Today's agenda
- Recursion
- How to write Recursive code.

Why recursion →
- Tree
- Backtracking
- DP → Amazon
- Graph

Google ←

Recursion:

↳ function calling itself.

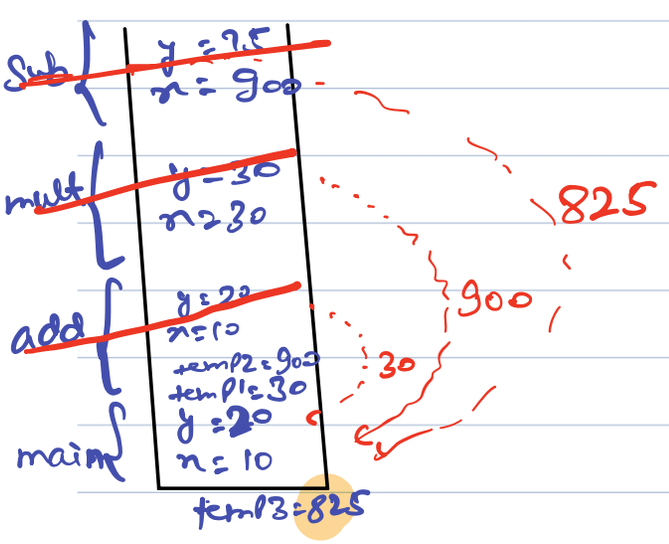
* function call

```
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);
}
```

```
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

N = 5

Sum(N) = 1 + 2 + 3 + ... + N-1 + N

Sum(5) = Sum(4) + 5

Sum(4) = Sum(3) + 4

Sum(3) = Sum(2) + 3

Sum(2) = Sum(1) + 2

Sum(1) = Sum(0) + 1

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

Three magical steps of recursion.

Faith: define what your function should do and have faith that it works.

Main logic: Solve your Problem with SubProblem
↳ smaller instance of same Problem.

Base Case: Solution to Smallest SubProblem.


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

    int temp= Sum(N-1);

    return temp+N;
}
```

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

Main logic:

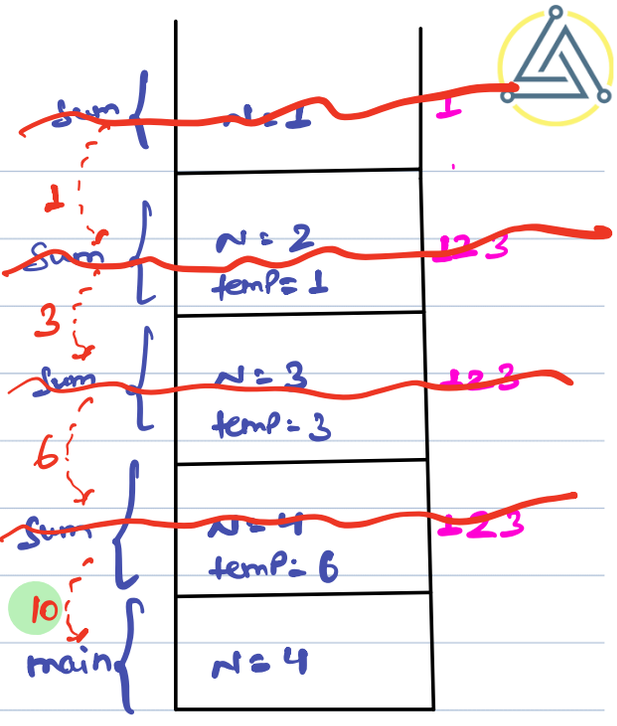
Sum(N) temp+N

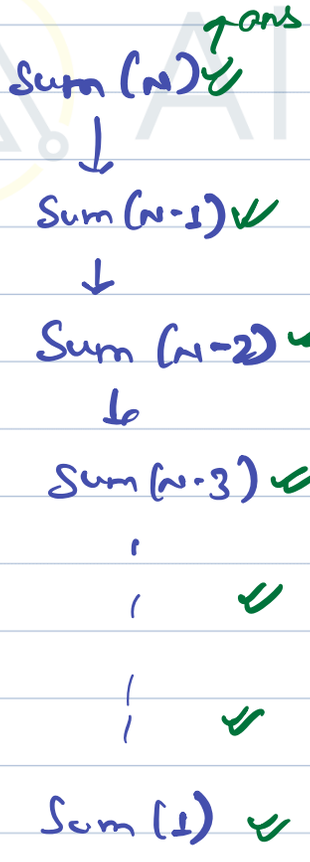
Sum(N-1) → temp

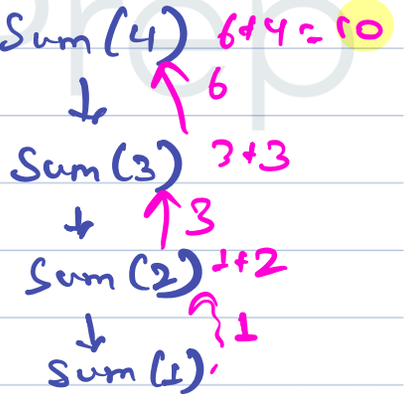
base case: N==1 → 1

Tracing → dry run

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

2  int temp= Sum (N-1);

3  return temp + N;

}
```







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;
}
```

Faith: Given N, Calculate & return factorial of N.

Main logic:

fact(N) temp*N

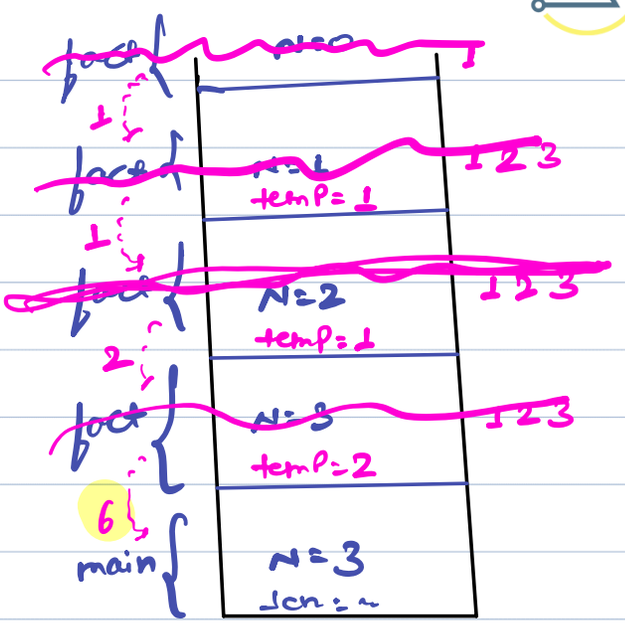
fact(N-1) -> temp

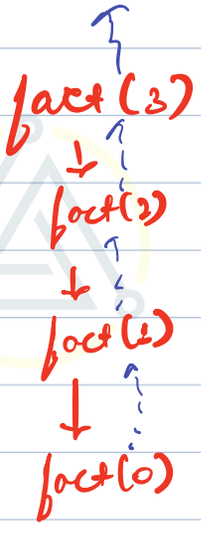
base case:

fact(0) = 1

dry run

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





Break till 9:40 PM

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

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

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

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

    int temp1 = fib(N-1);
    int temp2 = fib(N-2);

    return temp1 + temp2;
}
```

Faith: Given N, calculate & return Nth fibonacci number.

Main logic:

fib(N) → temp1 + temp2

fib(N-1) → temp1

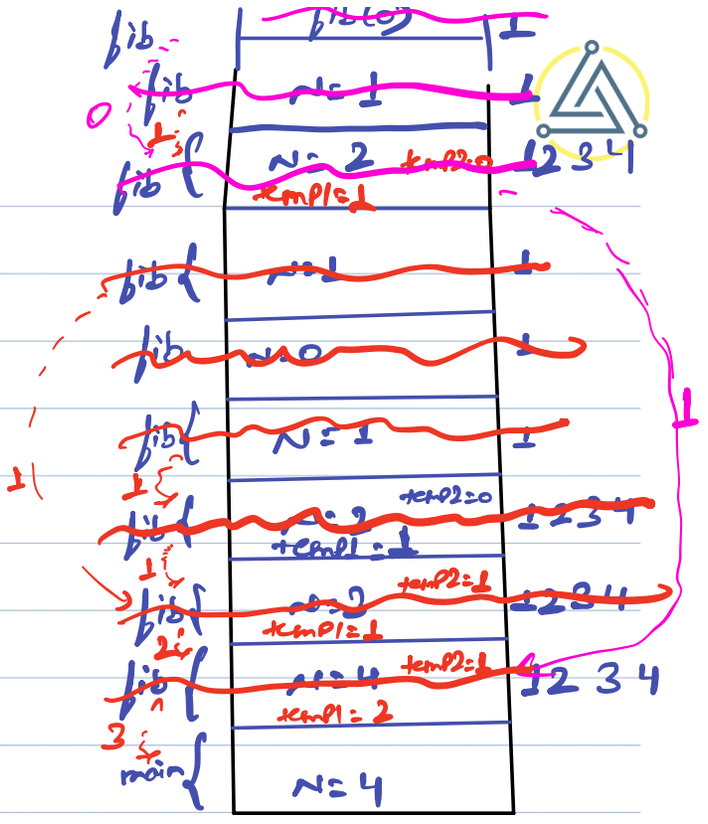
fib(N-2) → temp2

Base Case:

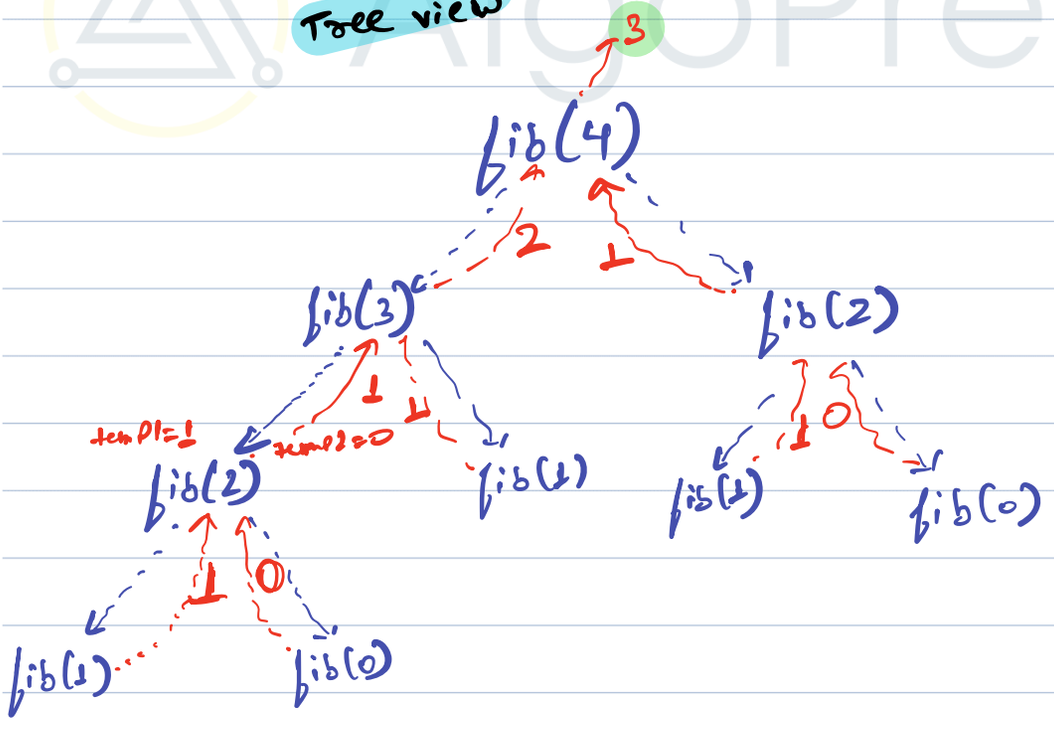
↳ fib(0) = 0

fib(1) = 1

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

2  int temp1 = fib(N-1);
3  int temp2 = fib(N-2);

4  return temp1 + temp2;
}
```



Tree view



Q) Print increasing

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

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

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

Faith: Given N, Print nos from 1 to N.

Main logic:

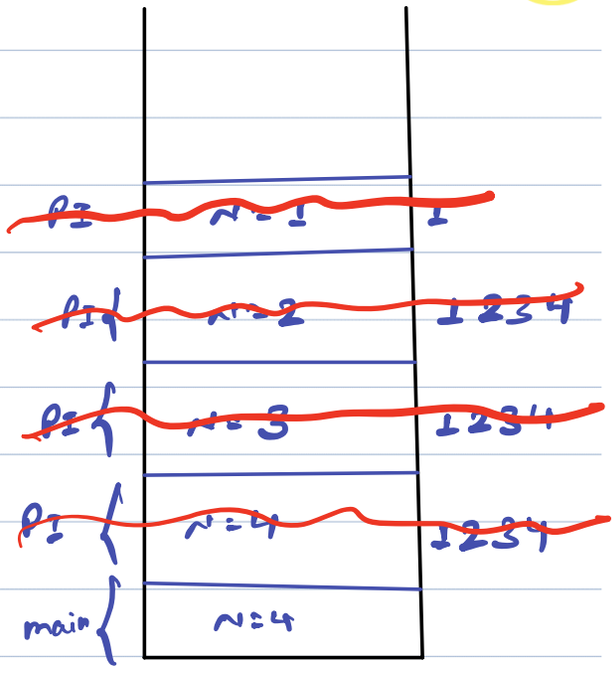
{1 2 3 . . N-1}N

Base Case:

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

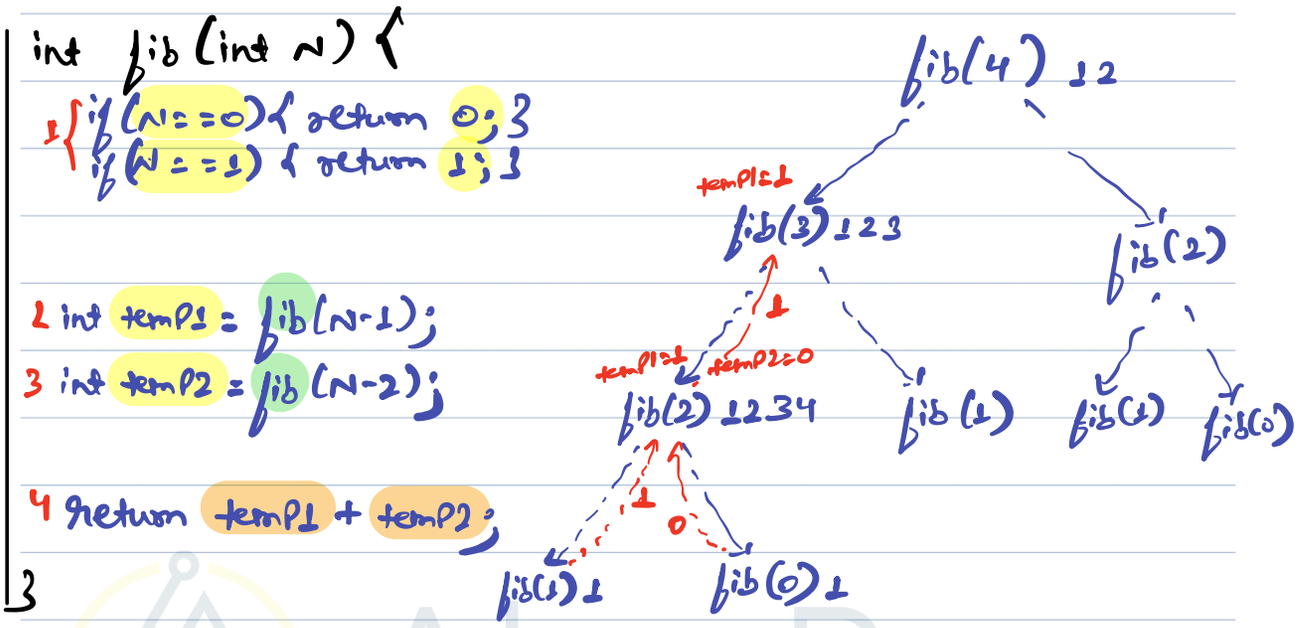
Tracing

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

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



1
2
3
4

PI(4) 4
↓
PI(3) 3
↓
PI(2) 2
↓
PI(1)


int fib(int N){
1 if(N==0){return 0;}
if(N==1){return 1;}
2 int temp1 = fib(N-1);
3 int temp2 = fib(N-2);
4 return temp1 + temp2;
}
fib(4) 12
temp1=1
fib(3) 123
fib(2)
temp1=1
temp2=0
fib(2) 1234
fib(1)
fib(1)
fib(0)
fib(1) 1
fib(0) 1