"Computer Science is not really about computers and it's not about computers in the same sense that astronomy is not about telescope"

music: "In Motion" from The Social Network movie

§. Why we should recursion?

① Recursion makes it easy to solve certain types of problem

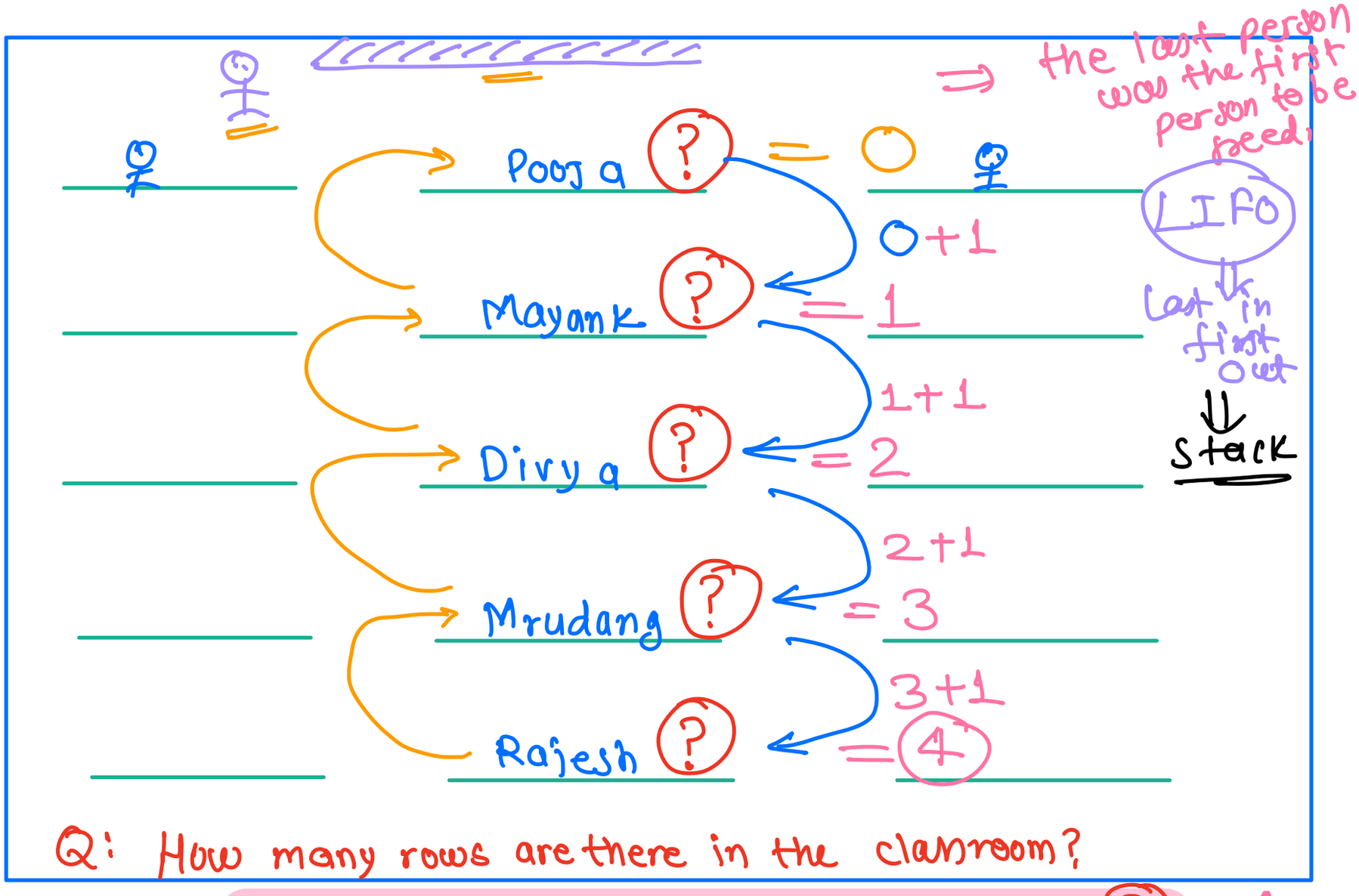
② A core foundation of problem solving (linked list, trees, graphs, DP)

③ Important for interviews.

→ function calling itself: recursive function

Recursion: Recursion is a problem solving method in which we solve a bigger problem by solving smaller instanies of the exact same problem.

(Reduce & conquer)



Q: How many rows are there in the classroom?

= How many rows are in front of me ⇒ ? = 4

\+ my own row (he is at the last row)

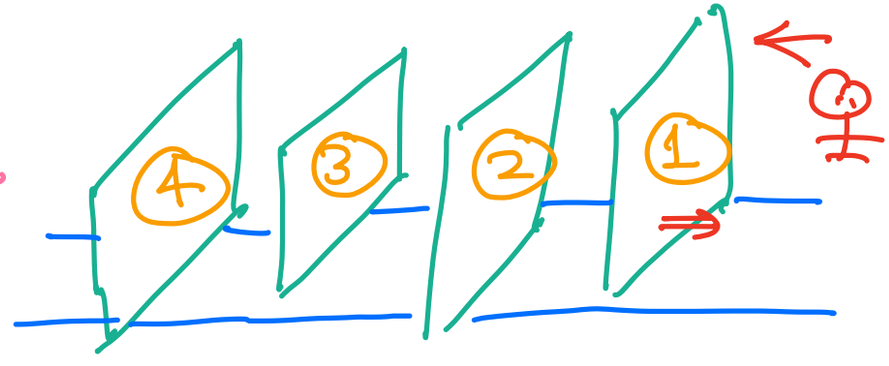
1

$= 5$ $\leftarrow$ $+1$

How many rows are in front of the person sitting in front of me $+1$

Russion doll video

Q. Mathematical induction:



Obs^{n}: (1) If we push plate #1, it will fall

(2) If a plate is falling then the plate next to it will also fall

$\Rightarrow$ If we push 1st plate, all the plates will fall.

Obs^n (I) The more cheese you have, the more holes you have.

(II) More holes you have, less cheese you have

$\Rightarrow$ the more cheese you have $\Rightarrow$ the less cheese you have

5. Abstract thinking:

↳ ignore lower level details.

eg when we book a cab:
1. Driver license
2. fuel

nop in-nop out

Recursion is easy: (1) If I have a soln for f(n-1) how do I get a soln for f(n)

(2) what is the base condition

⇓

where we have trivial soln.

Q. Given a number N, find the sum of first N natural numbers.

⇒ N = 1 ⇒ 1 = 1

⇒ N = 2 ⇒ 1, 2 = 3

⇒ N = 3 ⇒ 1, 2, 3 = 6

N = 4 6 + 4 = 10

sum(N+1) = sum(N) + (N+1)

base condition :

sum(4) = sum(3) + 4

sum(2) + 3

sum(1) + 2

1

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

Q. Given a number N, find factorial of N.

Ex.

$1! = 1$

$2! = 2 \times 1$

$3! = 3 \times 2 \times 1$

$4! = 4 \times 3 \times 2 \times 1$

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

$0! = 1 = ?$

$1! = 1 \times 0!$

ABC (ABC, ACB, BAC, BCA, CAB, CBA)

$= 3!$

$3 \times 2 \times 1$

$n!$ = ways to arrange N items.

⇒ 0 items in how many ways 9 can arrange them
= 1

⇒ {1,2,3} ⇒ [{∅} {1}, {2}, {3} ...]

Size 3 → 1
size 2 ⇒ 2
size 1 ⇒ 3
size 0 ⇒ 1

fact(N) = fact(N-1) * N

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

N	ans
0	1
1	1
:	:
:	:

§ Function call tracing: involves tracing the sequence of function calls that are made when a program is executed.

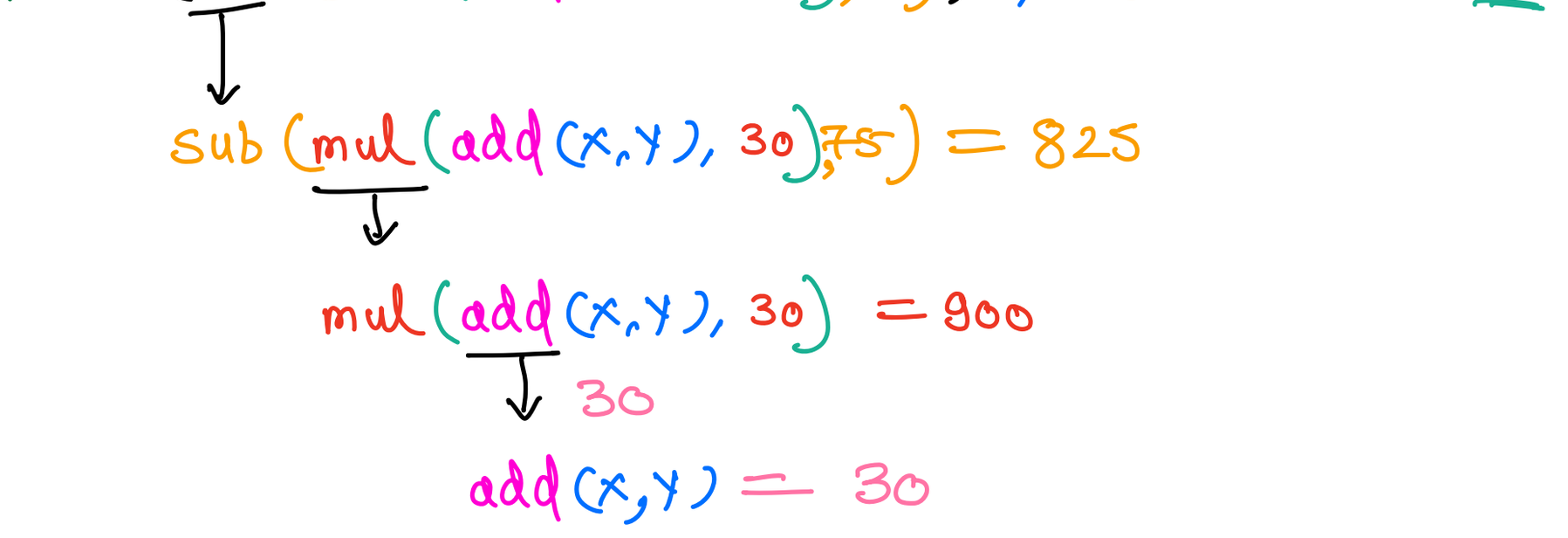
ex:

```
int add (x,y) {
        return x+y;
}
int mul (x,y) {
        return x*y;
}
int sub (x,y) {
        return x-y;
}
void print (x) {
        print (x);
}
```

```
int main () {
    int x = 10
    int y = 20

    print (sub (mul (add (x, y), 30), 75)) ;
    return 0;                          // successful
}
```

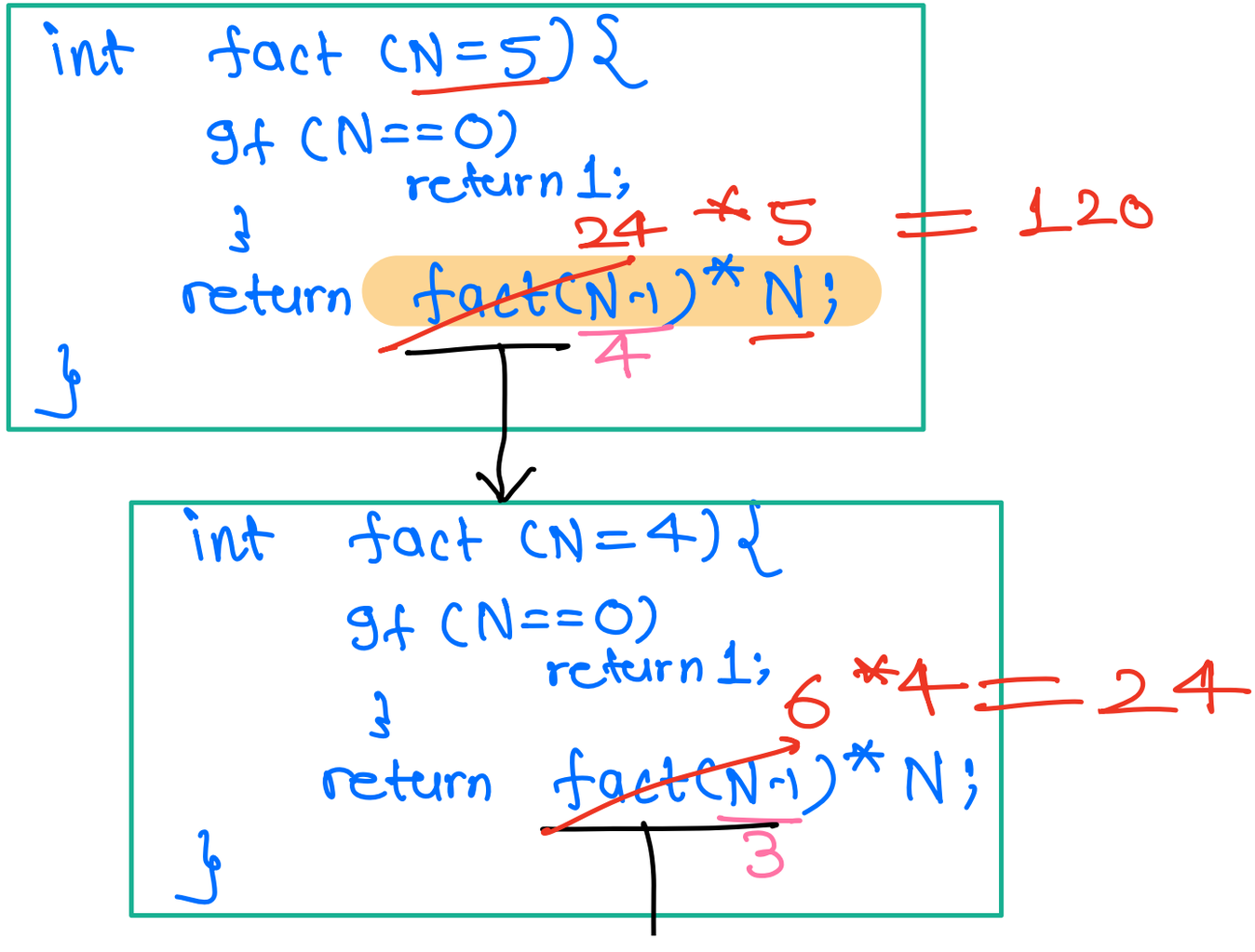
print (sub (mul (add (x, y), 30), 75)) ; 825 // on the console

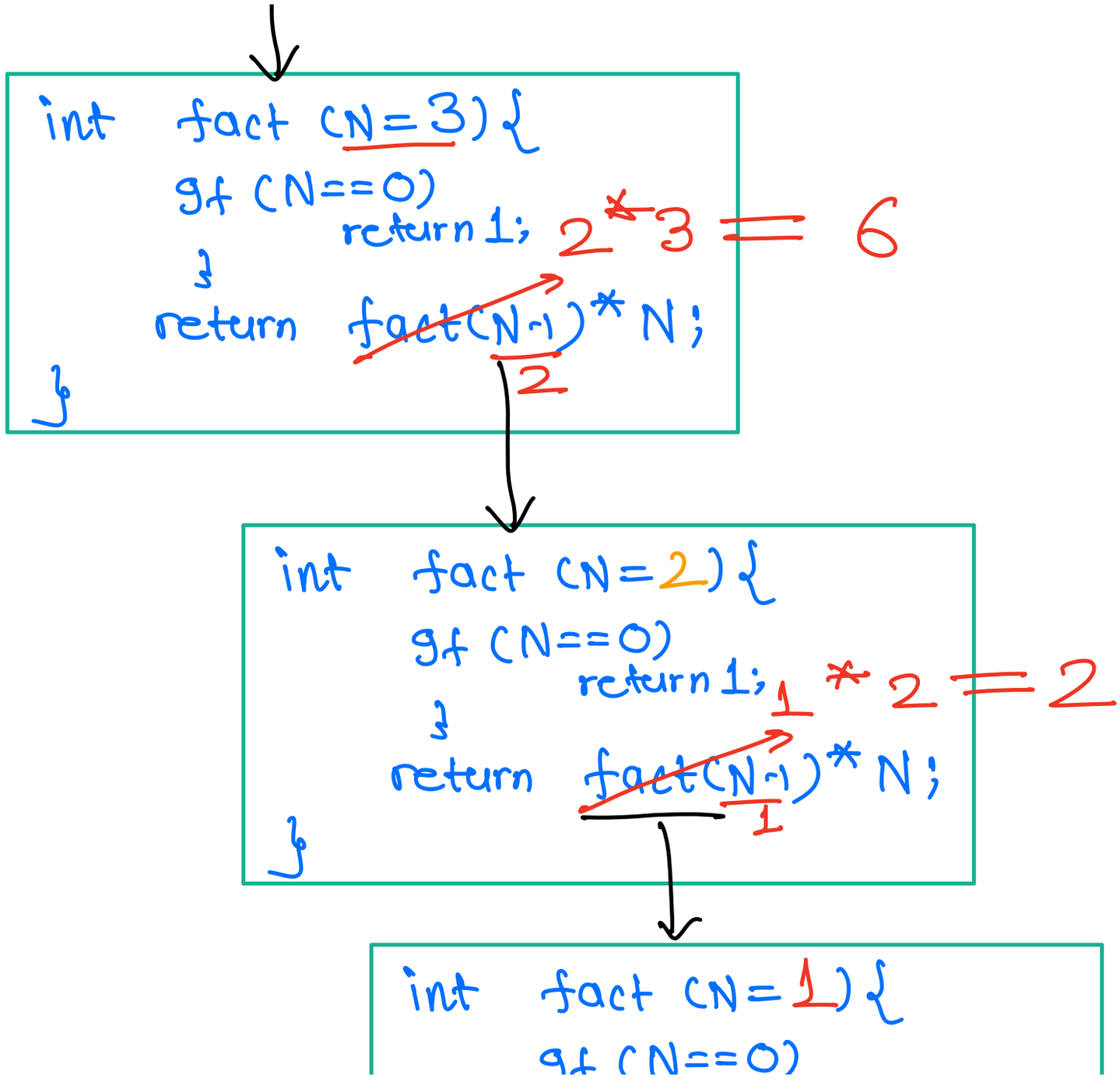


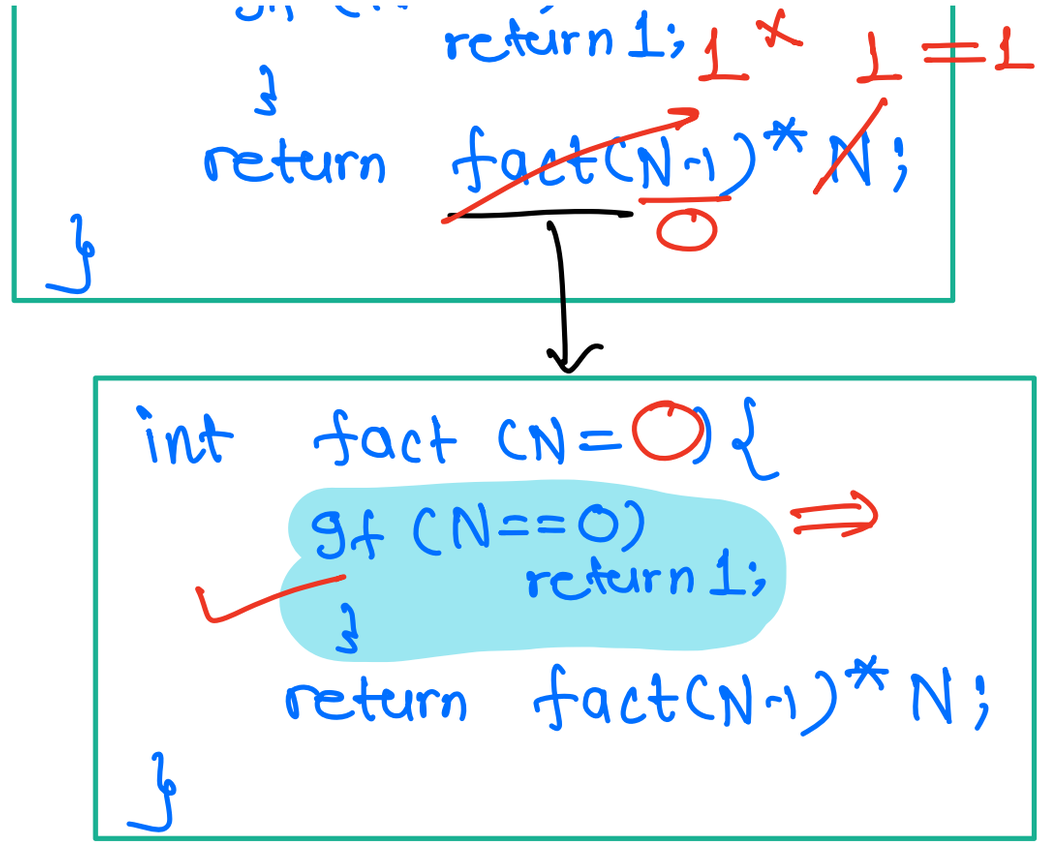
⇒ Recursion uses stack to keep track of

Smaller function calls value

8. function calls




int fact (N=3){
if (N==0)
return 1;
}
return fact(N-1)*N;
}
2*3 = 6
2
int fact (N=2){
if (N==0)
return 1;
}
return fact(N-1)*N;
}
1 * 2 = 2
1
int fact (N=1){
if (N==0)



Obsⁿ: In recursive functions, if base condition is not there or not possible to reach, then we will get MLE

why: infinite copies of function call.

↓

Stack overflow

NCERT class XI mathematical induction:

(I) $f(0)$ is true

(II) If $f(n)$ is true then $f(n+1)$ is also true;

$\Rightarrow$ we conclude $f(n)$ is true for all number $(0, \infty)$

10:37

Q. Fibonacci numbers: The fibonacci sequence is a series of numbers in which each number is the sum of two previous numbers.

N = 0, 1, 1, 2, 3, 5, 8, 13..
index 0 1 2 3 4 5 6, 7

$f(7) =$

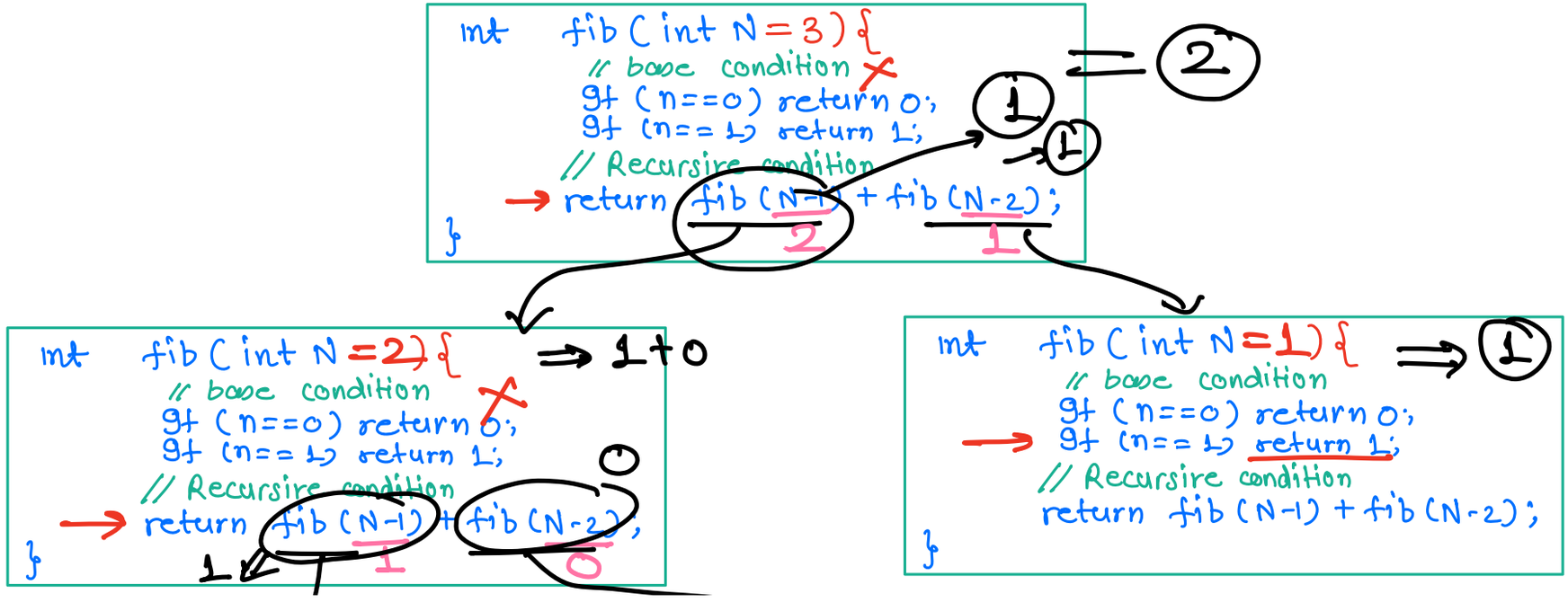
Soln: (1) $f(n) = f(n-1) + f(n-2)$

$fib(0) = 0$
$fib(1) = 1$

```
int fib(int N){
  // base condition
  if(n==0) return 0;
  if(n==1) return 1;
 // Recursire condition
 return fib(N-1) + fib(N-2);
}
```

fib(2) = f(0)+f(1)
f(3) = f(2+f(1)
f(4) = f(4)+f(3)

function call tracking for N = 3



```
int fib(int N=1){
  // base condition
  if (n==0) return 0;
  if (n==1) return 1;
  // Recursive condition
  return fib(N-1) + fib(N-2);
}
```

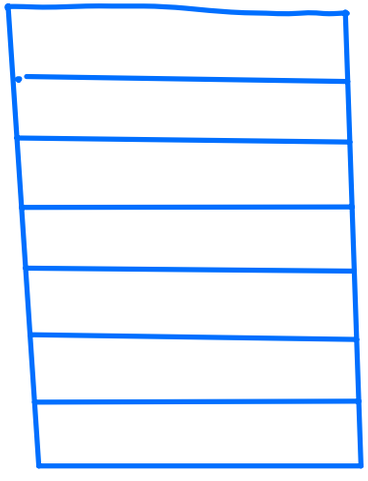
1

```
int fib(int N=0){
  // base condition
  if (n==0) return 0;
  if (n==1) return 1;
  // Recursive condition
  return fib(N-1) + fib(N-2);
}
```

0

3 + 5 = ?

function call in the machine:



Stack

f(3) = ? 2

= 1 + 1

= f(2) + f(1);

f(1) + f(0)

1 + f(0) → 0

= 1 + 0

Q: Given a positive number N, write a recursive function to print all numbers from 1 to N in increasing order.

N = 1 ⇒ 1

N = 2 ⇒ 1 2

N = 3 ⇒ 1 2 3

N = 4 ⇒ 1 2 3 4

N = 5 ⇒ 1 2 3 4 5

ex: N = 6 ⇒ 1 2 3 4 5 6

idea:
1. Start from N
2. assume N-1 numbers are already printed
3. print N

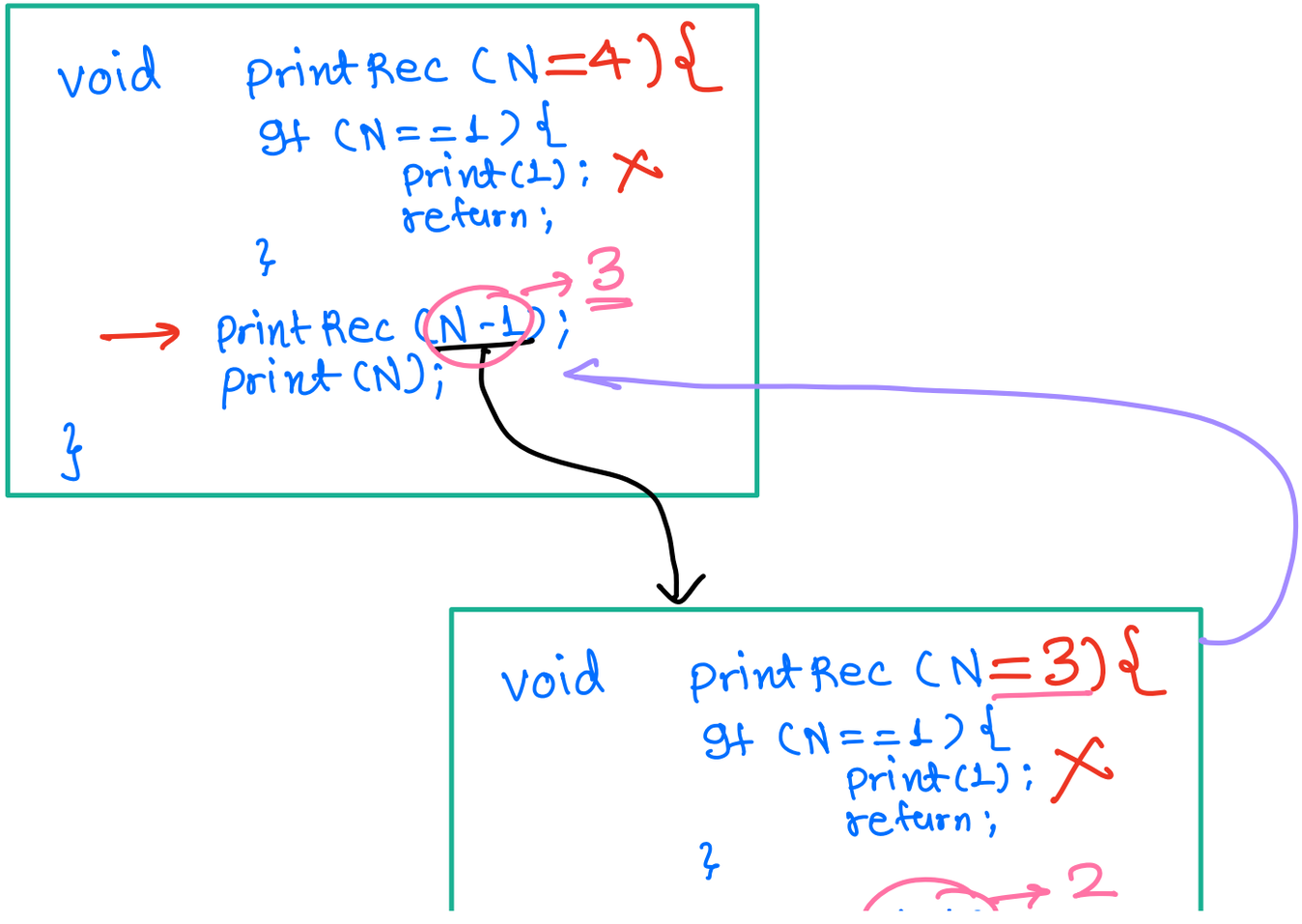
Obsⁿ: If we have the solⁿ for f(n-1) ⇒ we have the solⁿ for f(n)

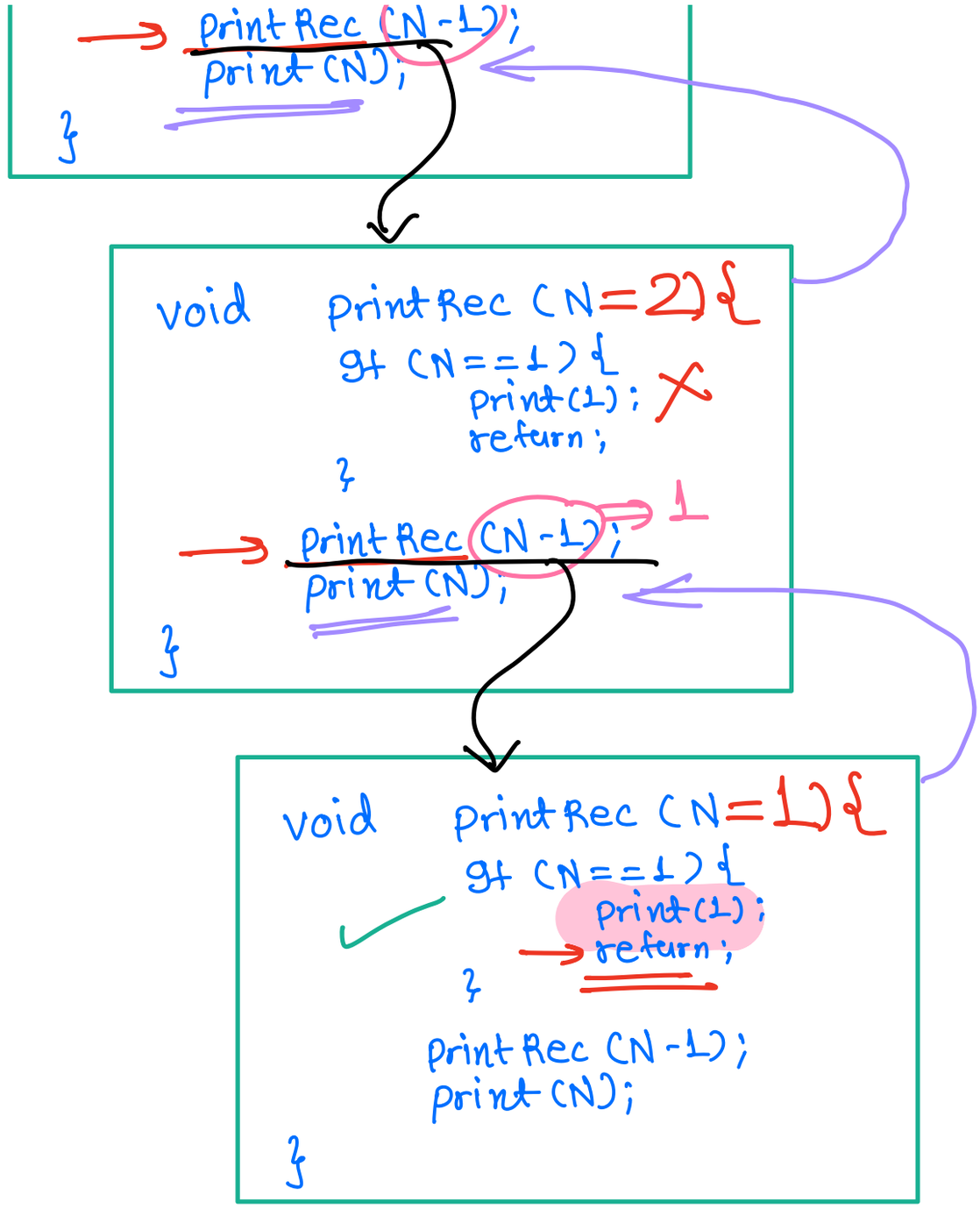
```
void PrintRec(N) {
// Base condition
if (N==1) {
        print(1);
        return;
}
// Recursive condition
PrintRec(N-1);        // By magic, it print first
```

```
    print(N);
}
```

N-1 number as expected

Dry Run for N = 4




printRec (N-1);
print(N);
}
void printRec (N=2){
if (N==1){
print(1);
return;
}
printRec (N-1);
print(N);
}
1
void printRec (N=1){
if (N==1){
print(1);
return;
}
printRec (N-1);
print(N);
}

Console >> 1 2 3 4

H.W. ⇒ Given a positive number N, write a recursive function to print all numbers from 1 to N in decreasing order.

Q. Given a string, check if it is a palindrome or not.

ex. S = 'madam', ans = true
S = 'abcdx', ans = false

quiz: (A) raor ✗ (B) deed ✓ (C) maddam ✓ (D) moon ✗ — B & C

quiz (A) radar ✓ (B) level ✓ (C) melon ✗ (d) hello ✗

Soln using recursion:

Objn: How do you reduce the problem size?

ex: St = "racecar" (0 1 2 3 4 5 6) s=0, e=6, s++, e--

Substring	st[s]	st[e]	if st[s] == st[e]
racecar	r	r	✓
aceca	a	a	✓
cec	c	c	✓
e	e	e	✓
-			

St is a palindrome.

idea! ① St. size is N

② check the start & end point
if they are not equal return false

else

check smaller substring is palindrome

initially will be called with (St, 0, N-1)

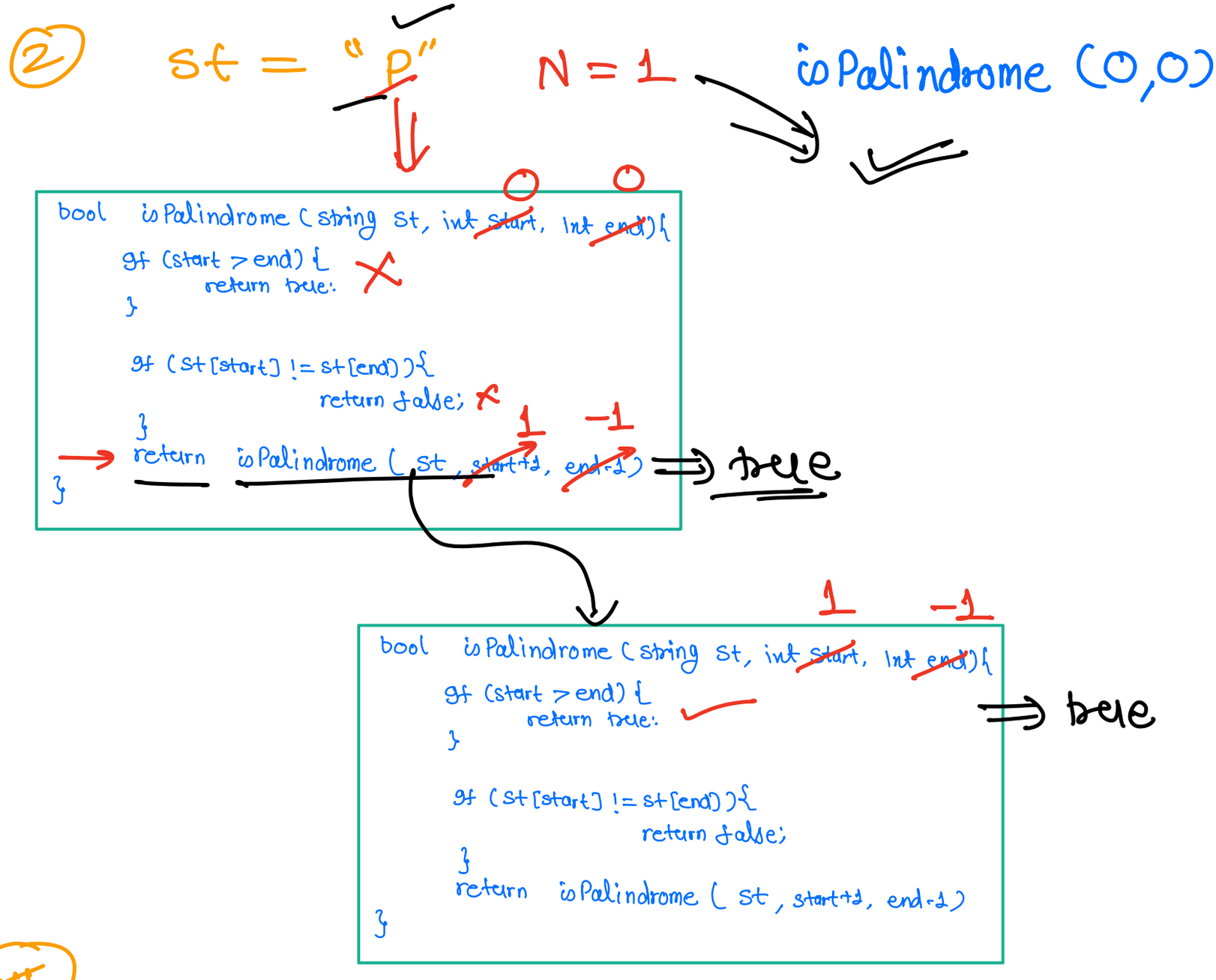
```
bool isPalindrome (string St, int start, int end){
    // Base condition
  // if st is empty return true;
   if (start > end) {
        return true;
   }

   if (St[start] != St[end]){
                return false;
   }
   return isPalindrome ( St , start+1, end-1)
}
```

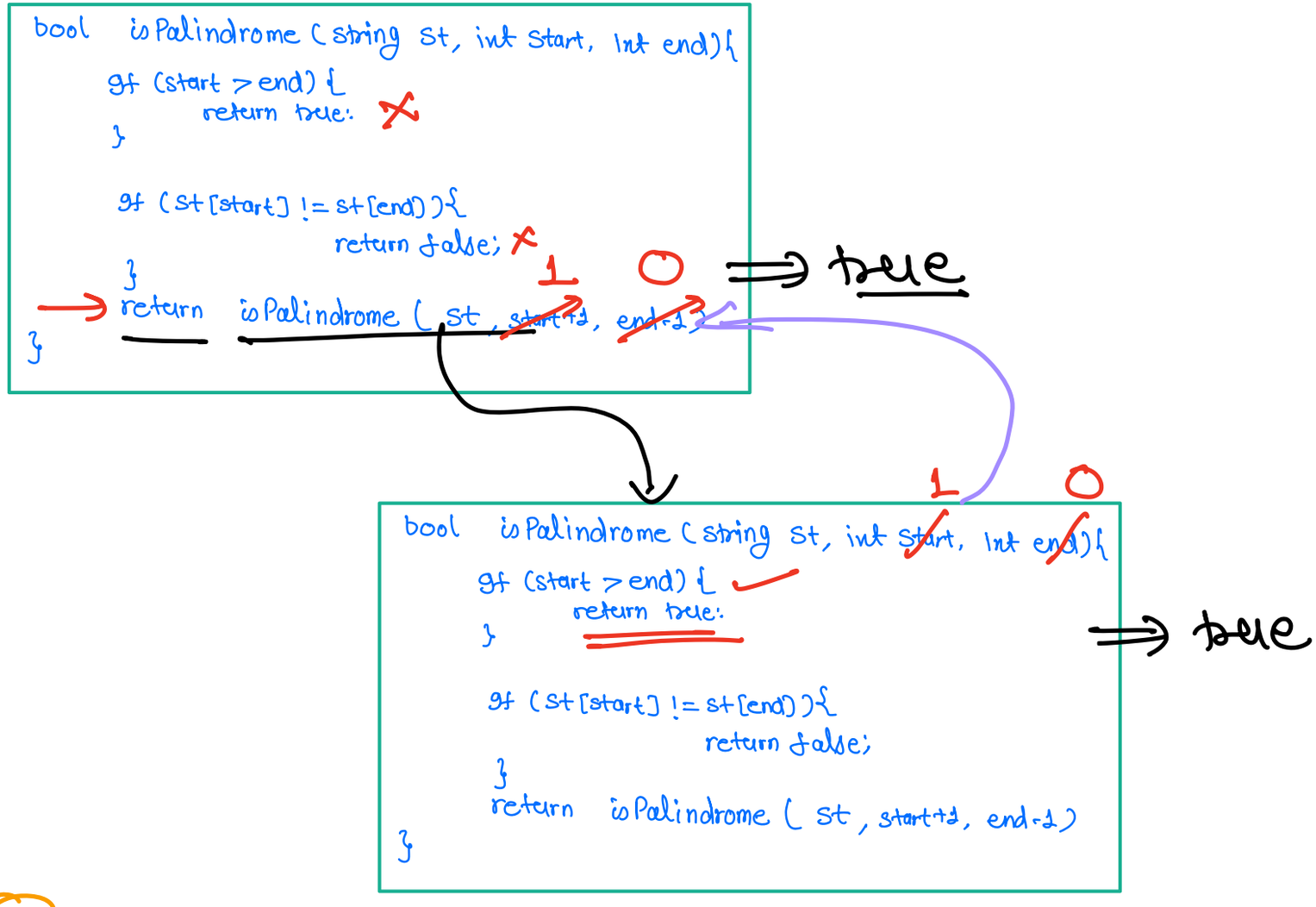
St = "ABC"
0 1 2

st = 0 → 1 → 2
end = 2 → 1 → 0

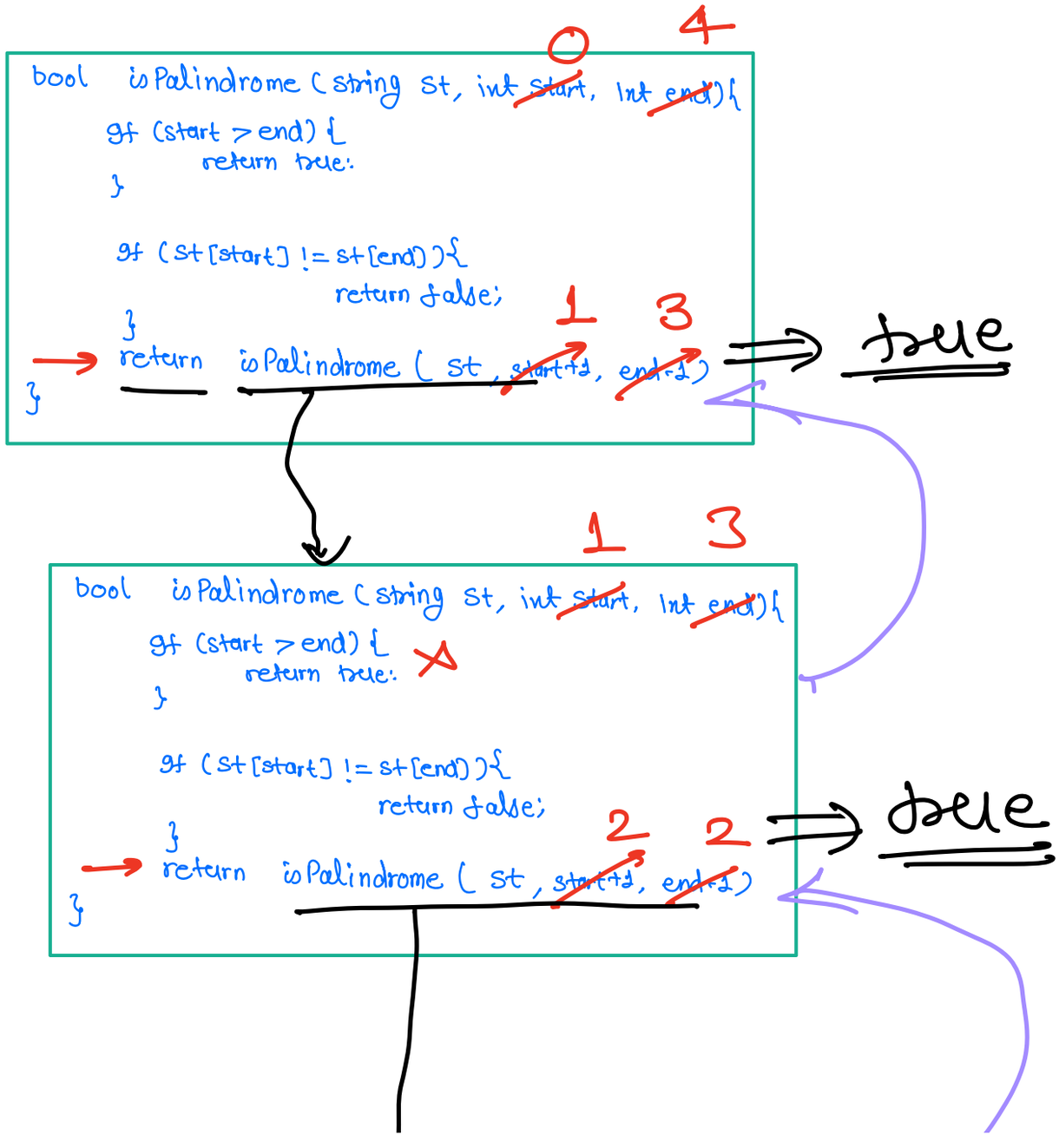
Dry run ① St = " " N=0 (0, N-1) ✓

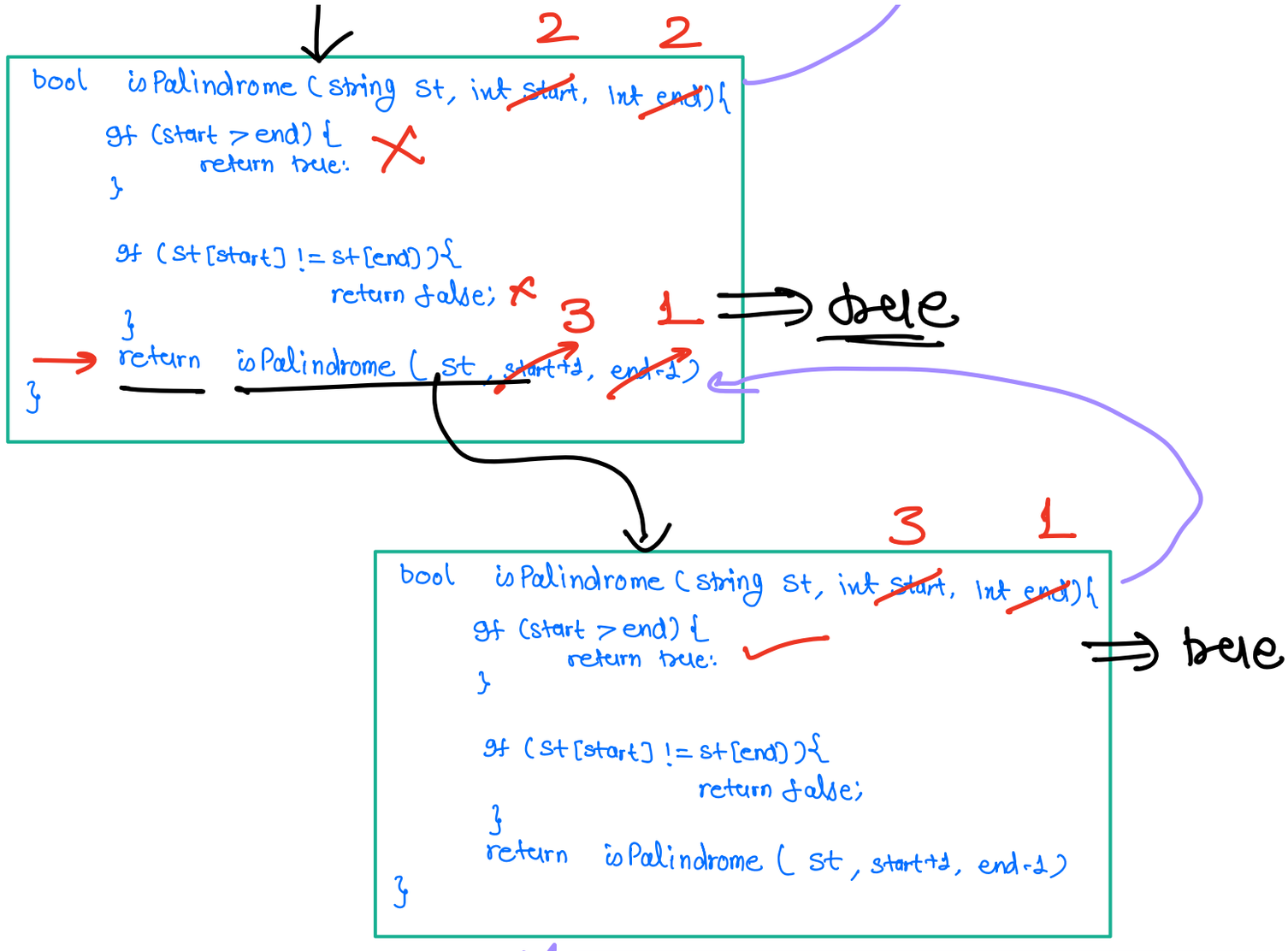

2
st = "p"
N = 1
isPalindrome (0,0)
bool isPalindrome (string st, int start, int end){
if (start > end) {
return true;
}
if (st[start] != st[end]){
return false;
}
return isPalindrome (st, start+1, end-1)
}
0 0
1 -1
true
bool isPalindrome (string st, int start, int end){
if (start > end) {
return true;
}
if (st[start] != st[end]){
return false;
}
return isPalindrome (st, start+1, end-1)
}
1 -1
true
III

st = "QQ" ⇒ isPalin (st, 0, 1)



(IV) st = "radar", N = 5, isPalind. (st, 0, 4)


bool isPalindrome (string st, int start, int end){
if (start > end) {
return true;
}
if (st[start] != st[end]){
return false;
}
return isPalindrome (st, start+1, end-1)
}
0
4
1
3
true
bool isPalindrome (string st, int start, int end){
if (start > end) {
return true;
}
if (st[start] != st[end]){
return false;
}
return isPalindrome (st, start+1, end-1)
}
1
3
2
2
true



1. office hours. 9 PM tomorrow
2. Mathematical induction in the class notes

$f(n-1)$ —how?→ $f(n)$

Q. $O(1)$ ⇒ analysis is required

↑ most cases