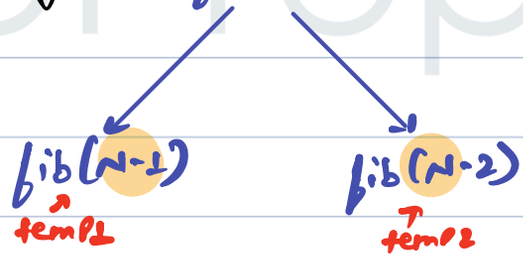
↳ fib(1) = 1

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

    int temp1 = fib(N-1)
    int temp2 = fib(N-2)
    return temp1 + temp2;
}
```

Faith: Given N, calculate and return Nth fibonacci no.

Main logic: fib(N) → temp1 + temp2



Base case:

↳ fib(0) = 0

↳ fib(1) = 1

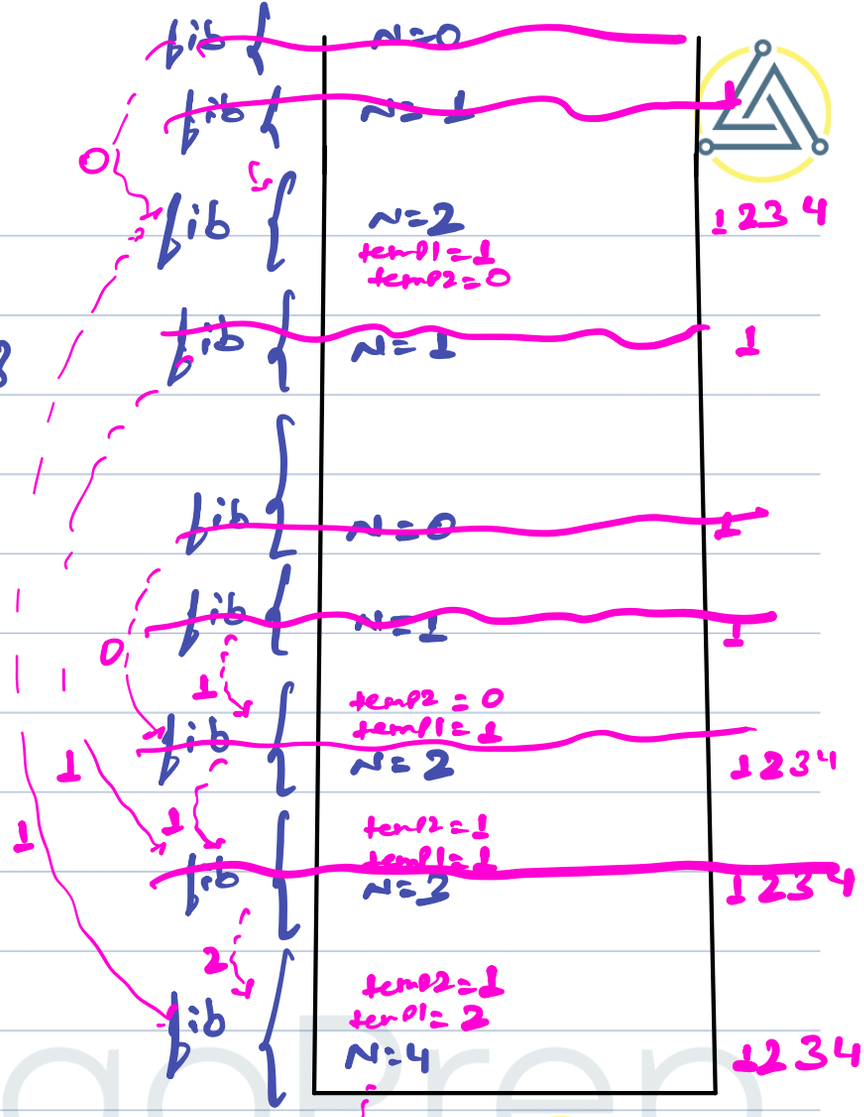
T.C of 1 func: O(1)

No. of func:

N=4

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

2 int temp1 = fib(N-1)
3 int temp2 = fib(N-2)
4 return temp1 + temp2;
}
```




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

2 int temp1 = fib(N-1)
3 int temp2 = fib(N-2)
4 return temp1 + temp2;
}
```


Q) Print increasing

↳ Given N, Print all the numbers from 1 → N, using recursion.

```
void PrintInc (int N){
    if(N==1){ S.O.P(1);
              return;}

    PrintInc(N-1);
    S.O.P(N);
    return;
}
```

Faith: Given N, Print nos 1 → N

main logic:

{1 2 3 . . . N-1} N

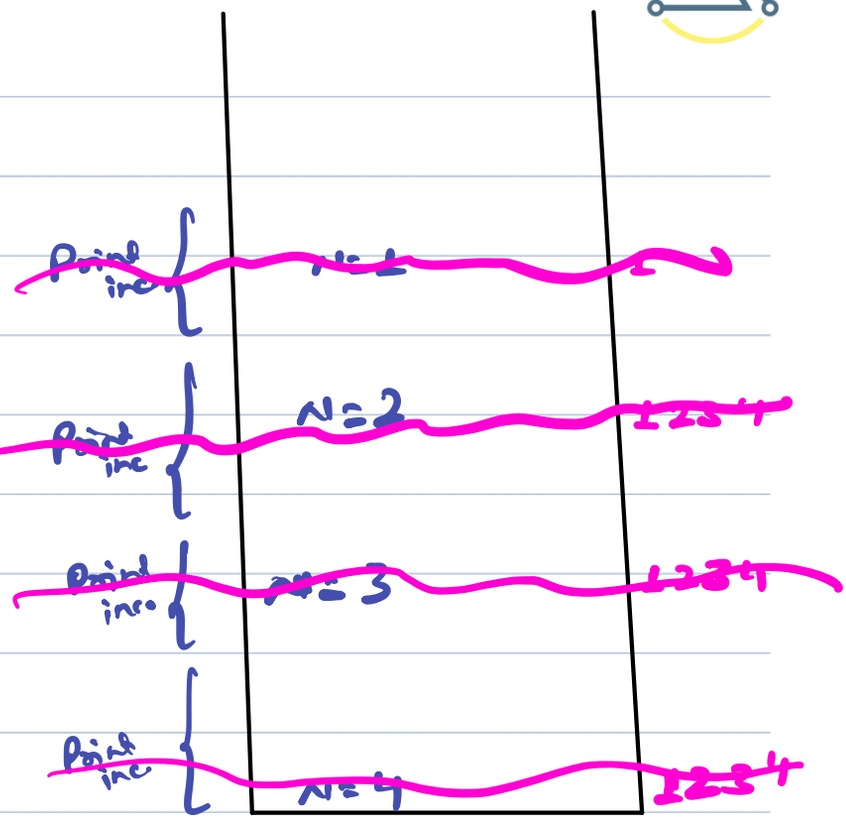
Base case:

N==1.

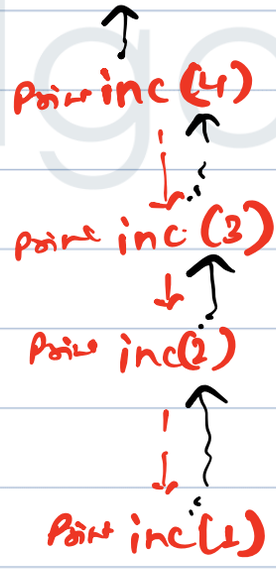
↳ S.O.P(1);

```
void PrintInc (int N){
1 if(N==1){ S.O.P(1);
            return;}

2 PrintInc(N-1);
3 S.O.P(N);
4 return;
}
```



1
2
3
4



1
2
3
4