Q Alternative Sol to

1 to n. // initialize k=1

```
void fun(int n, int k)
{
    if(n==0)
        return
    S.o.pln(k)
    fun(n-1, k+1);
}
```

// k=1

```
int fact(int n, int k)
    if (n==0 || n==1)
        return k.
    return fact(n-1, k*n).
```

(3,1)
↳ 2, 1×3
↳ 1, 3×2
= 6
n==1
ret k
6
6

Tail Recursion

Solved faster by modern day

Compilers. //

factorial n where $n \geq 0$

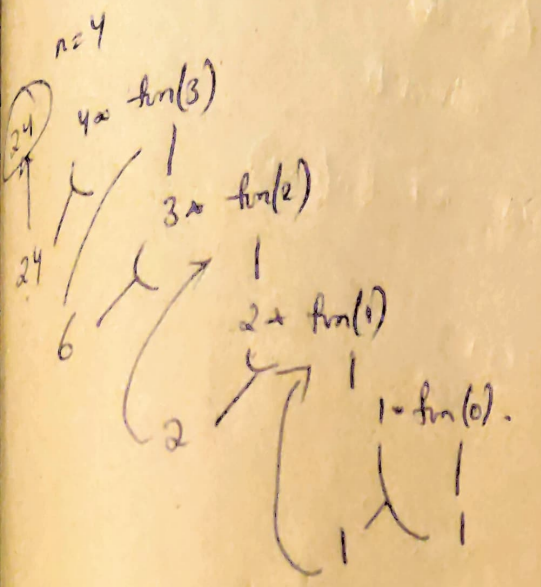
n = 4

o/p : 24.

int fun (int n)

if n == 0
return 1;

return n * fun(n-1)



---

fibonacci number $n \geq 0$.

n <= 1
return n;

int fib (int n)

if (n == 0) || n == 1
return 1;

return 0;

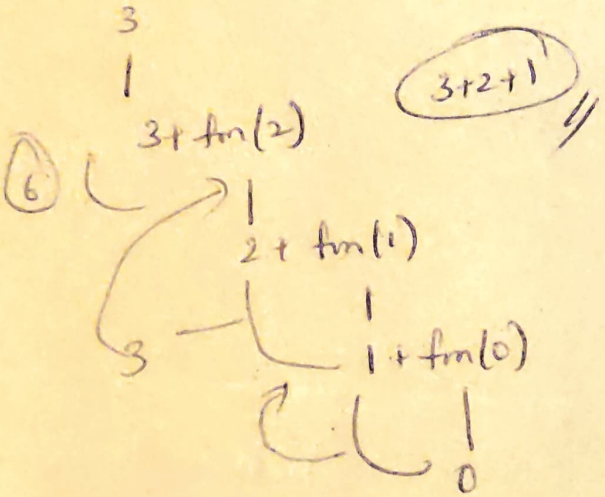
return fib(n-1) + fib(n-2);

Sum of 1st n natural no

n = 2

o/p : 3 ① ②

```
int fun(int n)
    if (n == 0)
        return 0;
    return n + fun(n-1);
```



Palindrome

// initially Start = 0;
End = n-1;

```
boolean { Palindrome (String str,
                      int Start,
                      int End)

    if Start >= End
        return true;
```

return (str.charAt(start)
== str.charAt(End)) &&

isPalindrome(str, Start+1, End-1);

Pal("acbca")

L pal("cbc")

Pal("b")

"abba"

"bb"

""

Recursion happens only if 1st & last char Equals ← // Short circuit //

Optimize
n <= 9
return n;

n = 253

int Sum(int n)
{
if (n == 0)
return 0;
return Sum(n/10) + (n % 10);

253

Sum(253/10) + 3

n = (25)

Sum(25/10) + 25 % 10

Sum(2/10) + 2 % 10

Sum(0)

0

3

d+1 — Recur Calls
O(d)

d+1 — fun call stack
O(d) A. Space