Res : Multiple Recursion Calls

```
f(){                    f(){
  :                       :
  :                     f() } multiple
} f() → single call     f() } calls
                        }
```

Fibonacci Number

0 1 1 2 3 5 8 13 21 34 . . .

0th, 1st, 2nd (1st + 2nd → next)

→N Q) Write a recursive function to find the nth fibonacci number. i.e $f(3) = 2$, $f(4) = 5$

Ans $f(4) = f(3) + f(2)$

simple code:

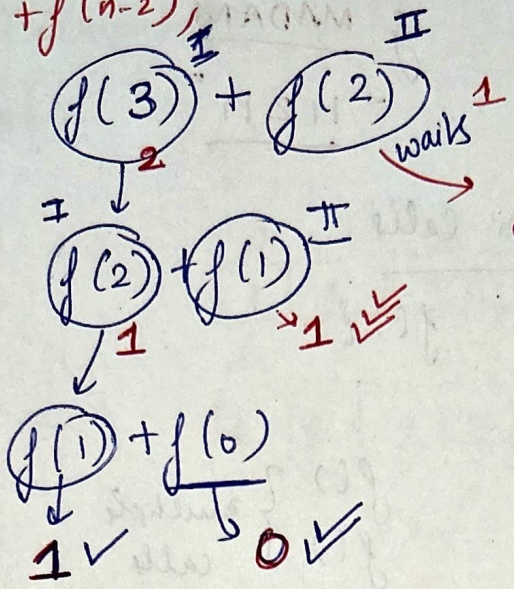
```
f(0) = 0, f(1) = 1
for (i = 2 → n)
    f(i) = f(i-1) + f(i-2)
```

$f(n) \rightarrow f(n-1) + f(n-2)$

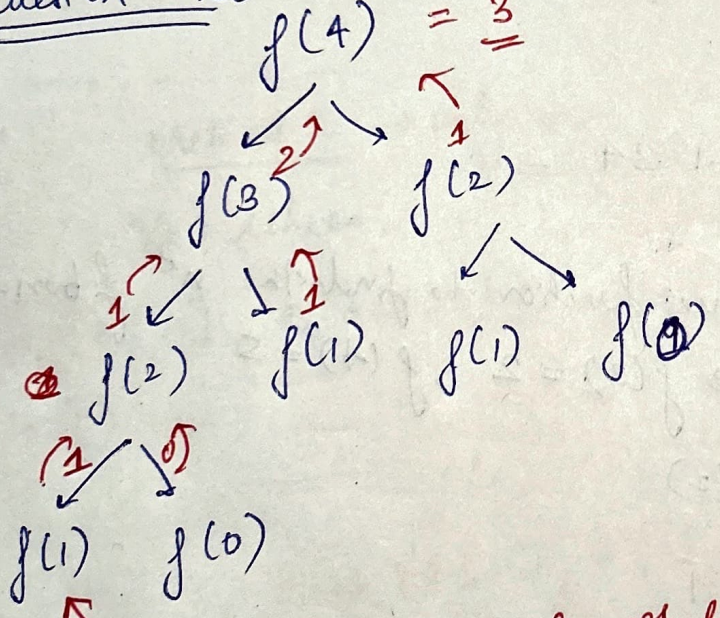
↳ recursive function code (next pg)

```
f(n) {
  if (n <= 1){
    return n;
  }
  return f(n-1) + f(n-2);
}
main() {
  n ← 4
  print (f(4));
}
```

Ans = 2 + 1 = 3



Recursion Tree

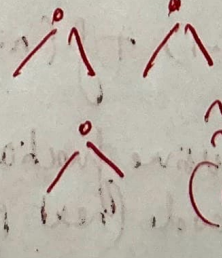


stacks from the lowest left branch & then its right & then goes up.

```
f() {
  f()
  f()
  f()
}
```

one ends, then other starts & so on..

Time complexity for f(4) was 9 as there 9 were 9 recursion calls



≈ (2^n) Time (exponential complexity in nature)