Recursion - 1

Content

→ Recursion ?

→ How to write recursive code / tracing

→ TC / SC of recursion codes
↳ next class

Why recursion

→ MergeSort / Quick Sort

→ Binary tree / BST / B-BST / segment trees / Tries

→ Dynamic Programming (DP)

→ Backtracking

→ Graphs

Recursion : function calling itself

↳ solving a problem, using smaller instance of a same problem.
⇓
sub-problem

eg sum(N) = 1 + 2 + 3 + ... + N-1 + N
⇓
sum(N-1)

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

↳ sub-problem

sum(4) = sum(3) + 4

↓

sum(2) + 3

↓

sum(1) + 2

↓

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

constraint : N >= 1

```
int sum (N) {   Ass.: Given N, calculate & return sum of first N natural no.
    if(N==1)
        return 1
    return sum(N-1) + N;
}
```

Question 2

fact(3) = 3×2×1 = 6

fact(4) = 4×3×2×1 = 24

fact(5) = 24×5 = 120

N >= 1

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

arg.: 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))
}
```

print(sub(mul(add (x,y), 30), 75)) : 825

sub(mul(add (x,y), 30), 75) : 900 : return 825

mul(add (x,y), 30) : 30 : return 900

add (x,y) : 10 20 : return 30

Data Structure

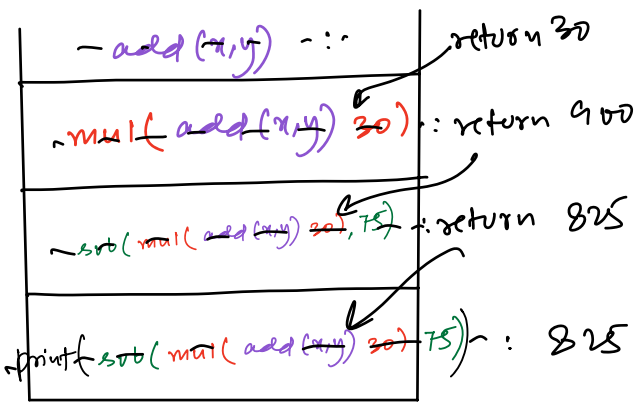
Observation 1 : Whenever a function call happens, we add the f^n call at top

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

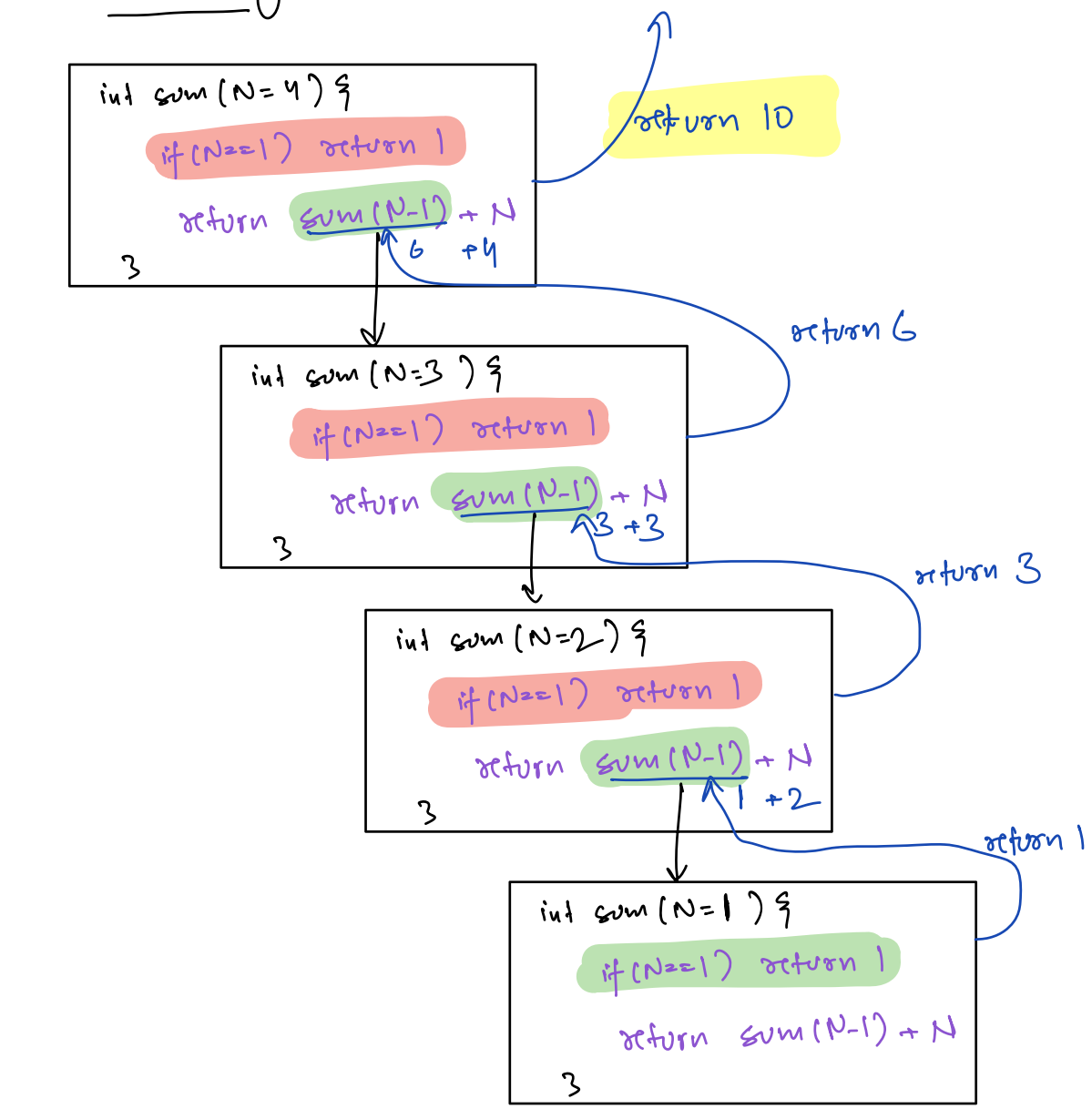
Last In First Out (LIFO)

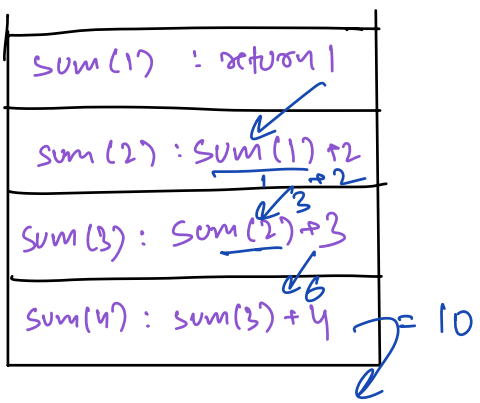
⇓

Stack



Sum(N) tracing





Question 3

$N >= 0$

# Input (N):	0	1	2	3	4	5	6	7	8
fib() :	0	1	1	2	3	5	8	13	21

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

```
int fib(N) {   arg.: calculate & return Nth fibonacci no.
    if(N==0) return 0 ]
    if(N==1) return 1 ] -> if(N<=1) return N
    return fib(N-1) + fib(N-2)
```

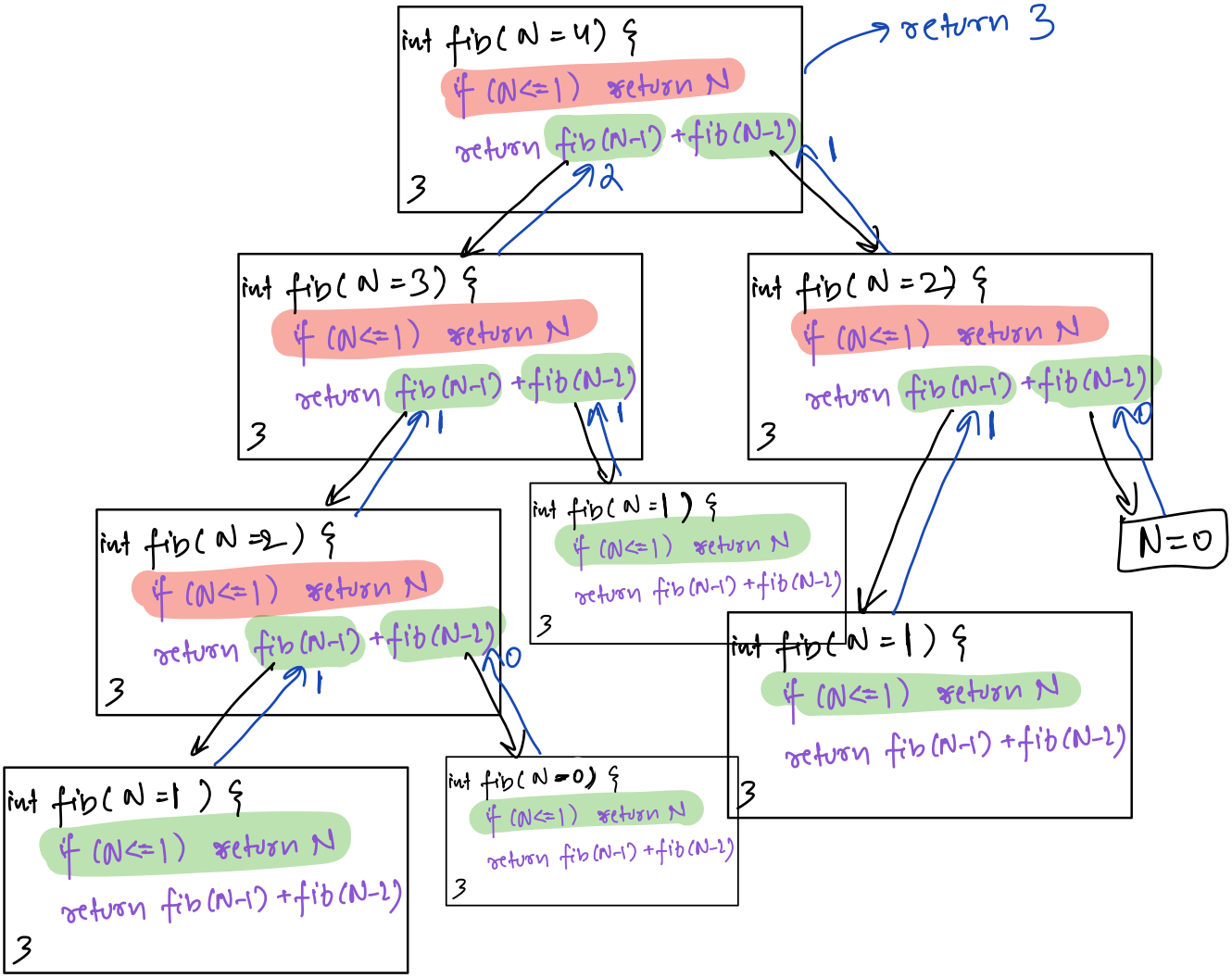
Note: How to properly figure out base condition(s)

→ 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 tracing with stack

Question 4

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

$N >= 1$

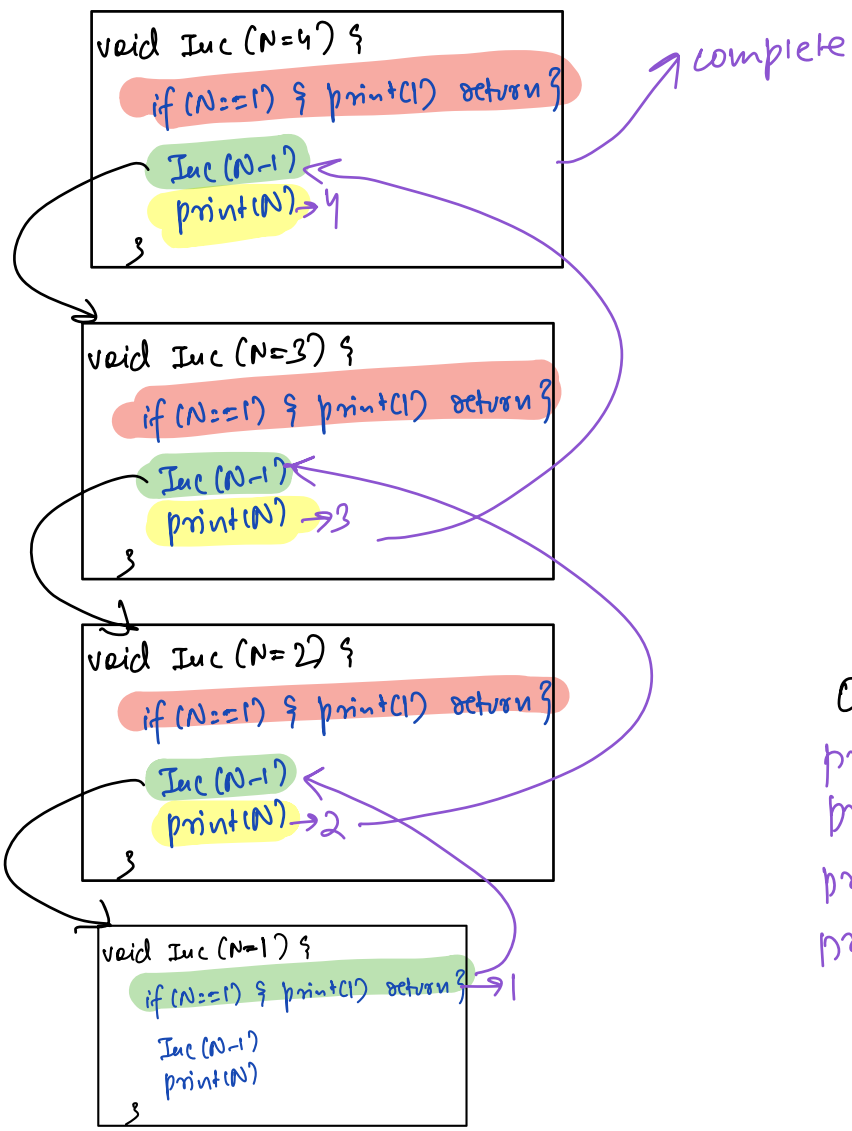
```
void Inc(N) {
    if(N==1) {
        print(1)
        return }
    Inc(N-1)
    print(N)
}
```

`Inc(N-1)` → 1 2 3 .... N-1

`print(N)` → N

Inc(3): 1 2 3

Inc(4): 1 2 3 4



Output:

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

Note

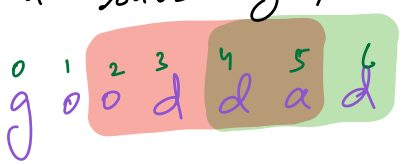
1. Even for void return type, we can return inside function.
2. Once a function completes it will go back to its parent function.

Homework: Print in decreasing order

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

Question 5

Given a substring, check if its palindrome or not?

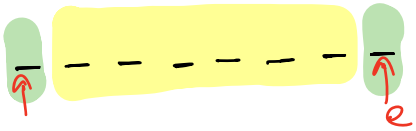
eg 

s=4, e=6 → return true

s=2, e=5 → return false

main problem → check whether [s,e] is a palindrom or not?

substring: ch



if (ch[s] != ch[e])

return false

if (ch[s] == ch[e])

check whether [s+1, e-1] is palindrome or not

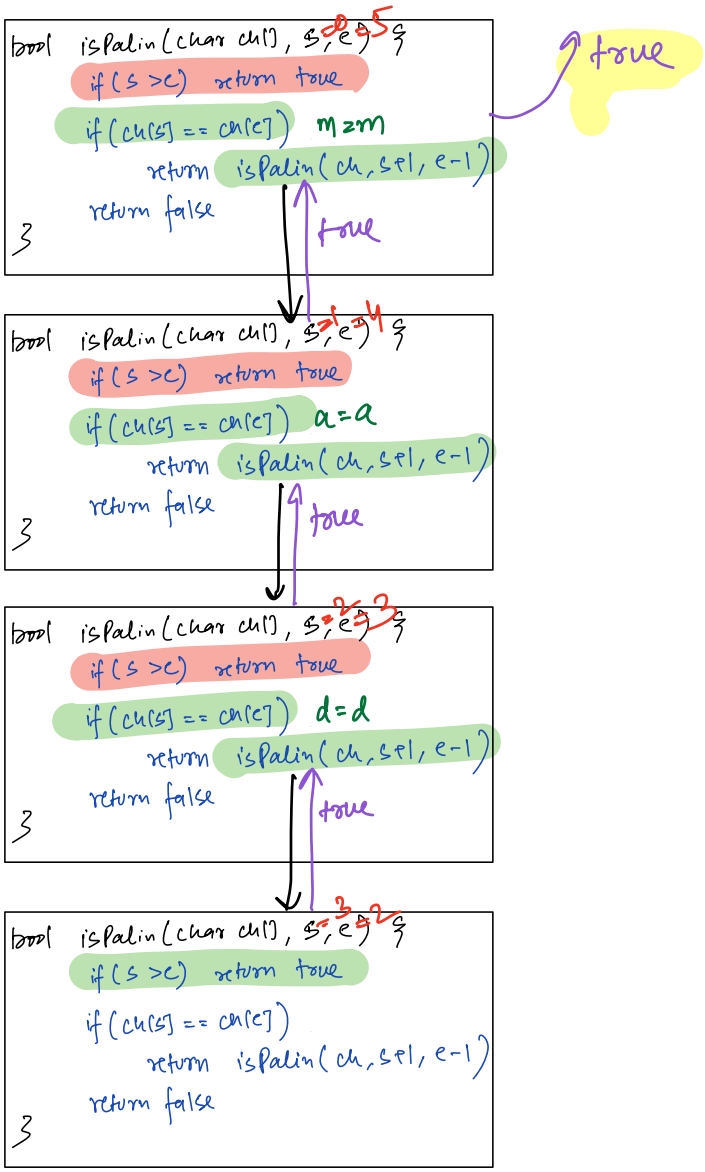
sub-problem

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

Input: m a d d a m

0 1 2 3 4 5

s=0, e=5



Input: a n m e t n a

0 1 2 3 4 5 6

s=0, e=6