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

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

Recursion:

↳ function calling itself.

* function call

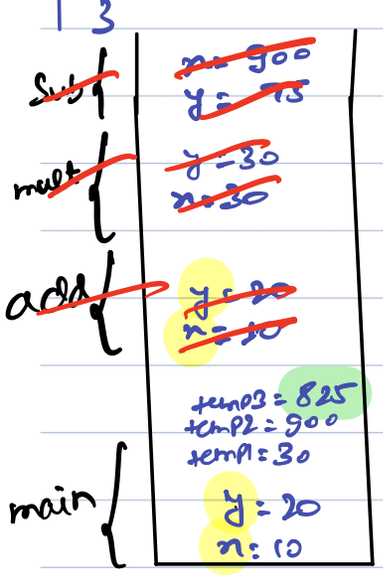
```
main () {
    int x = 10;
    int y = 20;
    int temp1 = add(x,y);
    int temp2 = mult(temp1,30);
→   int temp3 = Sub(temp2,75);

    S.o.p(temp3);
}
```

↳ 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

Sum(N) = 1 + 2 + 3 + 4 + ... + (N-1) + N

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

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

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

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

Sum(1) = Sum(0) + 1

Sum(5) = Sum(4) + 5

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

Three magical steps of recursion.

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

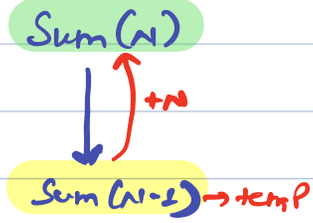
Main logic: Solving your problem with subproblem.
↓
Smaller instance of same problem

Base Condition: 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 N numbers.

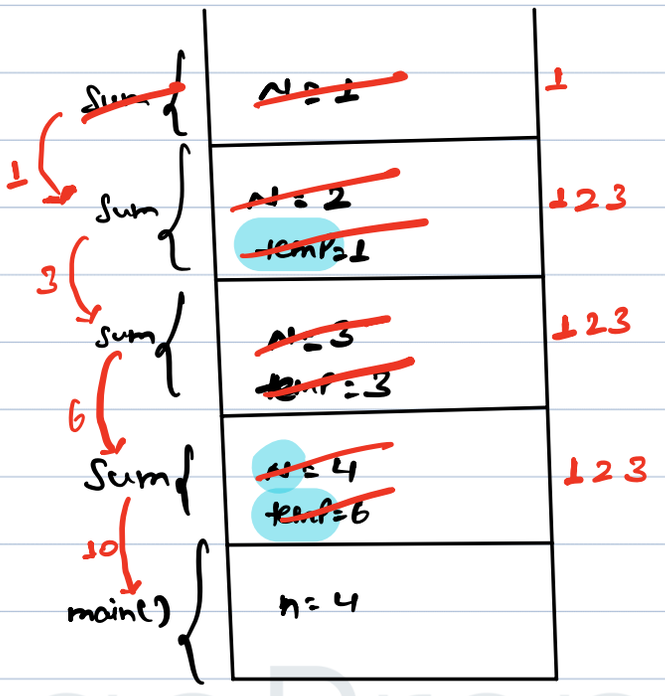
main logic:



base case:

sum (1) = 1

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



Sum(N) ✓✓
↓
Sum(N-1) ✓✓
↓
Sum(N-2) ✓✓
↓
Sum(N-3) ✓✓
⋮ ✓✓
⋮ ✓✓
⋮
Sum(1) ✓✓

Q) find factorial of N.

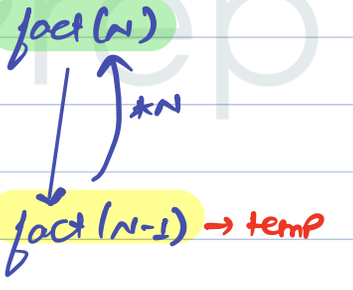
Ex: N=3 → 3*2*1 = 6

N=4 → 4*3*2*1 = 24

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

    int temp = fact(N-1);

    return N*temp;
}
```

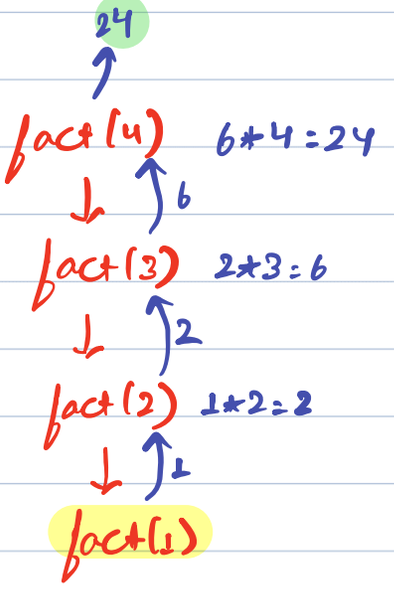
Faith: Given N, calculate & return factorial of N.

Main logic:



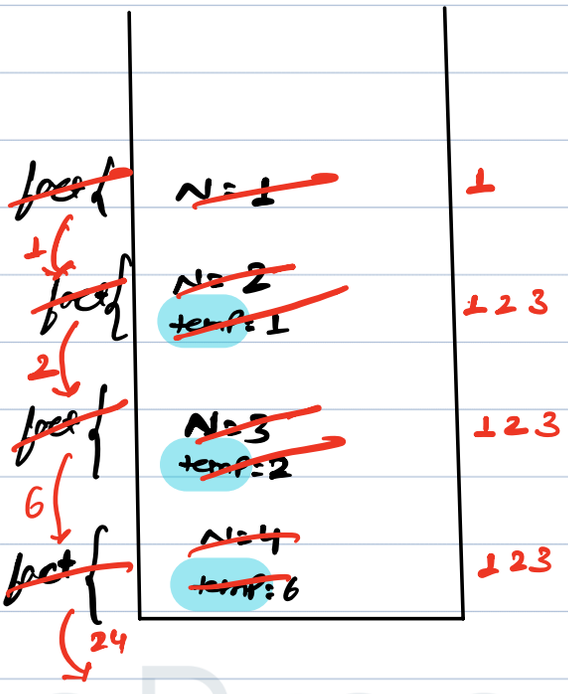
Base case:

fact(1) = 1



Break till 9:37 PM

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

2  int temp = fact(N-1);

3  return N*temp;

}
```



Q) Print N^{th} fibonacci number

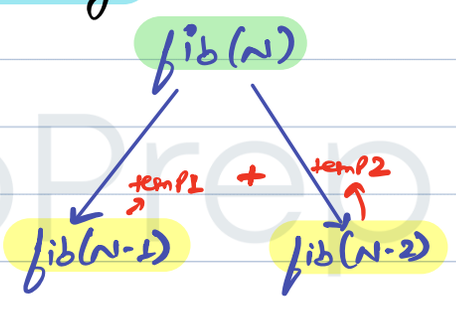
	0	1	2	3	4	5	6	7	8	9	10
Ex:	0	1	1	2	3	5	8	13	21	34	55

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

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

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

    return temp1 + temp2;
}
```

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

main logic:

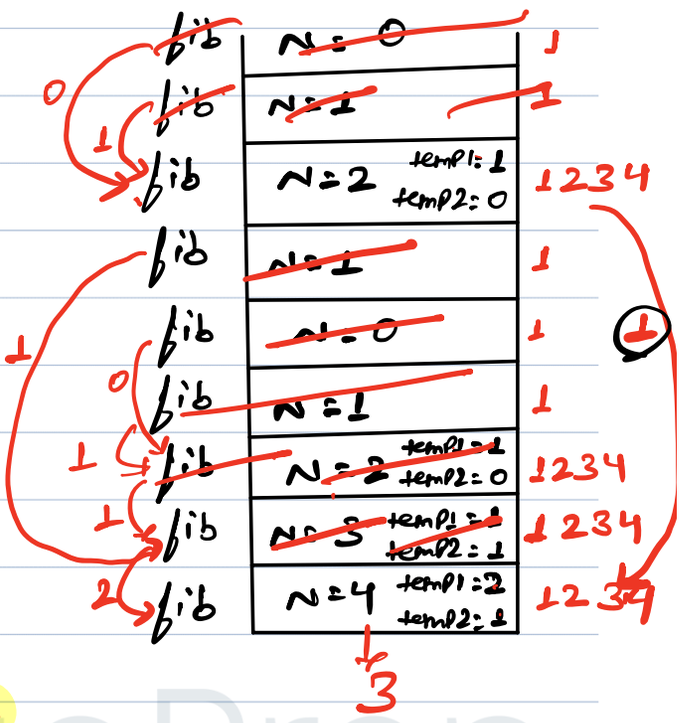


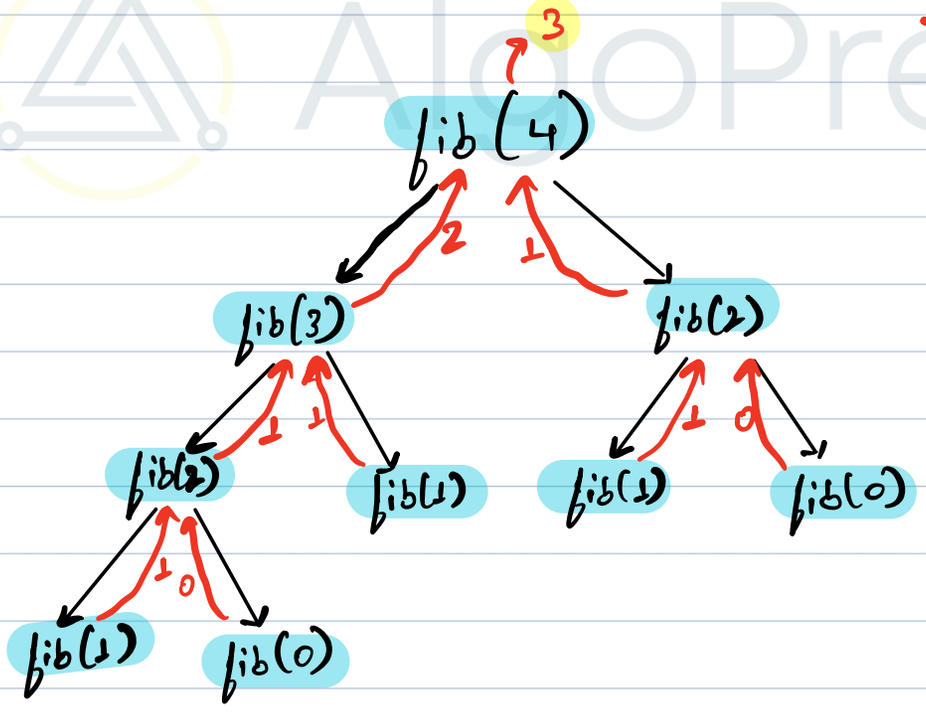
Base case:

↳ fib(0) = 0

↳ fib(1) = 1

```
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.

N=5 → 1 2 3 4 5

```
void inc (int N){
    if (N==1) {Print(1);
               return;}
    inc (N-1);
    s.o.p(N);
    return;
}
```

Faith: Given N, Print no. from 1 to N.

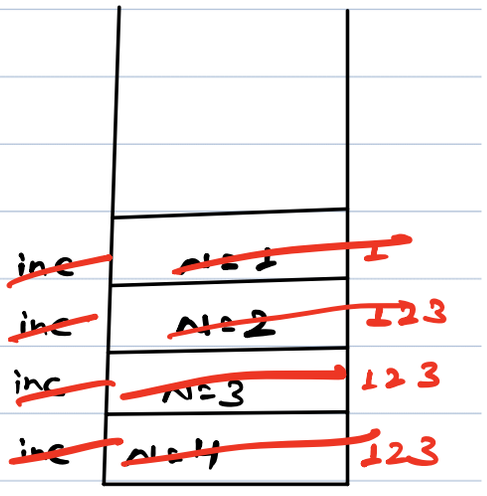
main logic:

{1, 2, 3, 4. . . . . N-1}N}

Base case:

↳ Print (1) for N==1.

```
void inc (int N){
1 if (N==1) { Print(1);
              return; }
2   inc(N-1);
3   S.O.P(N);
    return;

3
```



1 2 3 4

↳ you didn't come this far only to come this far.