Today's content

- Recursion
- How to write a recursive code / tracing
- TC & S.C of recursive codes → [next session]

Why recursion ?

- Merge Sort / Quick Sort / Many other algorithms
- Binary Tree / BST / BBST / Tries
- Dynamic Programming
- Backtracking
- Graph.

Recursion → function calling itself

↳ Solving a problem, using smaller instances of same problem

⇓

sub-problem

$sum(N) = 1 + 2 + 3 + 4 + \cdots + (N-1) + N$

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

$sum(5) = sum(4) + 5$ // sum(4) is a sub-problem.

How to write Recursive codes ?

Assumption : fix what your function should do.

Main Logic : Solving assumption using sub-problems.

Base Condition : Inputs, for which we want to stop recursion.

N>0

```
int Sum ( N) { //Ass→ Given N, calculate & return sum of first N natural nos.
    if (N==1) { return 1 }
    return { Sum(N-1) +N }
}
```

sum(3) = 6
sum(4) = 10

fact(3) = 3*2*1 = 6 , fact(4) = 4*3*2*1 = 24 , fact(5) = 120

N≥1

```
int fact ( N) { //Assm → Given N, calculate & return N!
    if (N==1) { return 1 }
    return { fact(N-1) * N }
}
```

fact(N) = 1*2*3* — —(N-1)*N
{ fact(N) = fact(N-1) * N }

Function Call Tracing.

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

int mul ( x, y) {
    return x*y
}

int sub ( x, y){
    return x-y
}
```

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

825:

900

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

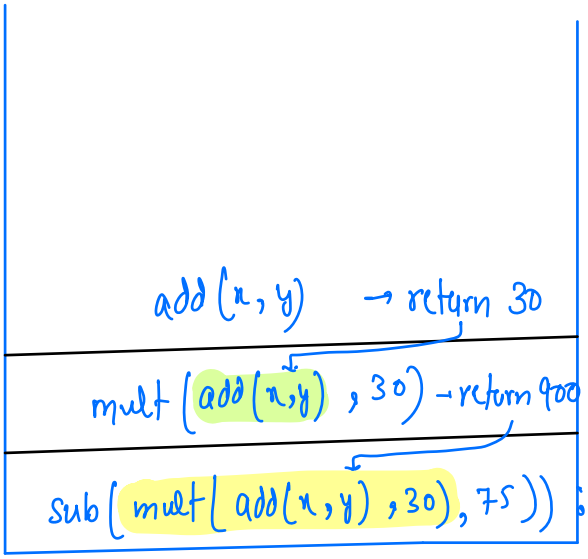
30

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

add (x, y) : return 30

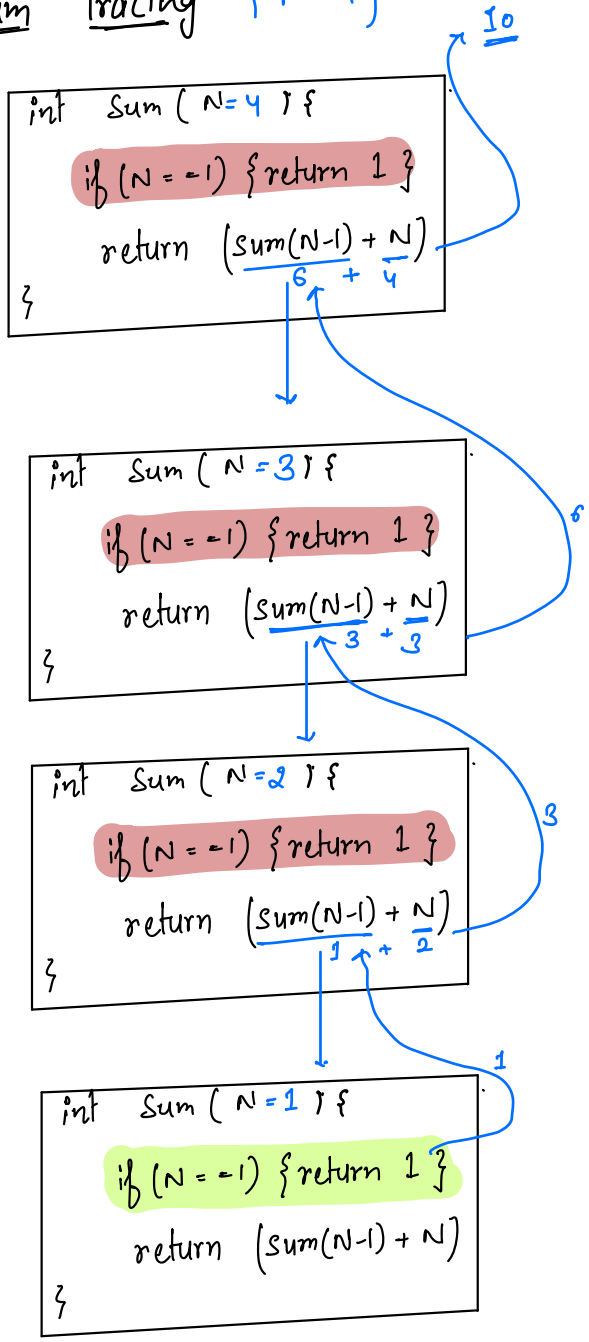
Data structure. ⇒ Stack.

// Last-In, first-out

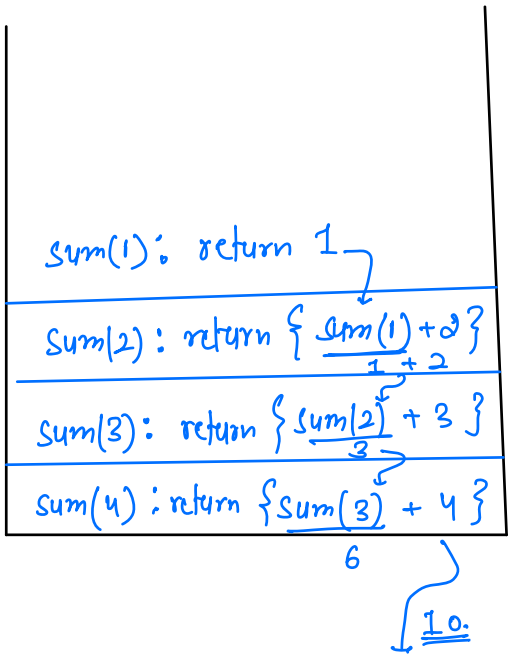


→ All functions calls are stored in stack.

//Sum Tracing { N=4 }



Stack trace.



Without Base Condition : Recursion won't stop, memory limit will exceed first.

- T.L.E
- Memory Limit Exceeded / Stack overflow

[Note - In your Recursive code, if you are getting "Memory Limit Exceeded" error. Try to verify your base condition.]

$N \geq 0$

# input :	0	1	2	3	4	5	6	7	8	9	10
fib() :	0	1	1	2	3	5	8	13	21	34	55

```
int fib (N) { // Ass^m => Calculate & return N^th fibonacci number
    if (N <= 1) { return N }
    return { fib(N-1) + fib(N-2) }
}
```

fib(5) : 5
fib(8) : 21

fib(N) = sum of previous 2 fibonacci numbers.

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

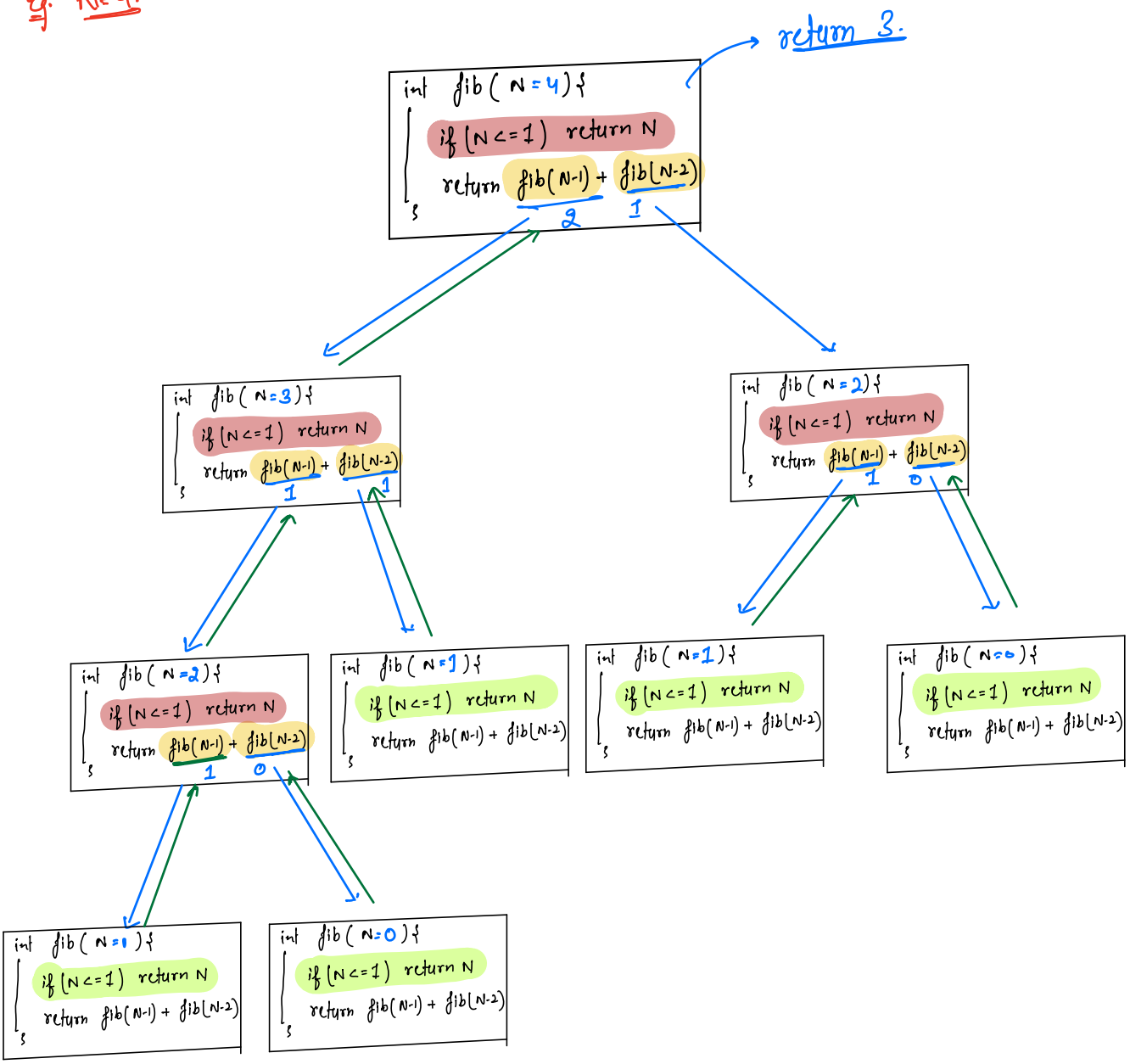
// How to find base conditions?
↳ Check for valid inputs where expression is invalid.

$fib(0) = fib(-1) + fib(-2)$ ✗

$fib(1) = fib(0) + fib(-1)$ ✗

$fib(2) = fib(1) + fib(0)$ ✓

Eg. N=4.



→ Trace using stack { # todo }

Break. { 9:03 → 9:10 }

// Given N, print all numbers from 1 → N in increasing order.

```
void pI (N) { // Ass^m - Given N, print all no's 1 → N in increasing order.
    if (N == 1) { print 1 }
    pI(N-1)
    print(N)
}
```

pI(3) = 1 2 3
pI(4) = 1 2 3 4

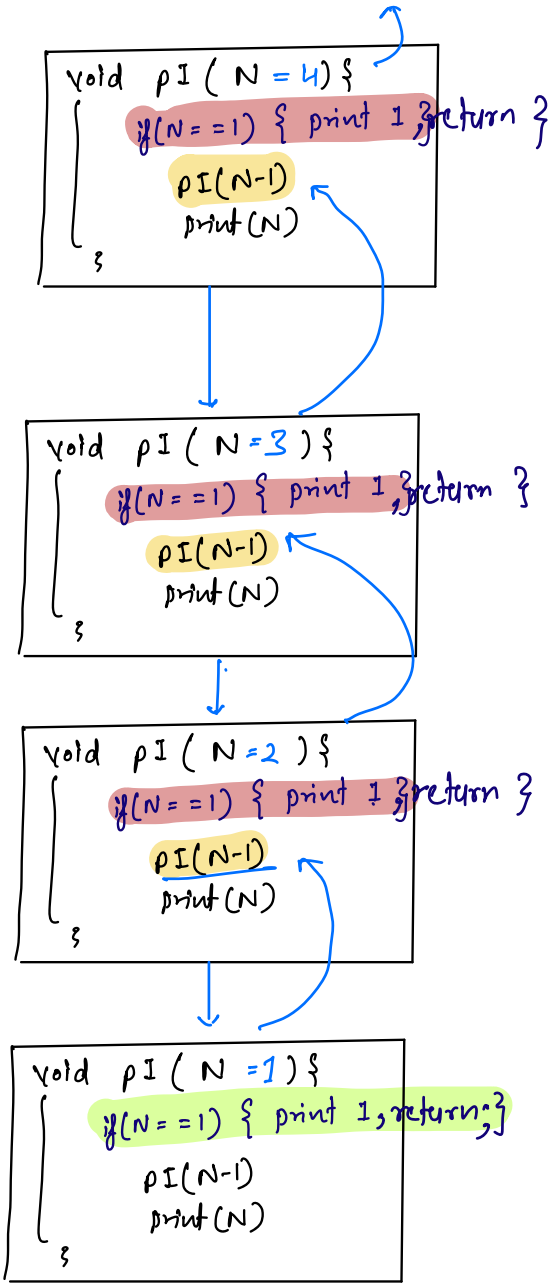
pI(N) → 1 2 3 ... (N-1) N

[pI(N-1)
  print(N)]

Stack trace { # todo }.

→ [All no's from 1 to N in decreasing order.] { # todo }

N=4.



o/p.

1
2
3
4

Note-1. → Even for void return type, we can use return inside function.

Note 2 → Once a function is completely executed, it will go back to its parent function.

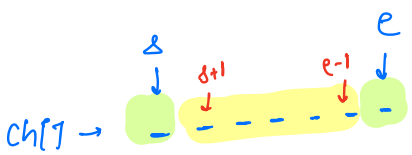
Q) Given a substring. Check if it is palindrome or not ?
[s <= e]

eg1 : g o o d d a d (0 1 2 3 4 5 6) , s = 4 , e = 6 ⇒ {return true}

eg2 : g o o d d a d (0 1 2 3 4 5 6) , s = 2 , e = 6 ⇒ {return false}

Assumption: Return, if substring from s to e is palindrome or not ?

```
boolean isPal ( char[] ch , int s , int e ){
    if ( s > e ) { return true }
    if ( ch[s] == ch[e] && isPal (ch, s+1, e-1) ){
        return true
    }
    return false ;
}
```

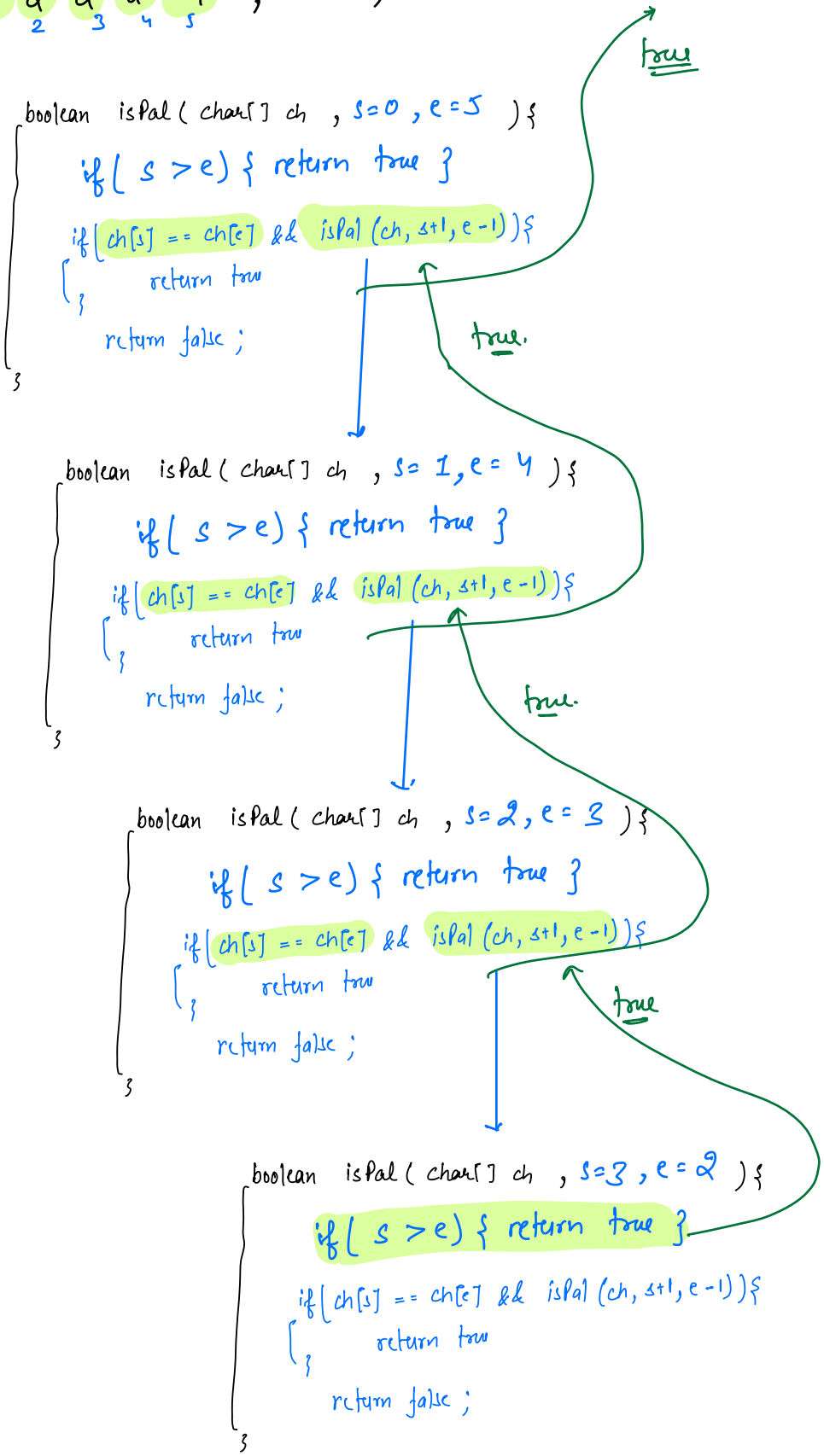


→ if ch[s] == ch[e]

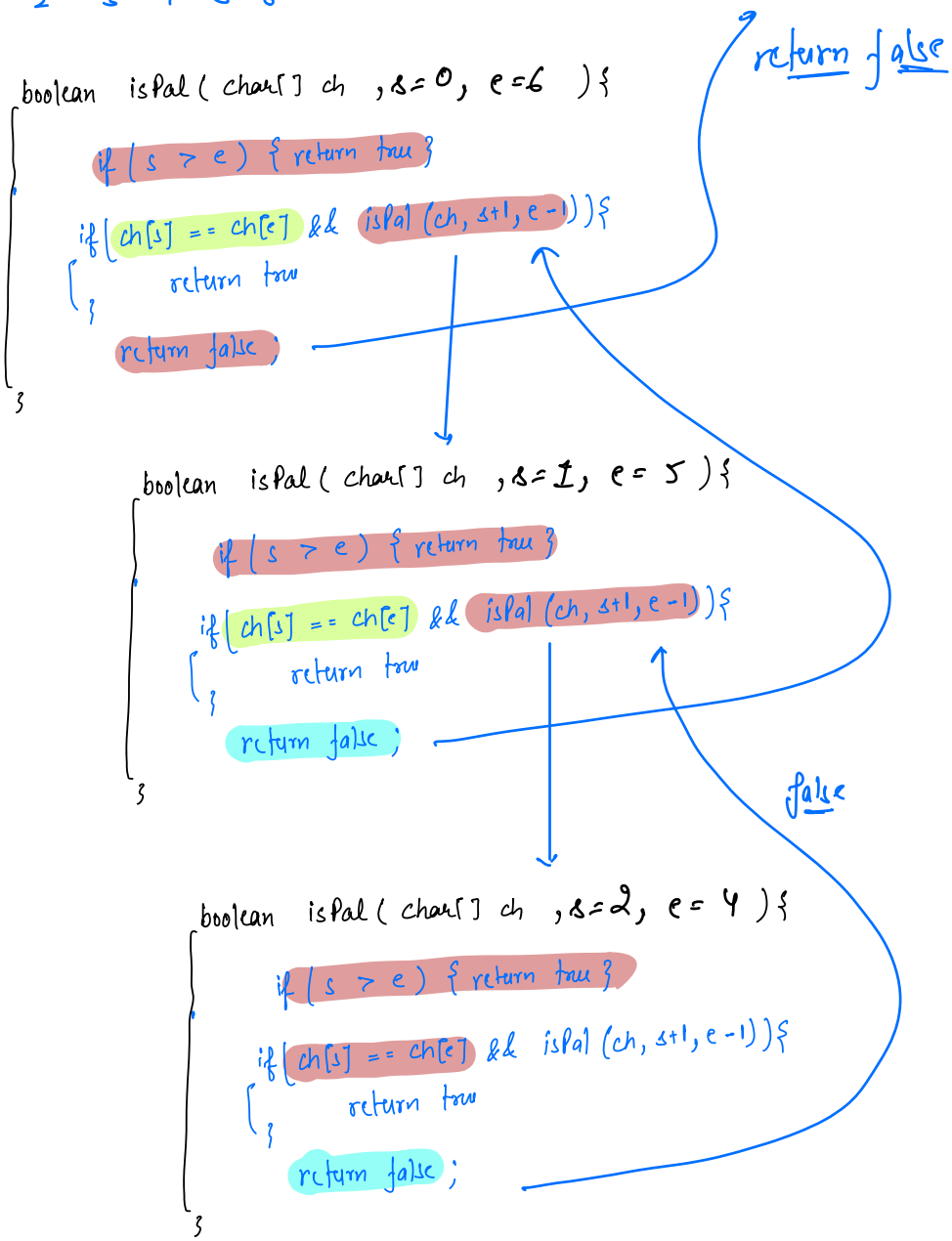
→ Substring from s+1 to e-1 will be considered as sub-problem.

→ isPal (ch, s+1, e-1)

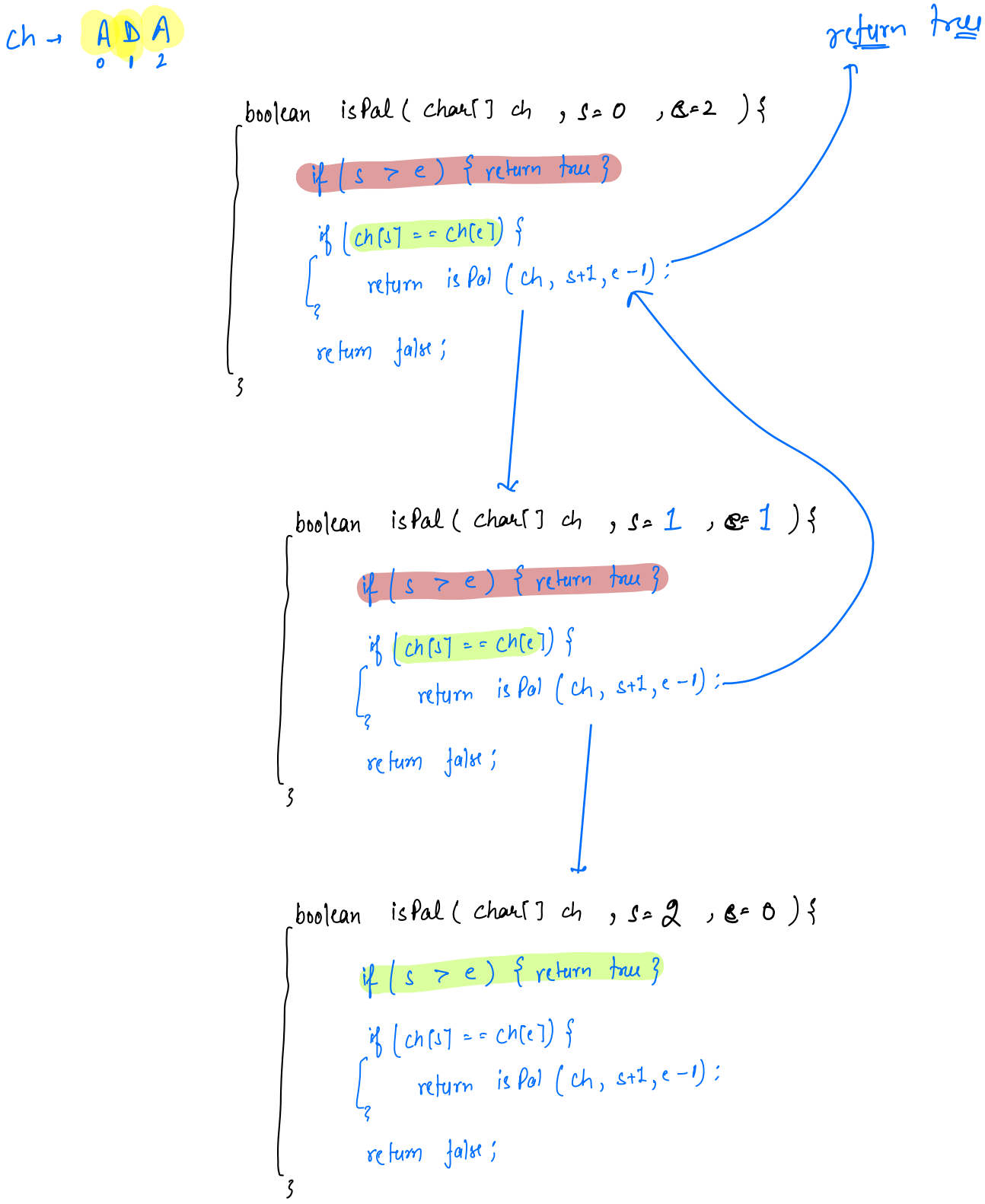
Input - m a d d a m , s=0 , e=5



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



[Note:→ Don't use pre & post increment & decrement operator in function calls.]


ch → A D A
0 1 2
return true
boolean isPal (char[] ch , s = 0 , e = 2) {
if (s > e) { return true }
if (ch[s] == ch[e]) {
return isPal (ch, s+1, e-1) ;
}
return false ;
}
boolean isPal (char[] ch , s = 1 , e = 1) {
if (s > e) { return true }
if (ch[s] == ch[e]) {
return isPal (ch, s+1, e-1) ;
}
return false ;
}
boolean isPal (char[] ch , s = 2 , e = 0) {
if (s > e) { return true }
if (ch[s] == ch[e]) {
return isPal (ch, s+1, e-1) ;
}
return false ;
}

{Contest → Online Assessment.}

① Time is fixed
② Can't ask for help.
③ Preparation of online assessments

[
① Relax & don't worry about result.
② Next day., try to solve problems again.
→ [2 hrs, 3 hrs.] → then ask for help.
③ Analyse.
]