Recursion - 1

Content

→ Recursion ?

→ How to write recursive code / tracing

→ TC/SC of recursive codes
↳ next class

Why recursion ?

→ MergeSort / QuickSort

→ Binary tree / BST / segment trees / Tries / Balanced BST

→ Dynamic Programming (DP)

→ Backtracking

→ Graphs

Recursion: function calling itself
↳ solving a problem, using smaller instance of the same problem.
⇓
sub-problem

eg $Sum(N) = \underbrace{1+2+3+\ldots\ldots\rightarrow N-1}_{\downarrow\ Sum(N-1)} + N$

$Sum(N) = \underbrace{Sum(N-1)}_{\text{sub-problem}} + N$

$Sum(4) = Sum(3) + 4 = 10$

$\downarrow$

$Sum(2) + 3$

$\downarrow$

$Sum(1) + 2$

$\downarrow$

1

How to write recursive codes ?

Assumption : fix what your function should do

Main logic : Solving assumption using sub-problem

Base condition : Inputs for which you want to stop recursion.

Question 1 : sum of N natural no.s constraint : N >= 1

```
int sum(N) {
    if(N==1)
        return 1

    return sum(N-1) + N
}
```

Ass.: Given N, calculate & return sum of first N natural no.

Question 2 :

fact(3) = 3*2*1 = 6

fact(4) = 4*3*2*1 = 24

fact(5) = 24*5 = 120

constraint : N >= 1

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

    return f(N-1) *N
}
```

Ass: Given N, calculate & return N!

fact(N) = 1*2*3.....*N-1*N

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

Base condition is must, otherwise

MLE : Stack overflow

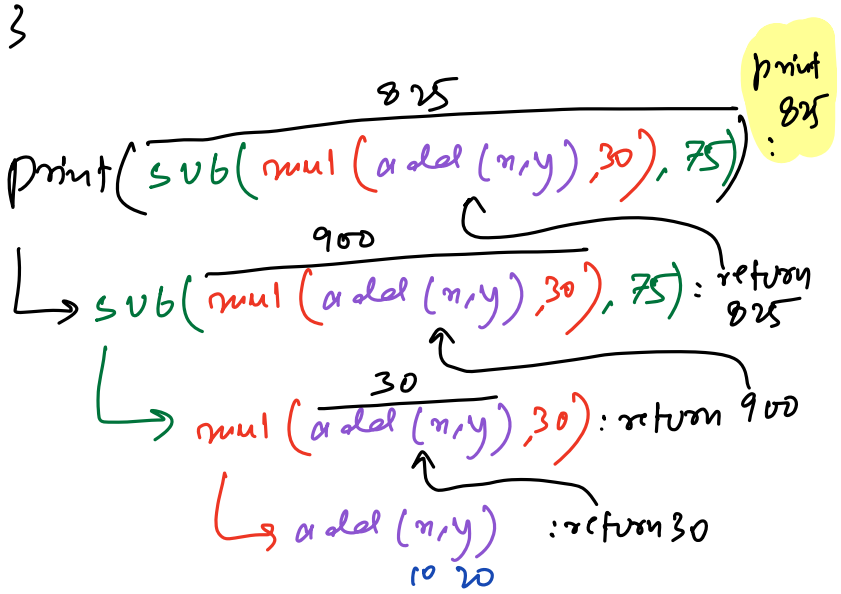
Stack Tracing

```
int add(N,M){
    return N+M
}

int mul(N,M){
    return N*M
}

int sub(N,M){
    return N-M
}
```

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



Data Structure for Stack Tracing ?

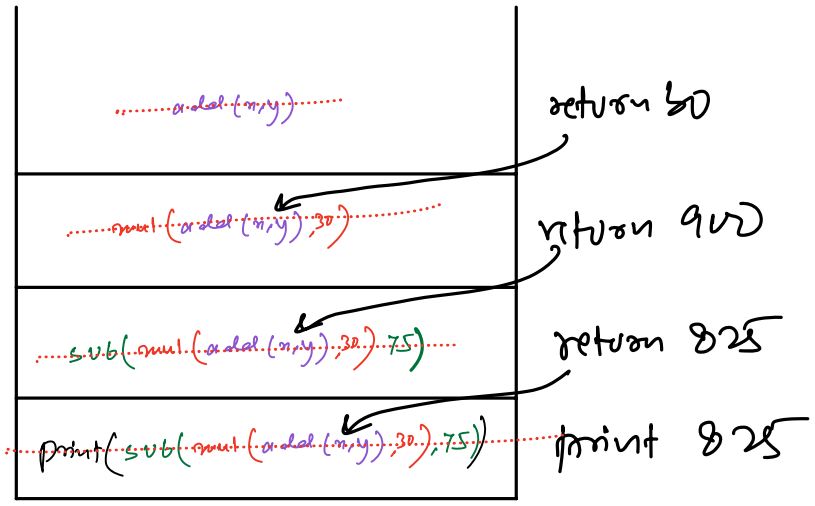
Observation 1 : Whenever a function call happens, we add the function call at top.

Observation 2 : When function returns, we remove it from top.

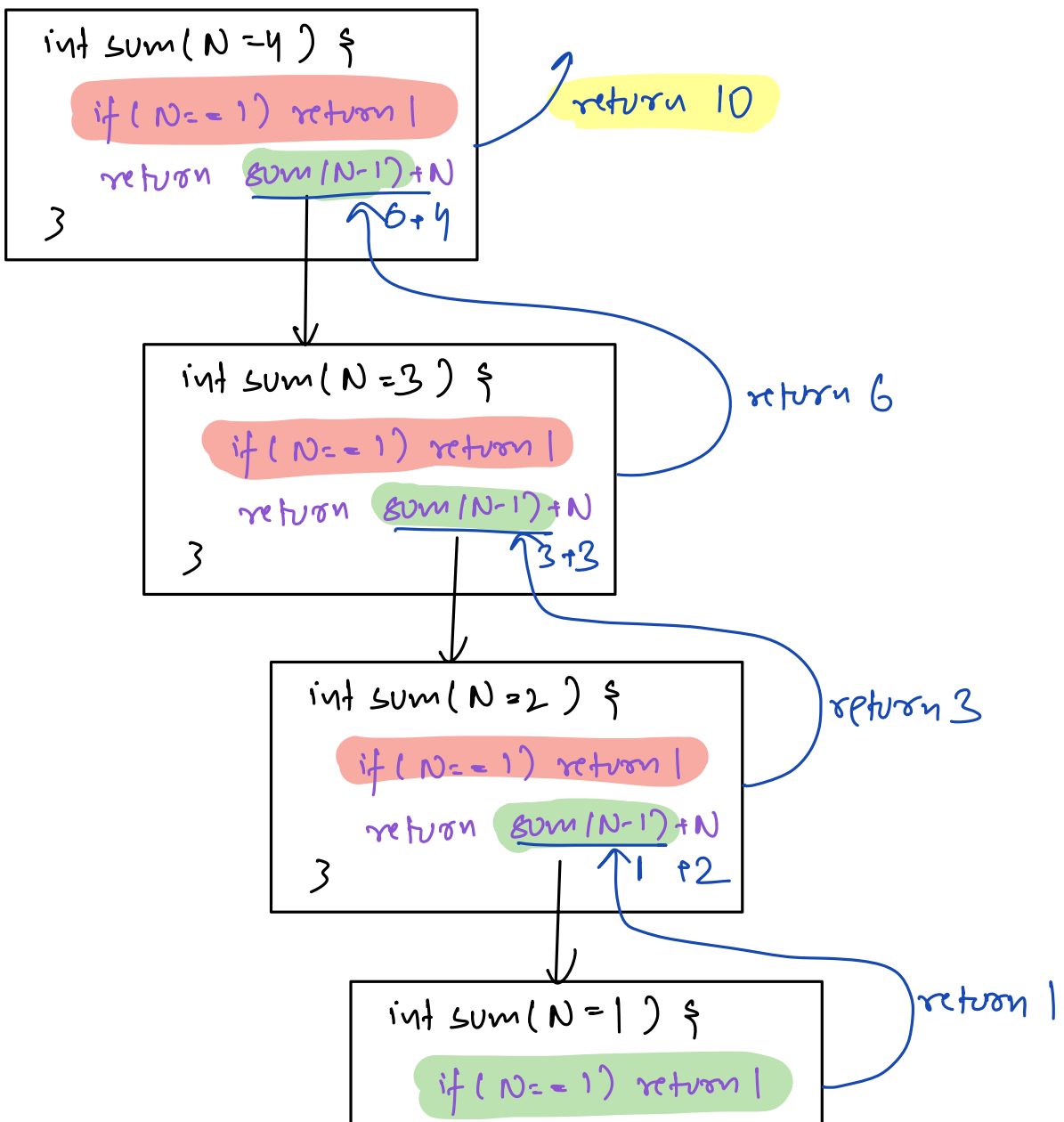
Last In First Out (LIFO)

⇓

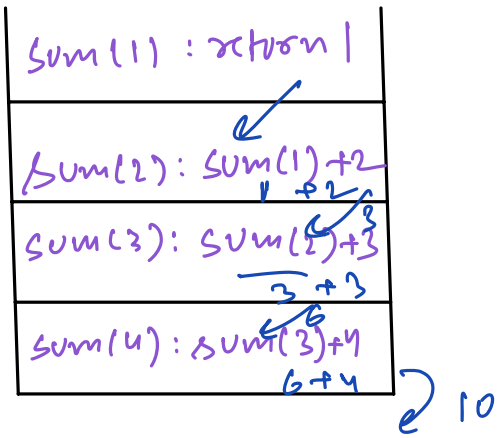
Stack



sum(N) tracing



```
    return sum(N-1)+N
}
```



Question 3

N>=0

Input(N) :	0	1	2	3	4	5	6	7	8
fib() :	0	1	1	2	3	5	8	13	24

N^{th} fibonacci no. = $(N-1)^{th}$ fibo no. + $(N-2)^{th}$ fibo no.

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

Ass.: calculate & return N^{th} fibo no.

→ if(N<=1) return N

Note: How to properly figure out base condition(s)

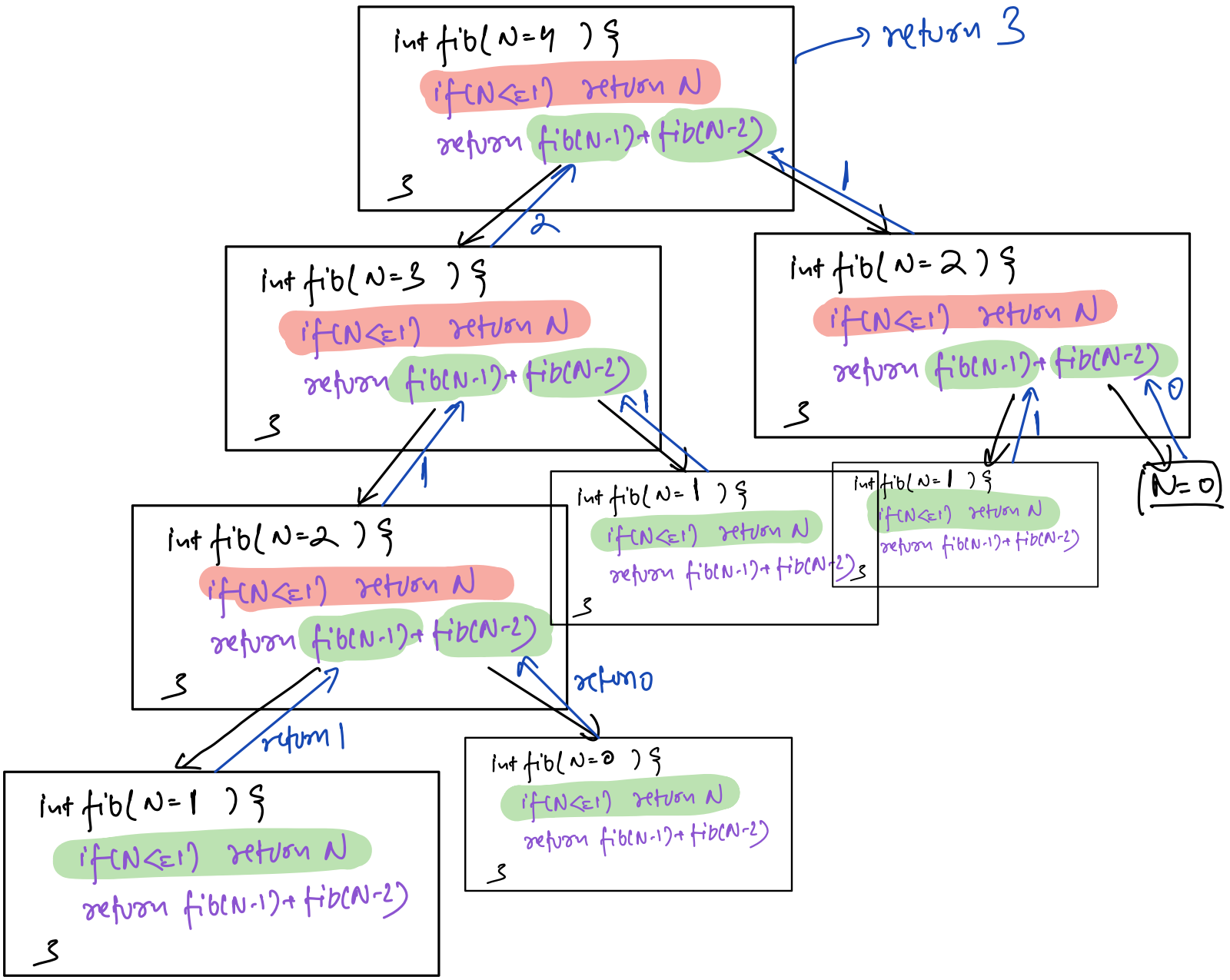
→ for which valid input, expression is invalid

↳ main logic

fib(0) = fib(-1) + fib(-2) ✗

fib(1) = fib(0) + fib(-1) ✗

fib(2) = fib(1) + fib(0) ✓



TODO: Try Stack Tracing with Stack

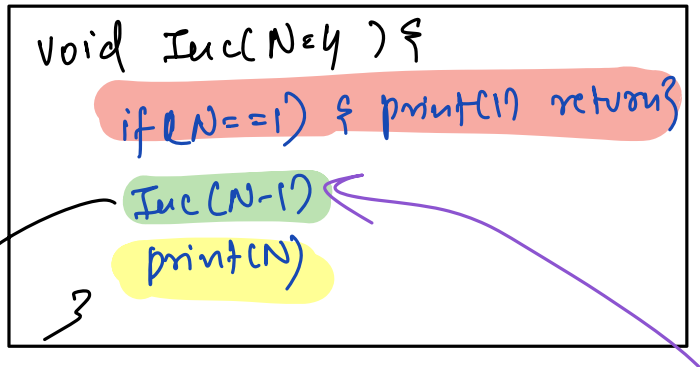
Question 4

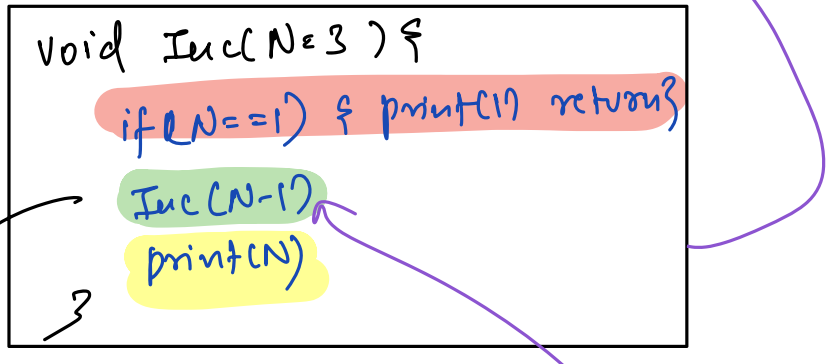
Given N, print all no. from 1-N in increasing order. $N >= 1$

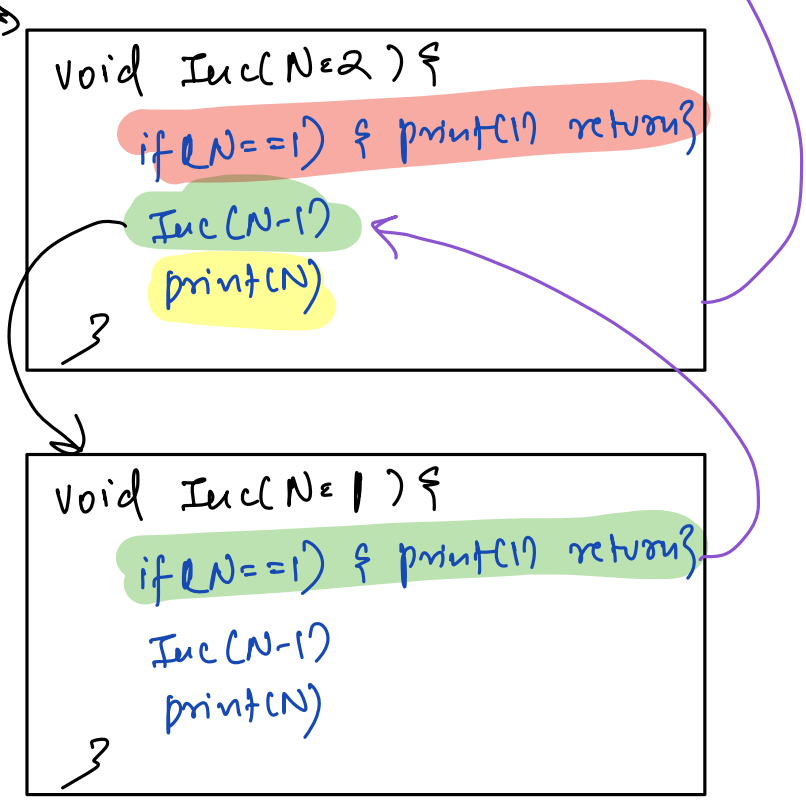
Inc(N) : 1, 2, 3, ... , N-1, N

Inc(N) : Inc(N-1) ; print(N)

```
void Inc(N) {
    if(N==1) {
        print(1) return
    }
    Inc(N-1) → 1 2 3 .... N-1 N
    print(N)
}
```







Output:

print(1) 1
print(2) 2
print(3) 3
print(4) 4

Note:

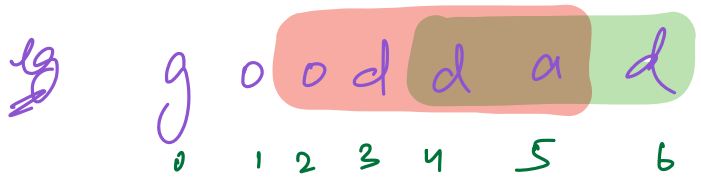
1. Even for void return type, we can return inside function.
2. Once a function completes it will go back to its parent function.

Homework: Print in decreasing order

Dec(N): N N-1 . . . . . 3 2 1

Question 5:

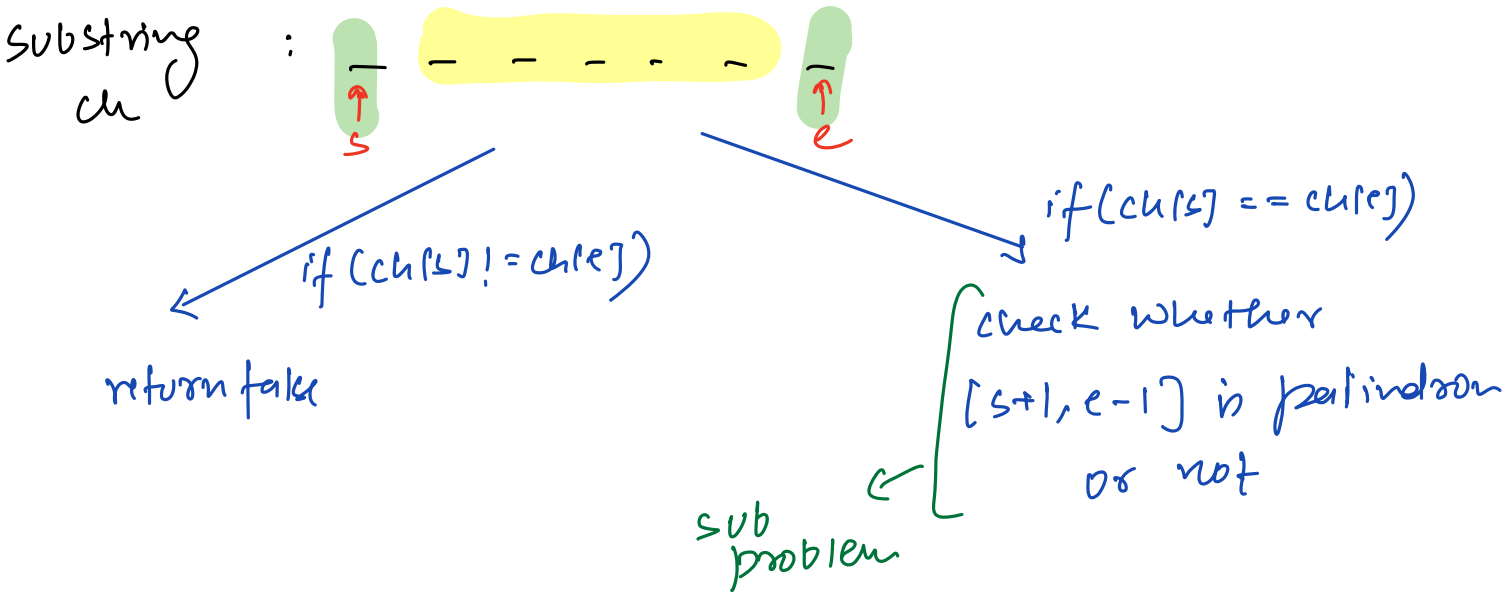
Given a substring, check if its palindrome or not?



s=4, e=6 → return true

s=2, e=5 → return false

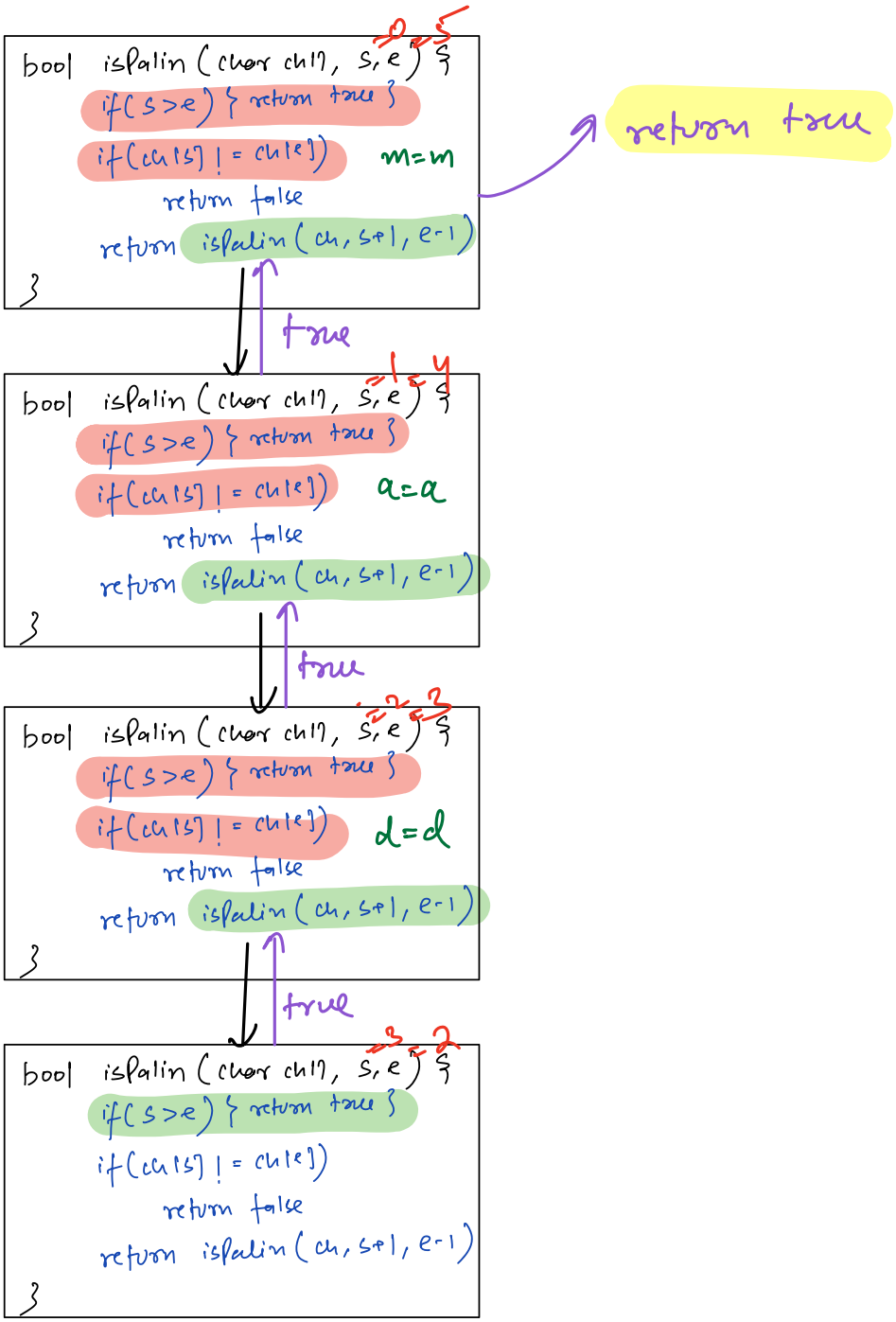
main problem
↳ check whether [s,e] is palindrome or not?




```
bool isPalin(char ch[], s, e) {
    if(s>e) { return true }
    if(ch[s] != ch[e])
        return false
    return isPalin(ch, s+1, e-1)
}
```

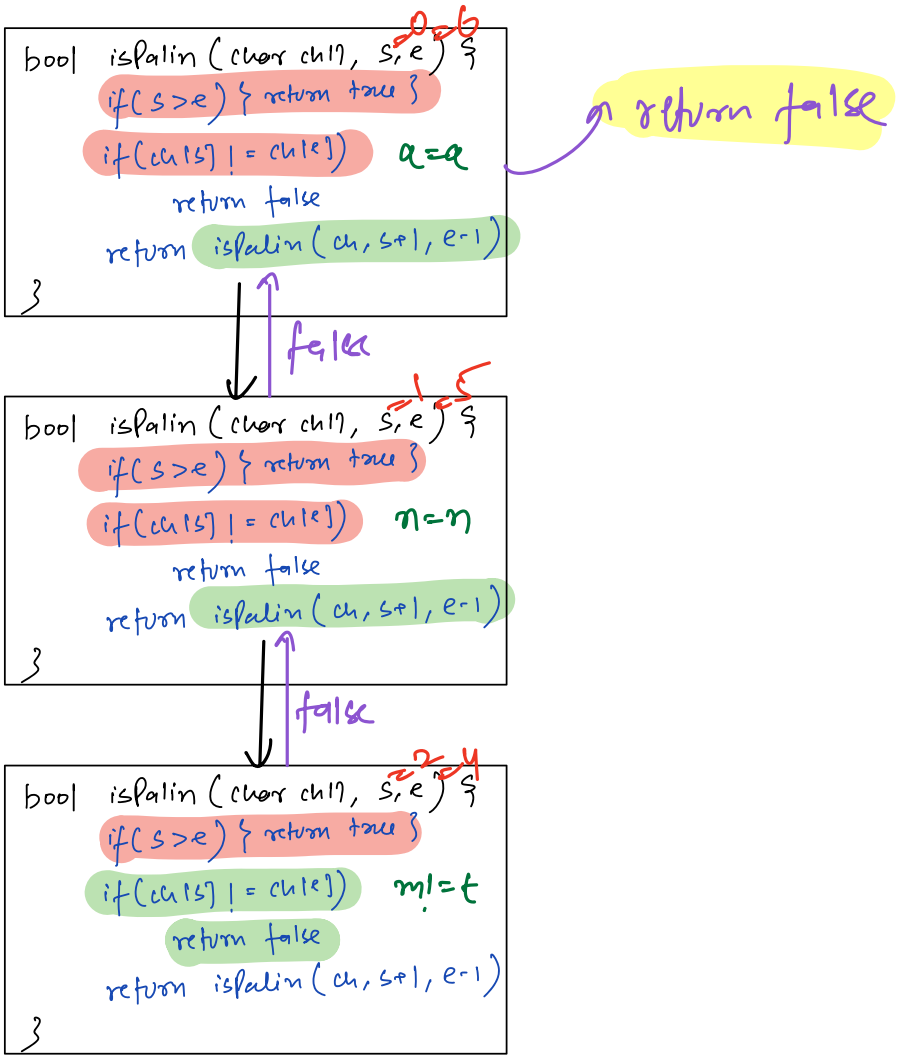
Input : m a d d a m s=0, e=5

0 1 2 3 4 5



Input: a n m e t n a s=0, e=6

0 1 2 3 4 5 6



Doubt

a[i] - a[j] = k

a[j] = a[i] - K