Recursion

① 1 TO N

1- get a number N
2- base case - run the loop till n = 0
3- keep calling the function by doing n-1
4- Print n

basically in every recursive call you're first executing 3rd step & pausing which keeps happening till n = 0 and returns back to calling function i.e f(1) where n = 1 moves to next step prints 1

ex: N=5 f(5) → f(4) → f(3) → f(2) → f(1) → f(0)
5 4 3 2 1

② N to 1

with the above same condition
if you place cout before calling it starts printing from n to 0.

__/__/____

③ Sum of N $\left(\frac{n(n+1)}{2}\right)$ if no rec

1. base case if N = 0 return 0
2. Sum = n + func(n-1)

ex: n = 5

5 = 5 + f(4) - 4 = 9
9 = 4 + f(3) - 3 =
3 + f(2) - 2 =
2 + f(1) - 1 =
1 + f(0) - 0 =

ex: n = 5

f(5) - 15
↓
5 + f(4) 10 = 5 + 10 = 15
4 + f(3) 6 = 4 + 6 = 10
3 + f(2) 3 = 3 + 3 = 6
2 + f(1) 1 = 2 + 1 = 3
1 + f(0) 0 = 1 + 0 = 1

here the base case is till 0 if 0 then return 0 and Sum = n + f(n-1) what it does is basically add the current functions n to the previous number like it goes upto f(0) & returns 0 for f(0)

f(1) then add 1 + 0
↓ & the Sum is f(1) = 1
n = 1
1 + f(1-1) f(0)

__/__/__

④ Factorial

just like the previous steps you're multiplying the current n with the result of the previous multiplied n & n-1
but base case till 1 if 0 return 0.

Ex: n = 3

f(3)

3 × f(2) 2 → 3 × 2 = 6

2 × f(1) 1 → 2 × 1 = 2

↓

return 1

⑤ Reversing the array

- Get the input array
- call the function by sending (arr, size, start, end)
- base case (start >= end)
- [illegible]
- swap start & end
- then call recursive function by moving start & end inden inwards
  i.e (start+1, end-1)
- and swap again till base case.

__/__/__

⑥ Palindrome string

1- get a string
2- get the size of the string

basically just like with array you keep 2 pointers for start & end when index comes to midpoint then its a palindrome cuz you compare start & end if its the same char you move ahead or you just return false

but comparing can be done in 2 ways.

- one is by calling start & end by assigning
  s[0] to start → Pali(s, start++1, end-1)
  s.size()-1 to end

- another is just passing i
  Pali(s, i) → if (s[i] != s.size() - i - 1)
  Pali(s, i+1)

⑦ finding n^{th} fibonacci

using recursion

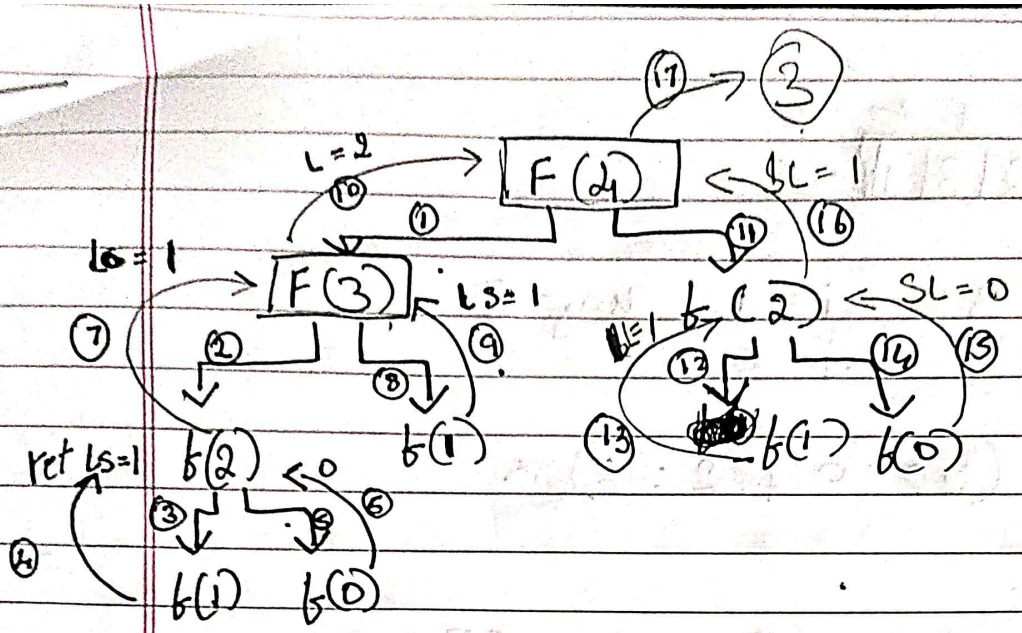
basically yore gettingh running the recursion loop till $n <= 1$ once it gets there return 0 or 1

& for calling the function its last & second last
Ex: $n = 5$ 5^{th} Fib number is 5

0, 1, 1, 2, 3, 5
↓ ↓
slast last

So $n - 1$ = last
$n - 2$ = slast
So goes down till 0 & 1 and starts adding up from there

and finally return last + slast



i = 4 j = 4

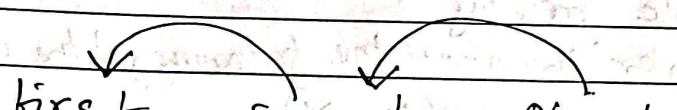
__/__/____

using for loop to find N^{th} fib

basically three variables for.

first = 0
second = 1
Next = 2

→ first = 1
second = 2
Next = 3



1st + 2nd = Next
1st = 2nd
2nd = Next

} this order cuz
if it is like 2nd = Next
then 1st = 2nd
then 1st will also get Next's
value which is in 2nd.