Why Recursion ??

- Merge Sort / Quick Sort
- Trees → Binary Tree / BST / Tries / Segment Trees
- Dynamic Programming.
- Backtracking
- Graphs

Recursion : Function calling itself.

→ Solving a problem using, smaller instances of the same problem

⇓

Subproblem

Return the sum of 1st N natural no.

→ Sum(N) = 1+2+3+... (N-1) + N

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

Subproblem

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

Sum(3)

How to Write Recursive Codes

1) Assumption : (i) Decide what your function should do
(ii) Assume it returns a correct answer.

2) Main Logic : Solving assumption using subproblems

3) Base Condition : Inputs for which we want to stop the recursion
↓
At the start of function.

Ex 1 Sum of 1st N natural no.

```
int sum(N){ Ass: Given N, calculates & returns the correct sum of 1st N natural no's.
    if(N==1){
        return 1;
    }                    → Base Case
    return sum(N-1)+N; // Main Logic
}
```

$N \rightarrow N-1 \rightarrow N-2 \rightarrow N-3 \ldots\ldots\ldots 1$

$\downarrow$

1

Ex 2 Calculate $(N!)$ $\quad N \geq 1$

$fact(N) = 1 \times 2 \times 3 \times 4 \ldots\ldots (N-1) \times N$

$fact(N) = fact(N-1) \times N$

Code

```
int fact(N){   Ass: Given N, calculates & returns N! correctly
    if (N==1){
        return 1;
    }
    return fact(N-1)*N;
}
```

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

Function Call Tracing

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

900

30

900 - 75 = 825

825

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

825

900

sub (mul (add (x, y), 30), 75) ⇒ 825

900

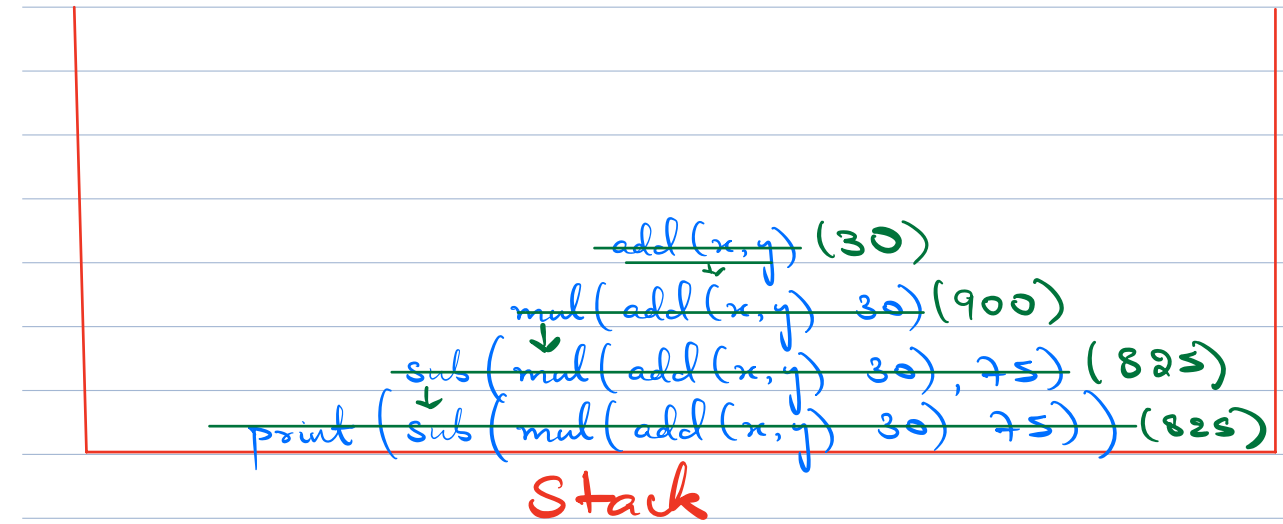
30

mul (add (x, y), 30) ⇒ 900

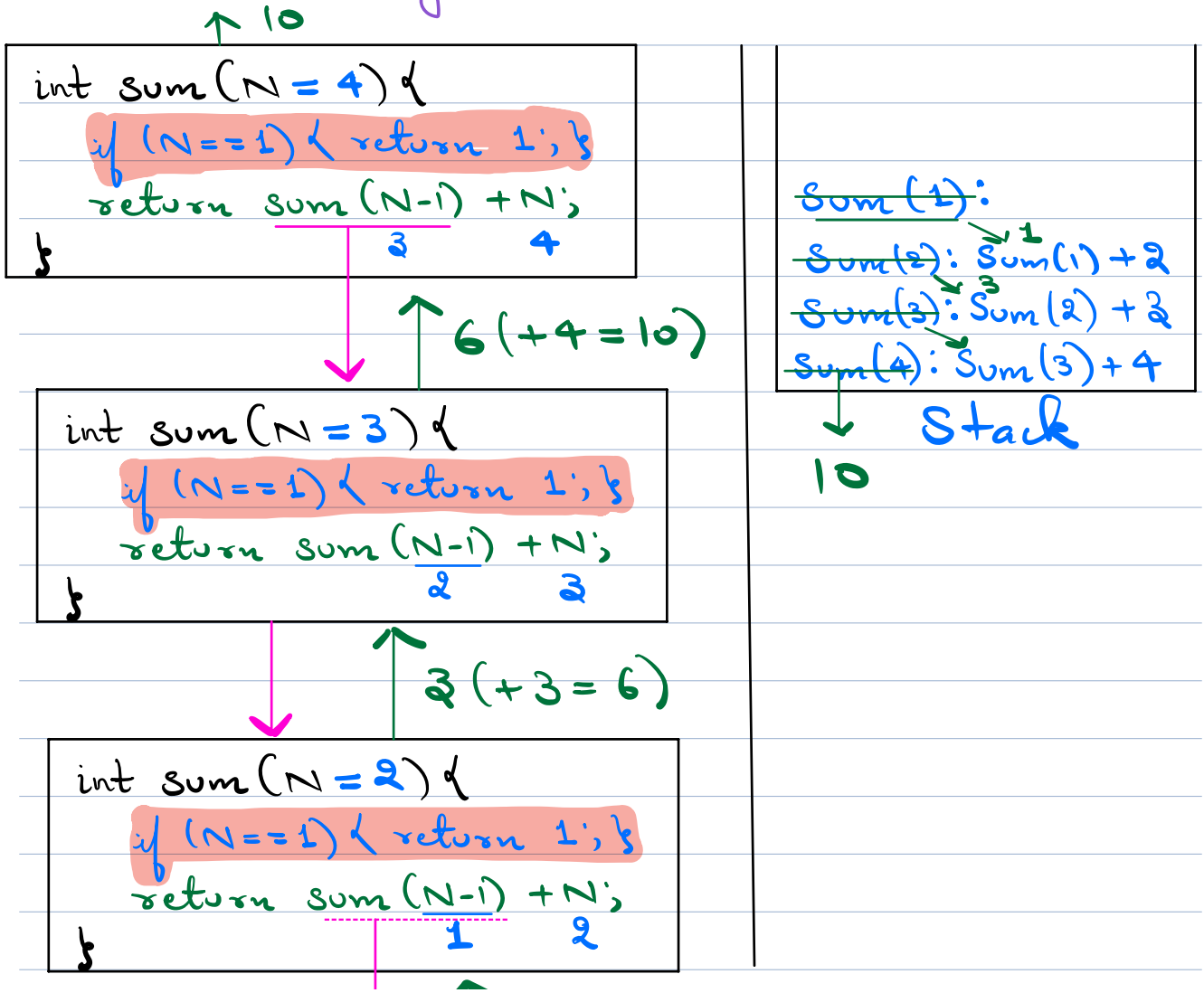
30

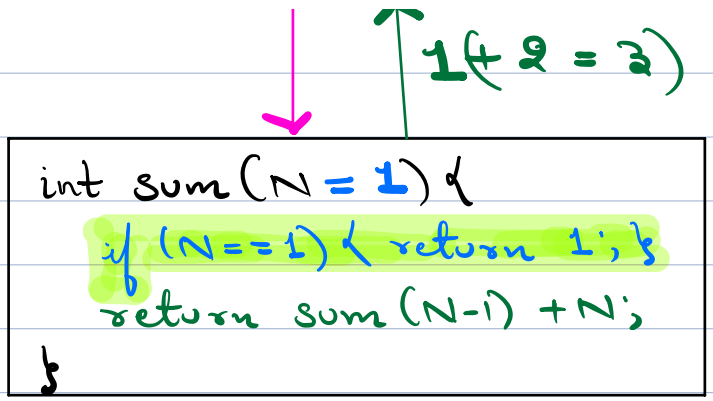
add (x, y) ⇒ 30

Data Structure ⇒ Stack (last in first Out)



Sum Tracing (N = 4)





Without Base Case ⇒ Recursion won't stop

4 → 3 → 2 → 1 → 0 → (-1) ..... $-\infty$

1) TLE

2) Memory Limit Exceeded / Stack Overflow.

How to define Base Case ??

Is subproblem valid ??

$$Sum(N) = Sum(N-1) + N;$$

Transition.

Invalid

$$Sum(1) = Sum(0) + 1 \quad (N = 1)$$

OR

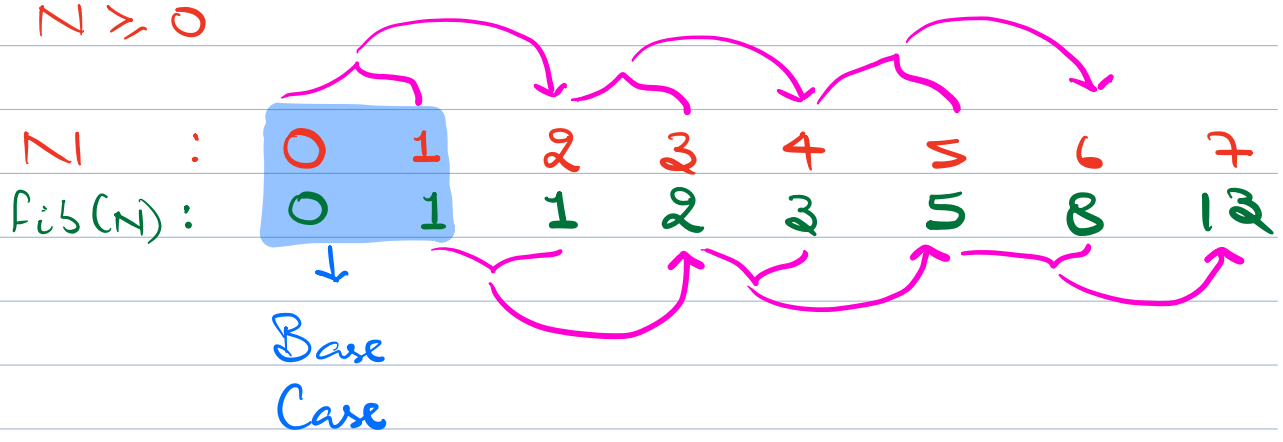
Invalid

Valid $$Sum(0) = Sum(-1) + 0 \quad (N = 0)$$

Fibonacci

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

$N \geq 0$

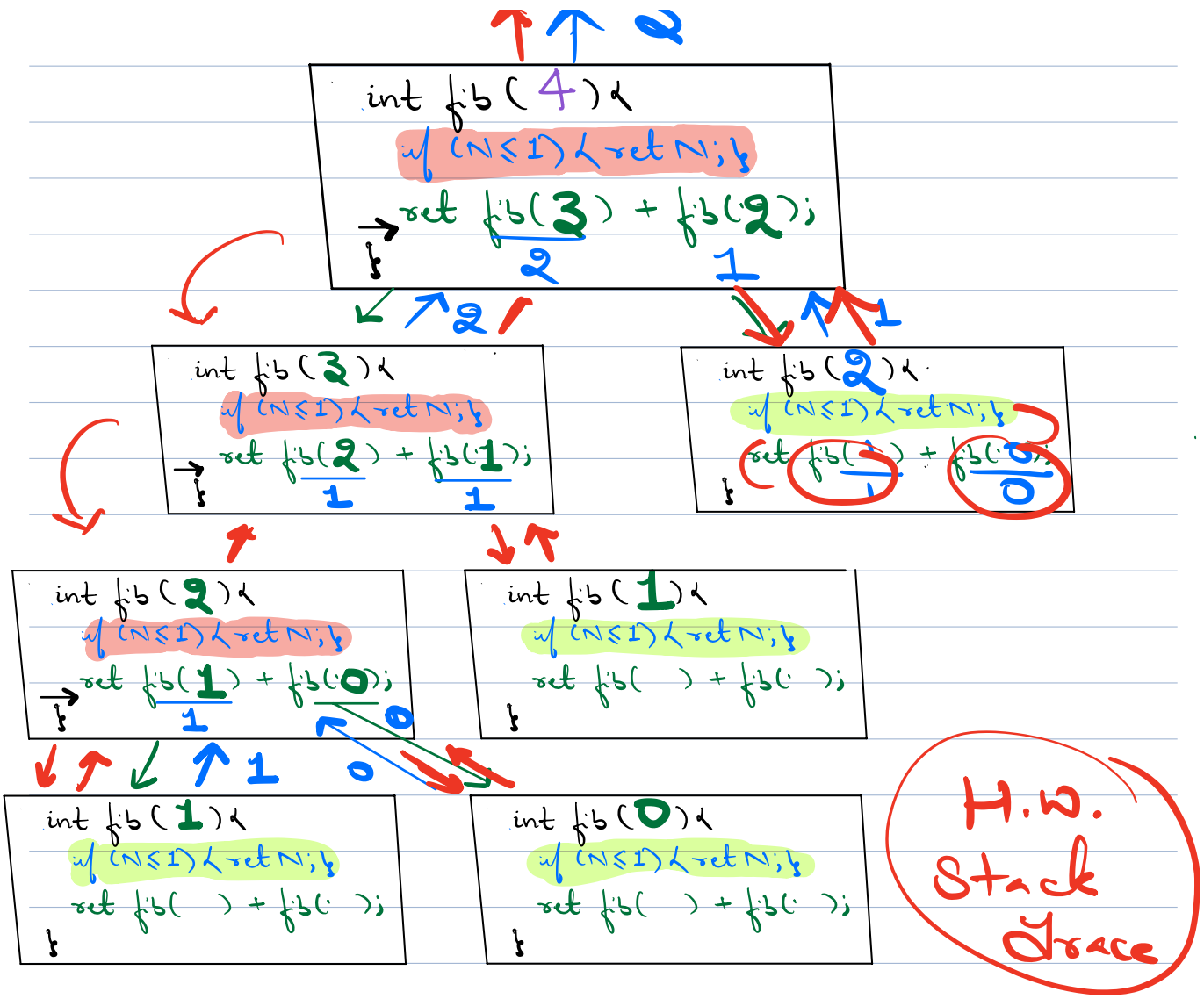


Given N, find the N^{th} fibonacci number.

Base Case
- $fib(0) = fib(-1) + fib(-2)$
- $fib(1) = fib(0) + fib(-1)$

Code

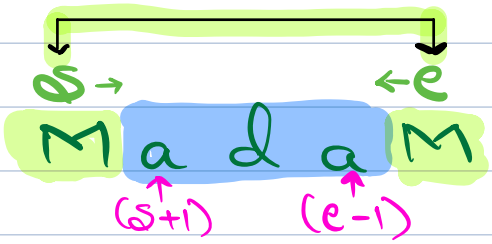
```
int fib(N) {
    if (N == 0 || N == 1) { return N; }
    return fib(N-1) + fib(N-2);
}
```

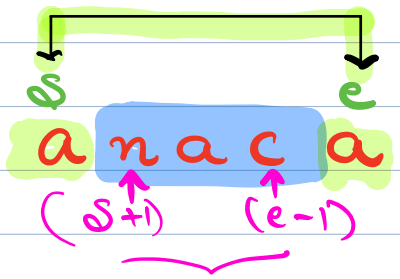


Q Given a substring of string (s, e) Check if it is a pallindrome recursively.

Ex1 S: 0 1 2 3 4 5 6 g o o d d a d (s = 4, e = 6) ✓

Ex2: S: 0 1 2 3 4 5 6 g o o d d a d (s = 2, e = 5) ✗



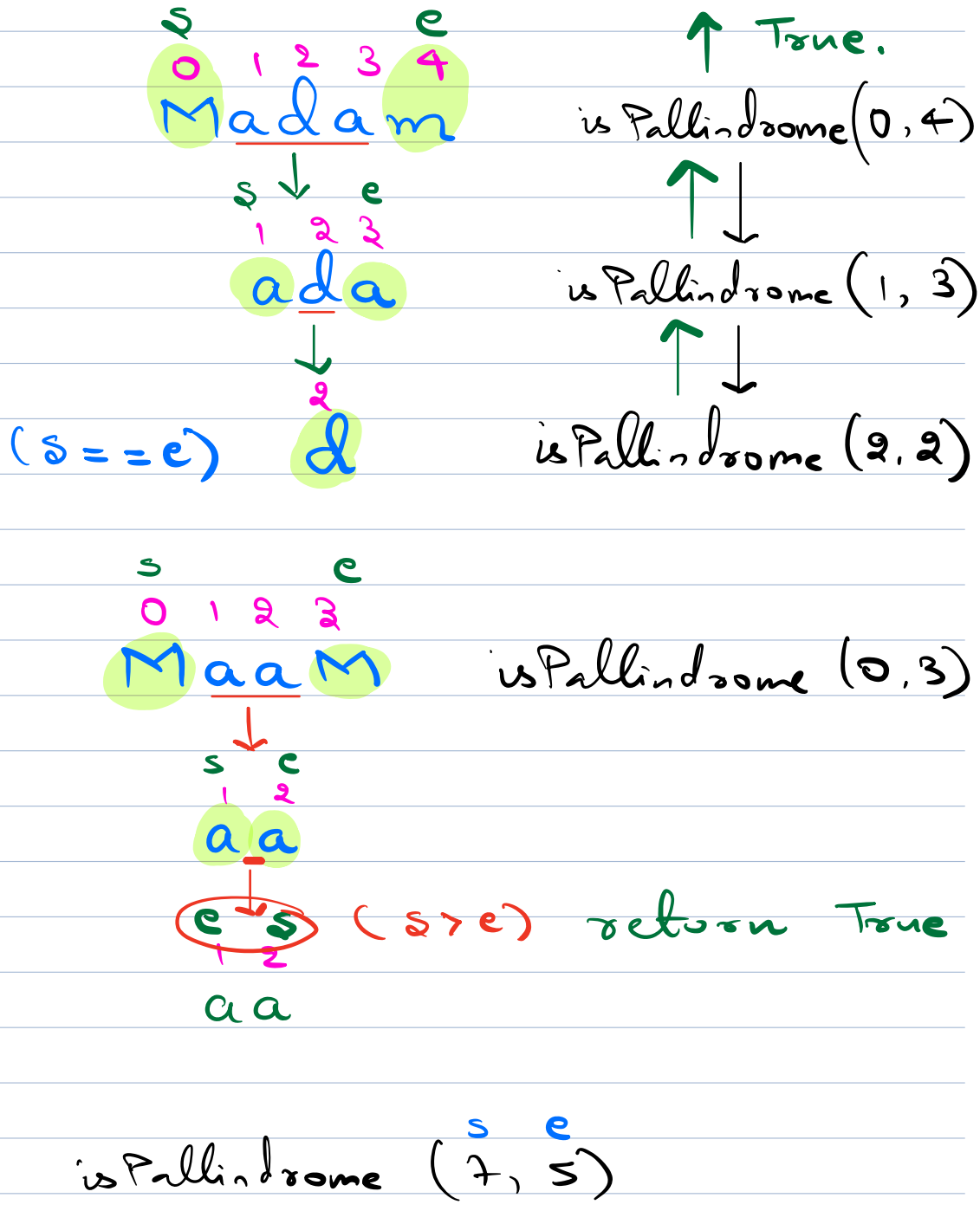


Not a pallindrome

Code

```
bool isPallindrome(str, s, e) {
    if (s >= e) { return True; }
    if ( str[s] == str[e] ) {
        return isPallindrome(str, s++, e--);
    }
    return false;
}
```

(0 over s, N-1 over e)



isPallindrome (7, 5) (s, e)

NOTE : Don't use postfix increment operators in Recursion
Use prefix