Recursion - 1

Definition: Function calling itself.

Why? → A problem is broken down into smaller problem (subproblem) and the solution is generated using the subproblems.

Eg. Sum of first N natural numbers.

$$Sum(5) = 1 + 2 + 3 + 4 + 5$$

$$= Sum(4) + 5$$

Writing recursive code :

1) **Assumptions :** Decide what the function will do and it's output.

2) **Main Logic :** Break the problem into subproblem to solve the assumption.

3) **Base Case :** Identify the inputs for which we will stop recursion.

Function call Tracing

It involves the sequence of function calls that are made when a program is executed.

825

```
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)
{   printf(x);
}

int main()
{   int x=10;
    int y=20;
    print(sub(mul(add(x,y),30),75));
    return 0;
}
```

main() {
x=10, y=20
print(sub(mul(add(x,y),30),75));
}

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

825

sub(mul(add(x,y),30),75)

900

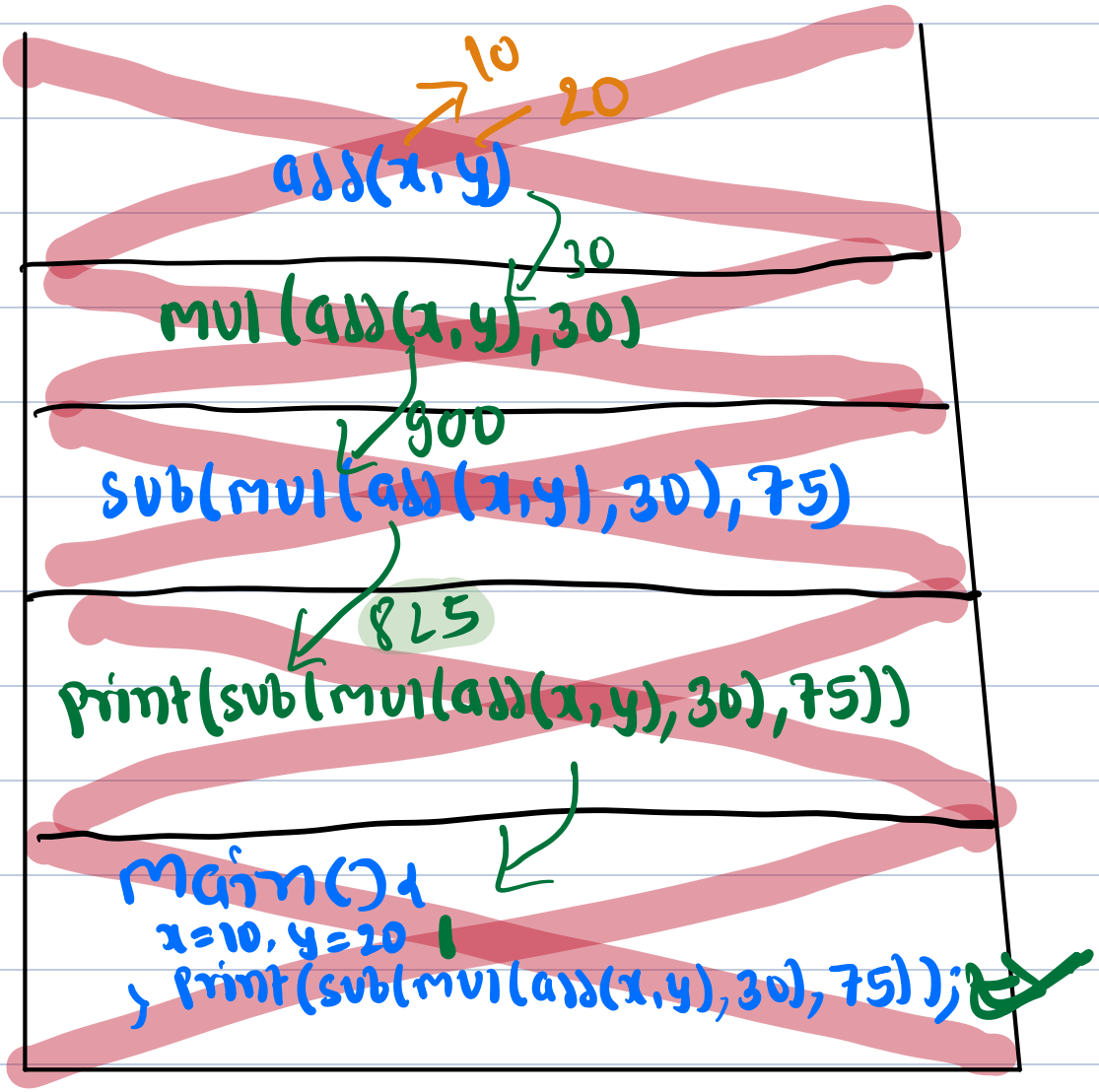
mul(add(x,y),30)

30

add(x,y)

10 20

DS involved in function Calls



Q1) Given a positive integer N, find the factorial of N.

Quiz 1:

N = 5

5! = 5 * 4 * 3 * 2 * 1 = 120

fact(5) = fact(4) * 5

Solution:

Assumption: Creat a function which takes N as an input and return fact(N) as an output.

Main logic: fact(N) = fact(N-1) * N

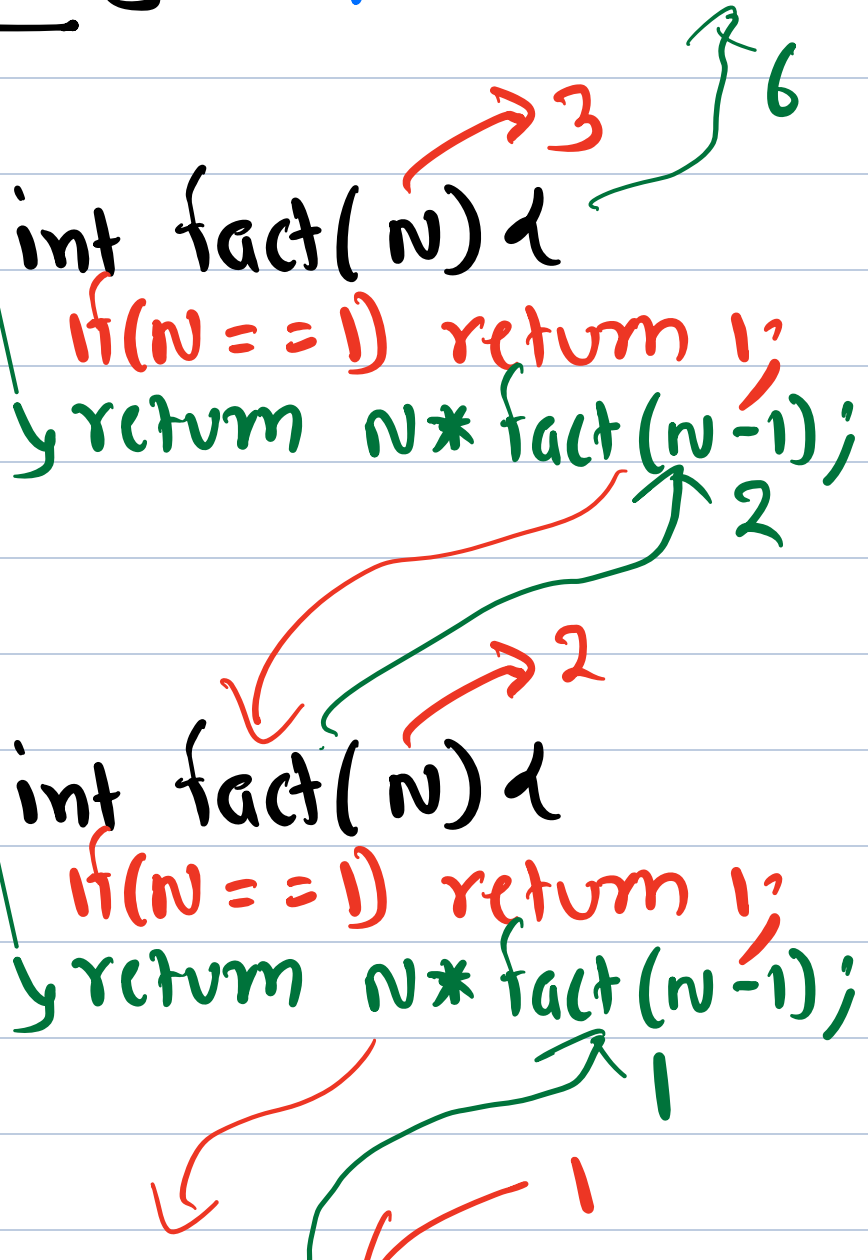
Base case: if (N == 1)

{ return 1; }

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

6

Call Tracing: print(fact(3));



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

Q) Print number in increasing order from 1 to N.

Eg. N = 5 1 2 3 4 5

Quiz :

Solution:

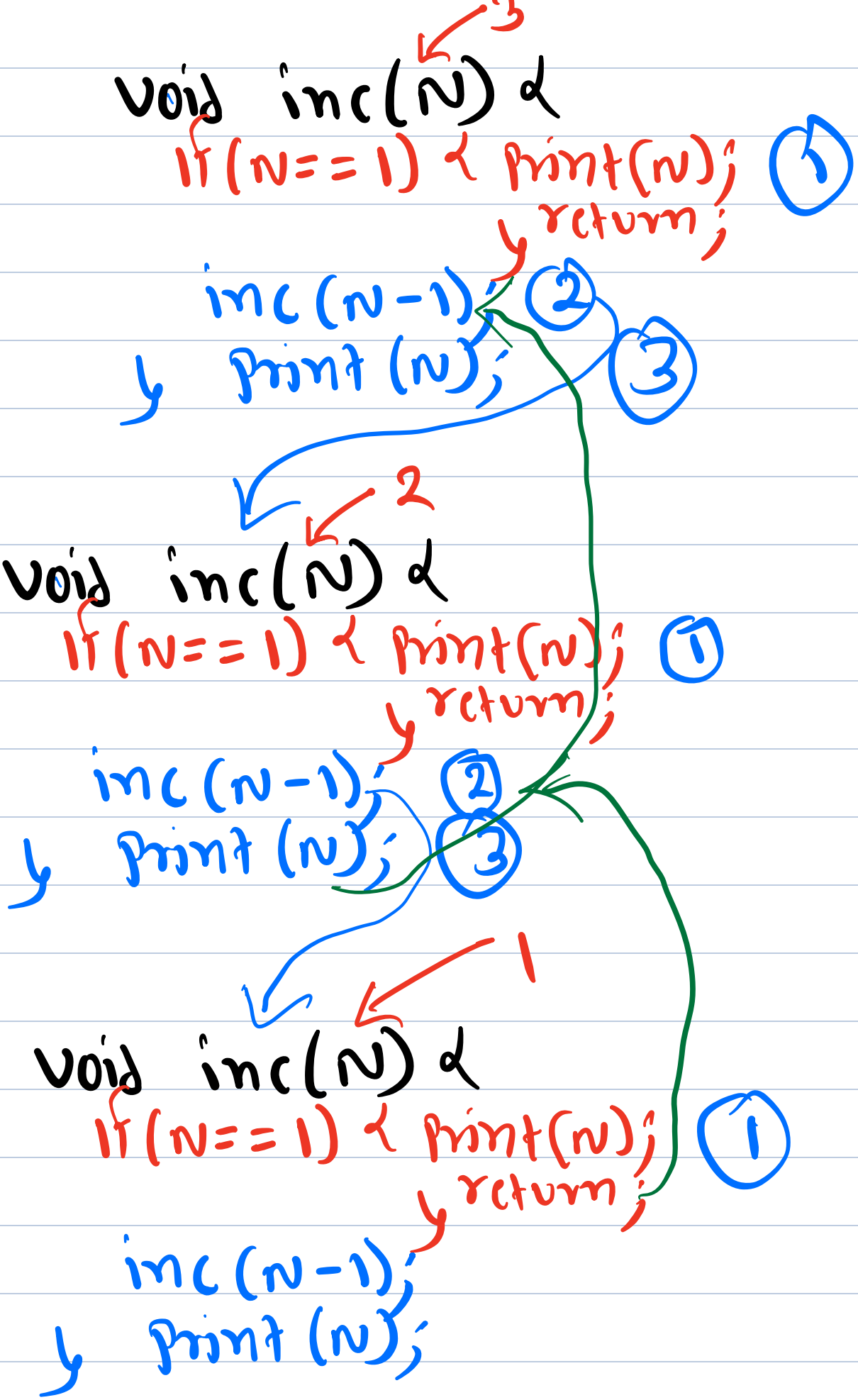
Assumption: Create a function which takes N as an input and print all no. from 1 to N.

Main logic :

inc(N) = inc(N-1) { print(N);

Base case: if(N==1) { Print 1; return }

```
void inc(N) {
  if(N==1) { print(N); return;
  inc(N-1);
  Print(N);
}
```


3
void inc(N) {
if(N==1) { Print(N); return; }
inc(N-1);
Print(N);
1
2
3
2
void inc(N) {
if(N==1) { Print(N); return; }
inc(N-1);
Print(N);
1
2
3
1
void inc(N) {
if(N==1) { Print(N); return; }
inc(N-1);
Print(N);
1

Output: 1 2 3

9)

Problem

Whirlpool wants to design a timer for their **washing machines**. This feature is a simple countdown timer. When a user sets a time, for example, 10 minutes, the washing machine needs to show each minute passing, counting down until it reaches 0.

Your task is to write a program that takes an integer **A** (the time in minutes set by the user) and then prints out each minute as it counts down to **0**.The requirement is that after a user set a timer for the washing machine for some time say **A** , the washing machine should display each minute after that decremented one by one till the time becomes **0**.

Simplified Problem statement

Given **N**, print all numbers from **1 to N** in decreasing order.

Eg. N=5 5 4 3 2 1

```
void des(N){
  if(N==1){ print(N); return; }
  print(N);
  des(N-1);
}
```

TC: O(N)

SC: O(N)

Time and Space Complexity of Recursion

Time complexity = O(no. of function call * time per function call)

Space complexity = O(maximum depth of recursion tree / stack space + stack per fun call)

Factorial of number

Code

```
int fac(int n){
  if(n==0){
    return 1;
  }
  return n*fac(n-1);
}
```

N = 5

fac(5)

↓

5 * fac(4)

↓

4 * fac(3)

↓

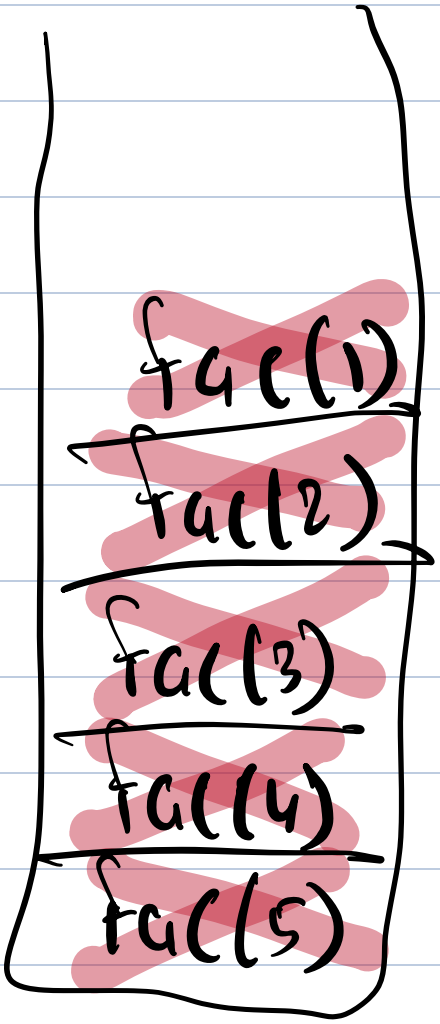
3 * fac(2)

↓

2 * fac(1)

↓

fac(1)



TC: O(n)

SC: O(n)

Q) Printing number in Decreasing Increasing order.

Eg. N=1 → 1 1

N=3 → 3 2 1 1 2 3

N=5 → 5 4 3 2 1 1 2 3 4 5

DesInc(5) = print(5) + des(Inc(4) + print(5)

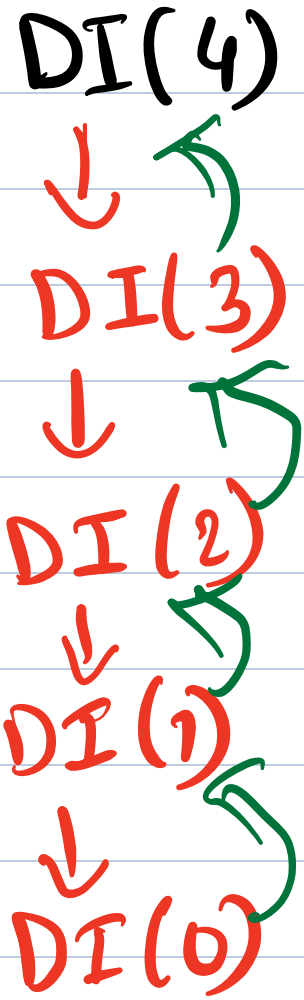
↓ subproblem

Output: 4 3 2 1 1 2 3 4

Code:

```
function desinc(n)
{
    if(n==0) return;

    print(n);
    desinc(n-1);
    print(n);
}
```



Break:
10:30

Q) Printing N^{th} number in fibonacci series.

Series: 0 1 1 2 3 5 8 13

0 1 2 3 4 5 6 7

N=6

Quiz N=7.

$$fib(7) = fib(5) + fib(6)$$
$$= 5 + 8$$
$$= 13$$

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

Solution:

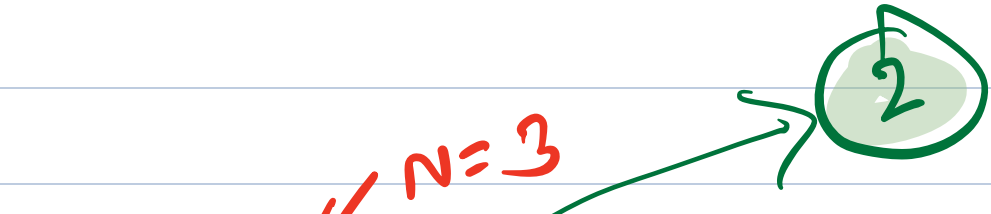
Assumption: Create a fun which takes a integer N as input and return Nth term of fibonacci series.

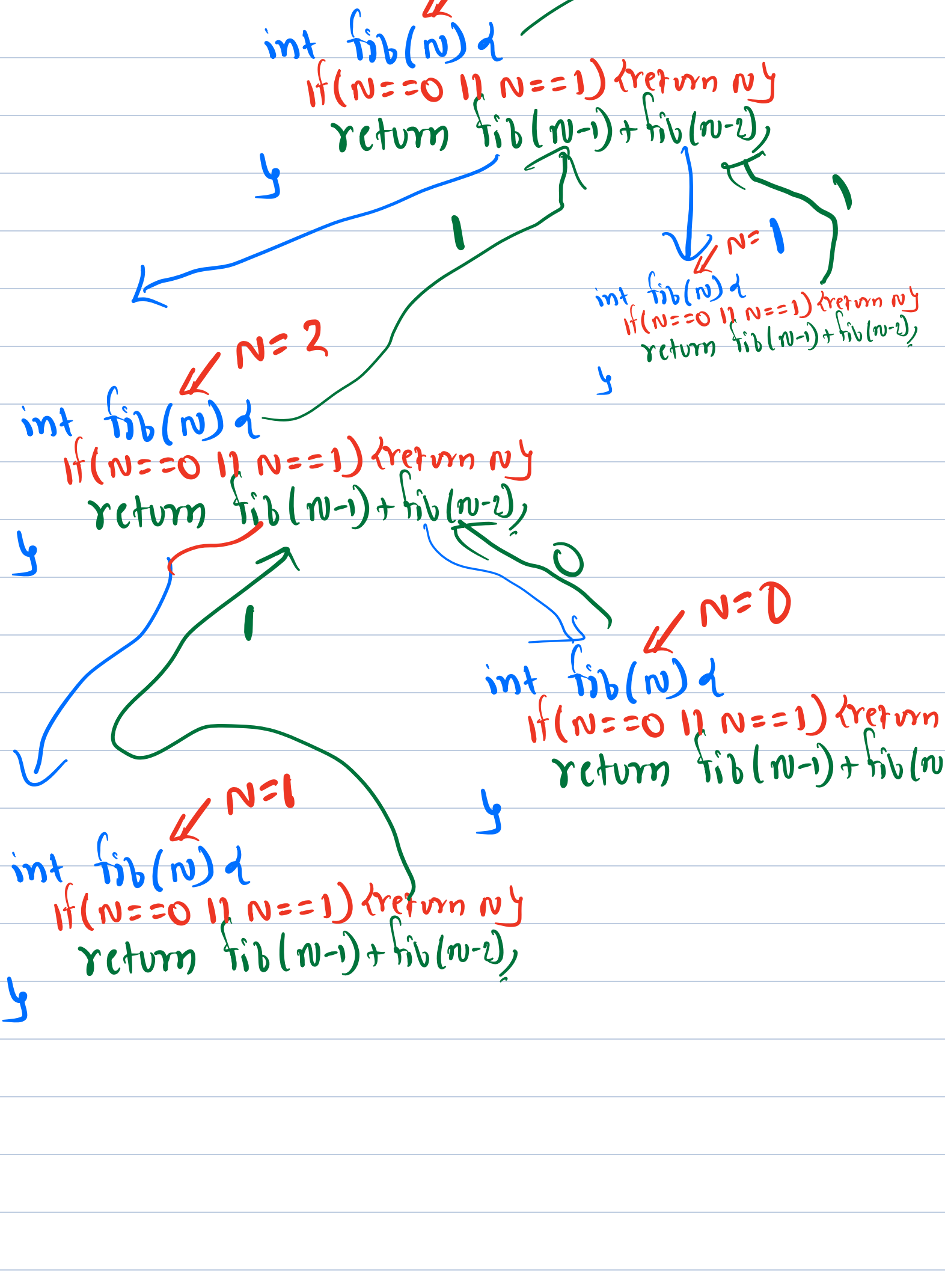
Main logic:

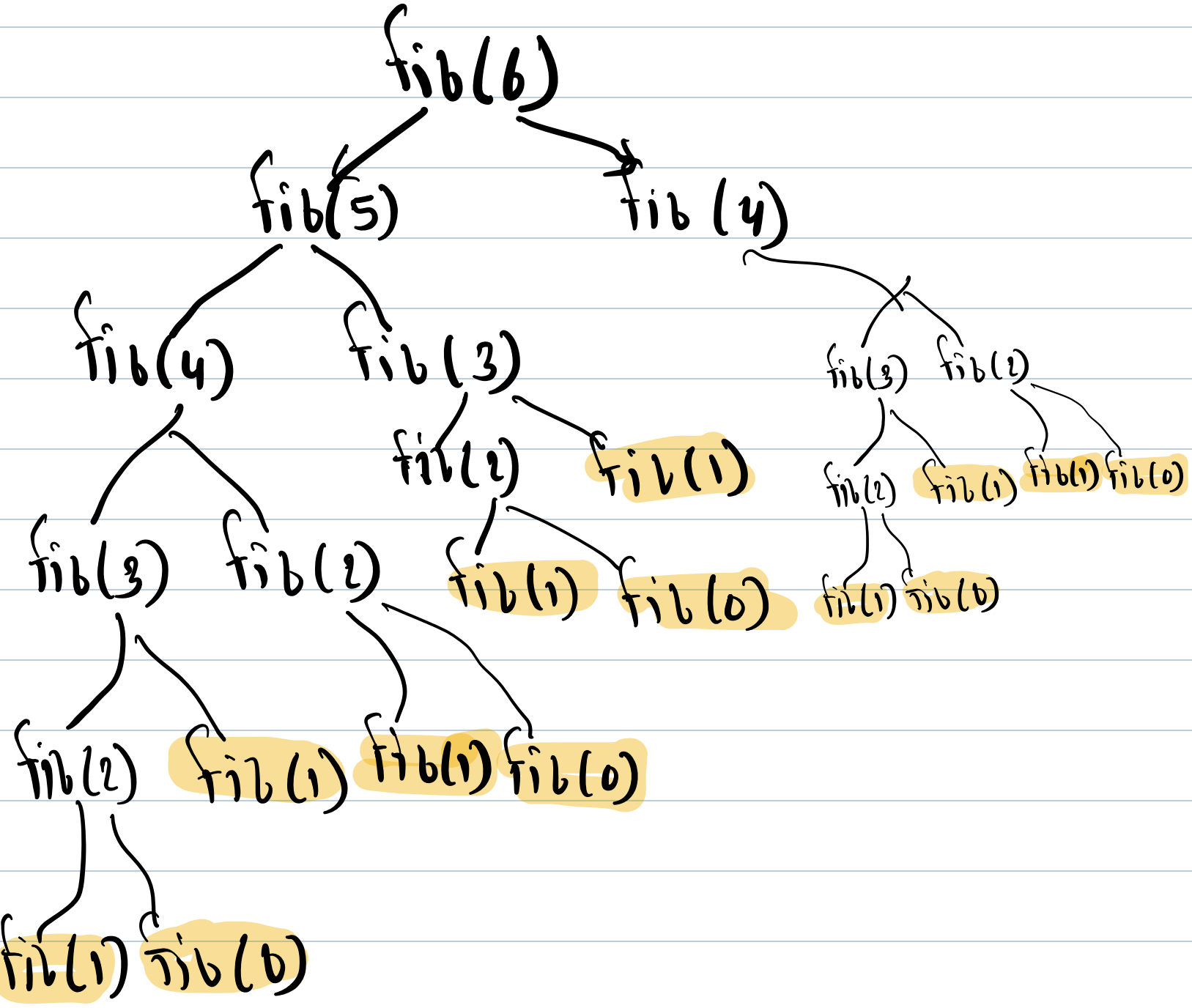
$$fib(N) = fib(N-1) + fib(N-2)$$

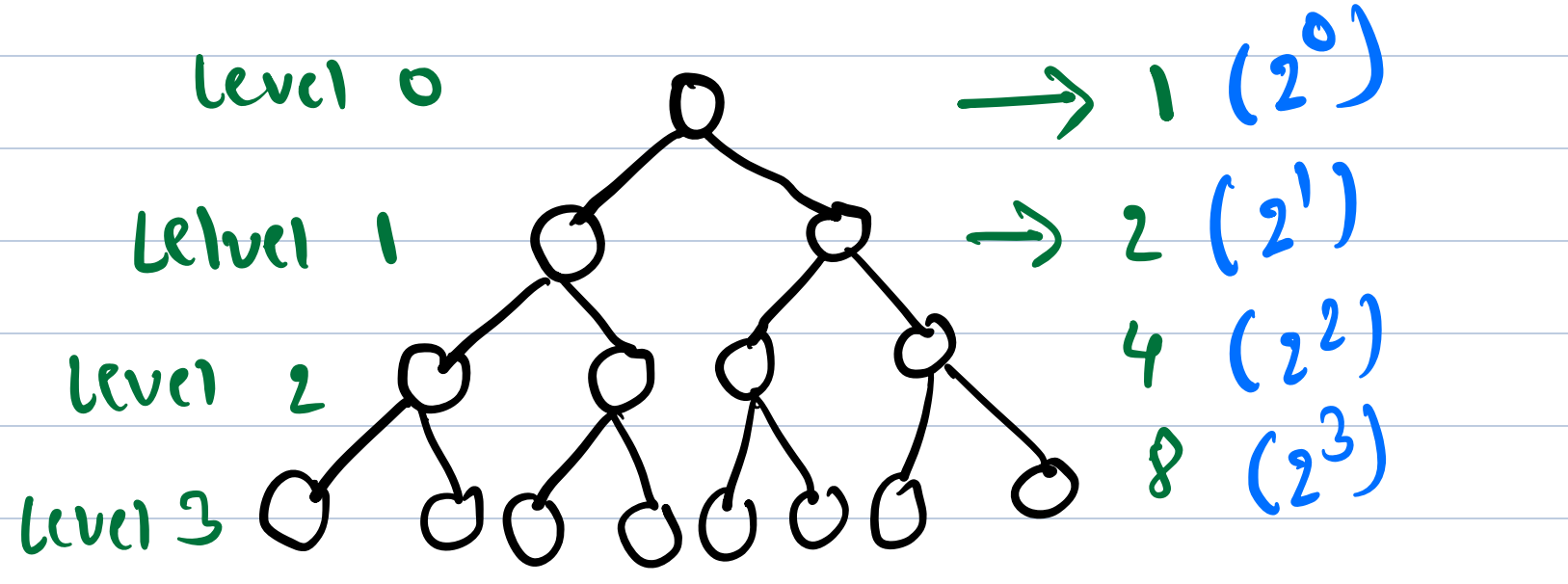
Base case:

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




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


fib(6)
fib(5)
fib(4)
fib(4)
fib(3)
fib(3)
fib(2)
fib(3)
fib(2)
fib(2)
fib(1)
fib(1)
fib(0)
fib(2)
fib(1)
fib(1)
fib(0)
fib(1)
fib(0)
fib(3)
fib(2)
fib(1)
fib(1)
fib(0)
fib(2)
fib(1)
fib(0)


Level 0
Level 1
Level 2
Level 3
→ 1 (2^0)
→ 2 (2^1)
4 (2^2)
8 (2^3)

Level 4 16 (2^4)

Level 5 32 (2^5)

$$2^0 + 2^1 + 2^1 + 2^2 \quad \cdots \quad 2^t$$

$$\frac{a(r^N - 1)}{(r-1)}$$

$$= \frac{1 * (2^N - 1)}{(2-1)}$$

$$= 2^N - 1$$

TC $\approx O(2^N)$

SC $= O(N)$

max stack space = 0 1 2 3 4 5

current stack space = 0 1 2 3 4 5 4 5 4 3 4 5 4 3 2 3 4 3 2 1