Recursion

```
void print_h() {
    printf("Hello World");
}
```

Eg:

```
void main() {
    print_h();
    print_h();
}
```

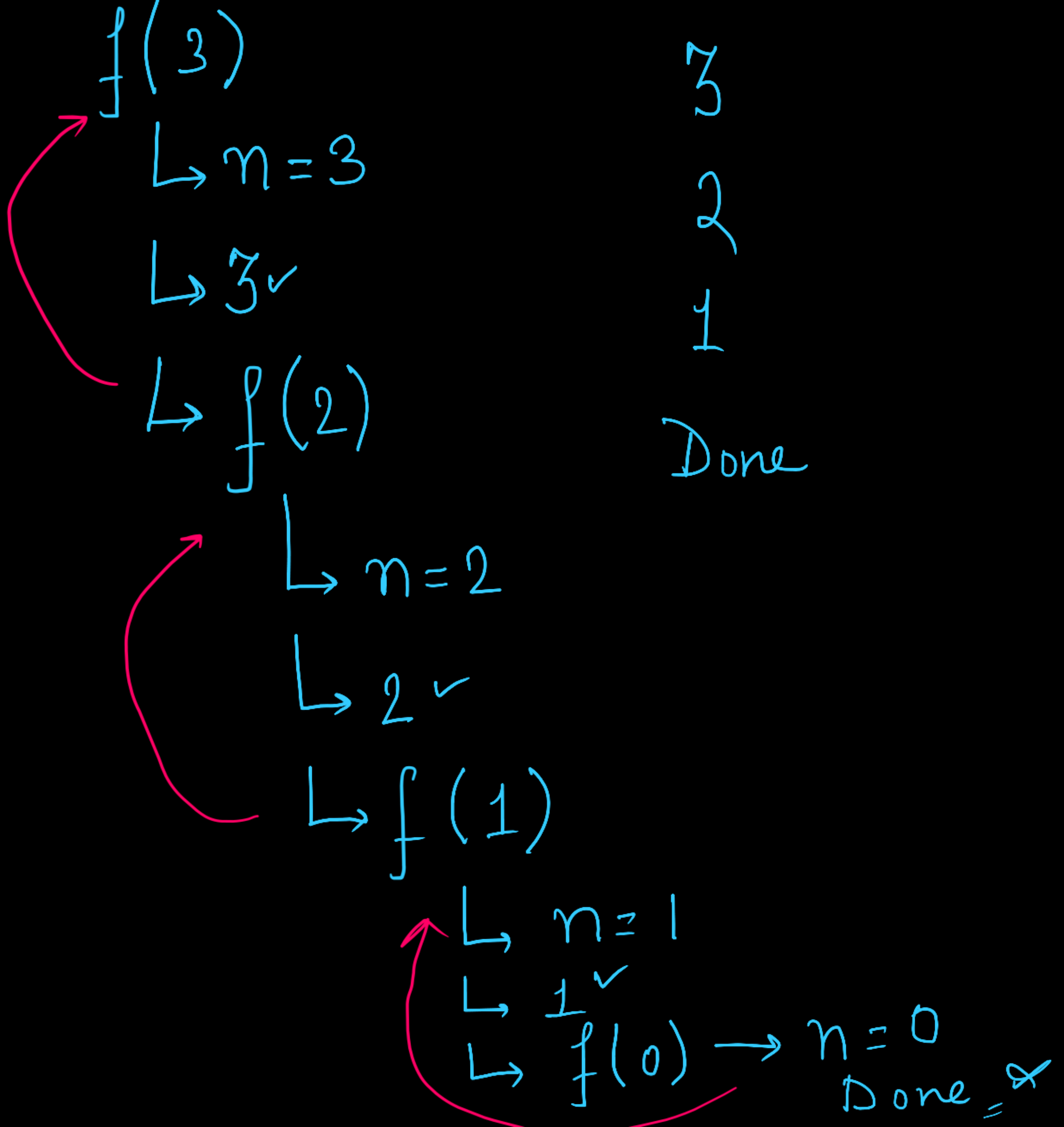
Recursion

When a function calls itself.

```
function fn( int x) {                {loop}
        _ _ _ _ _ _
        _ _ _ _ _ .
        _ _ _ _
        fn( x-1 )
}
```

Q Count down timer (reverse) 3 2 1 Done

```
→ Func contdown(n) {

  → if n == 0
        print("Done")        } Base case

    else
    → print(n)               } Recursive
    → contdown(n-1)            case
    →

→ Countdown(3)
```



Recursion tree

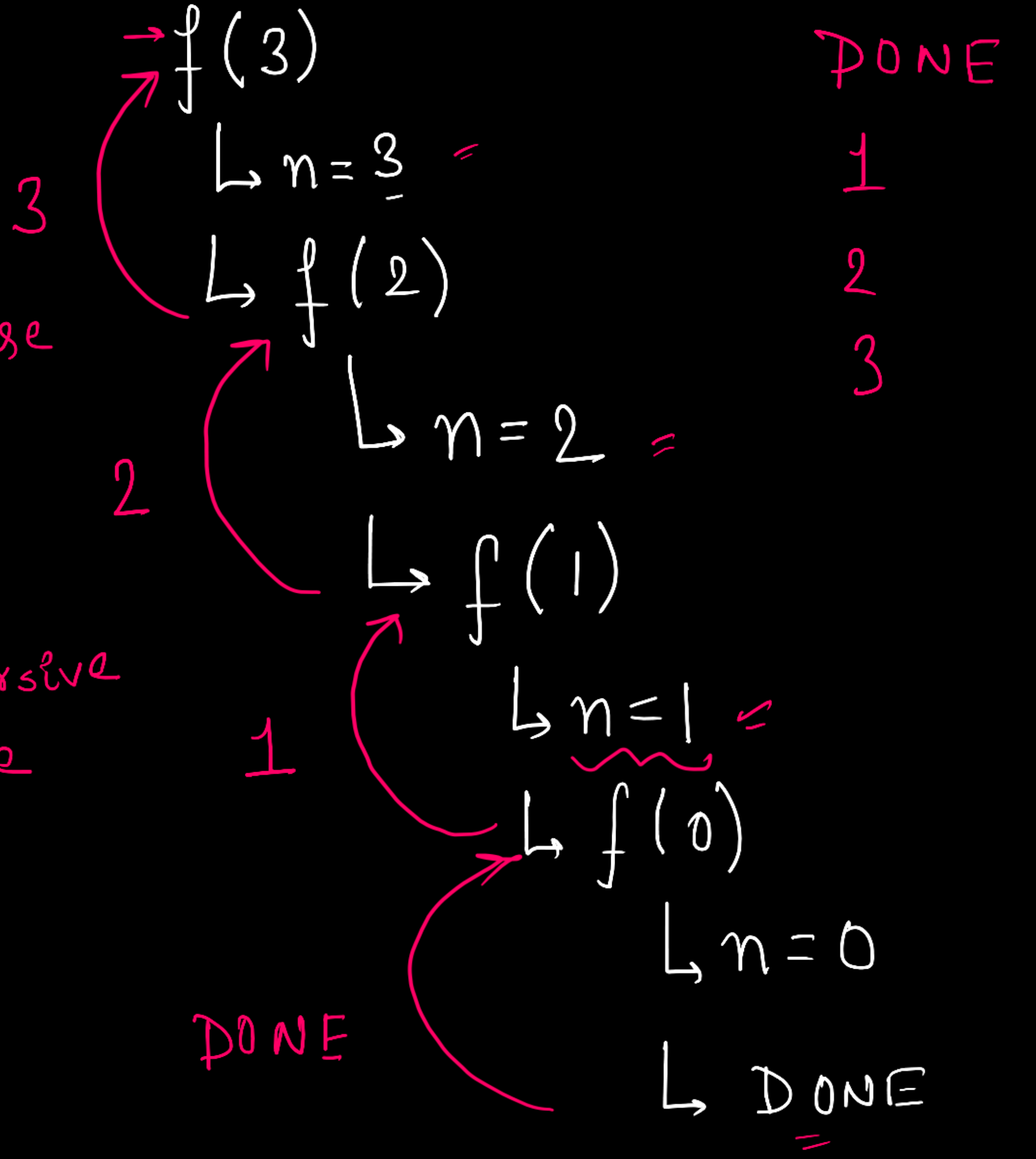
3

2

1

Done

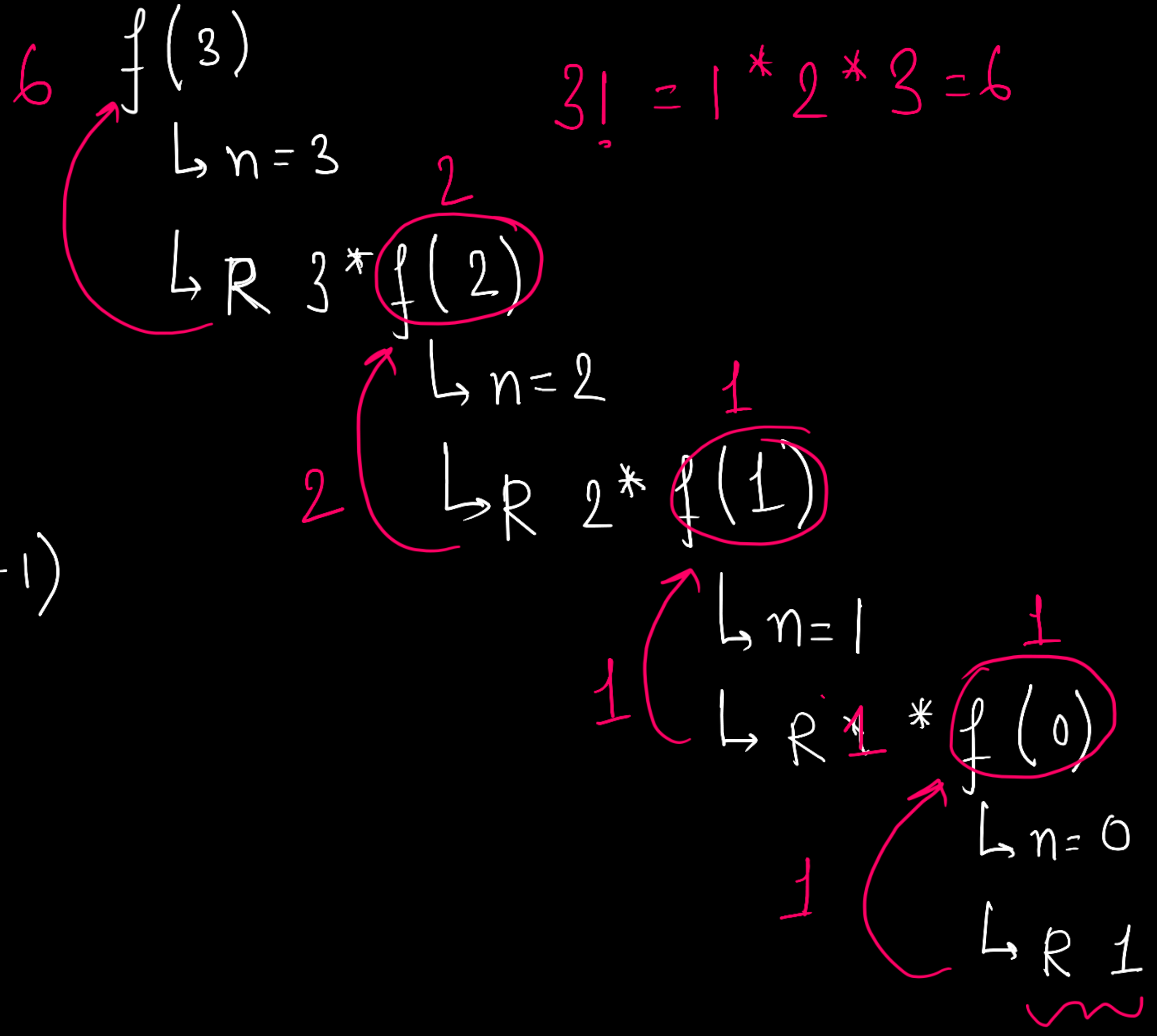
```
Func contdown(n) {

    if n == 0
        print("Done")          } Base case

    else
        ~~print(n)~~
        contdown(n-1)          } Recursive case
        print(n)


countdown(3)
```



```
Func    fact(n)
    if  n==0
        RETURN 1
    else
        RETURN n* fact(n-1)
```

= fact (3)

$3! = 1 * 2 * 3 = 6$



```
Fun sum_n(n):
    if n==0
        RETURN 0
    else
        RETURN n + sum_n(n-1)
```

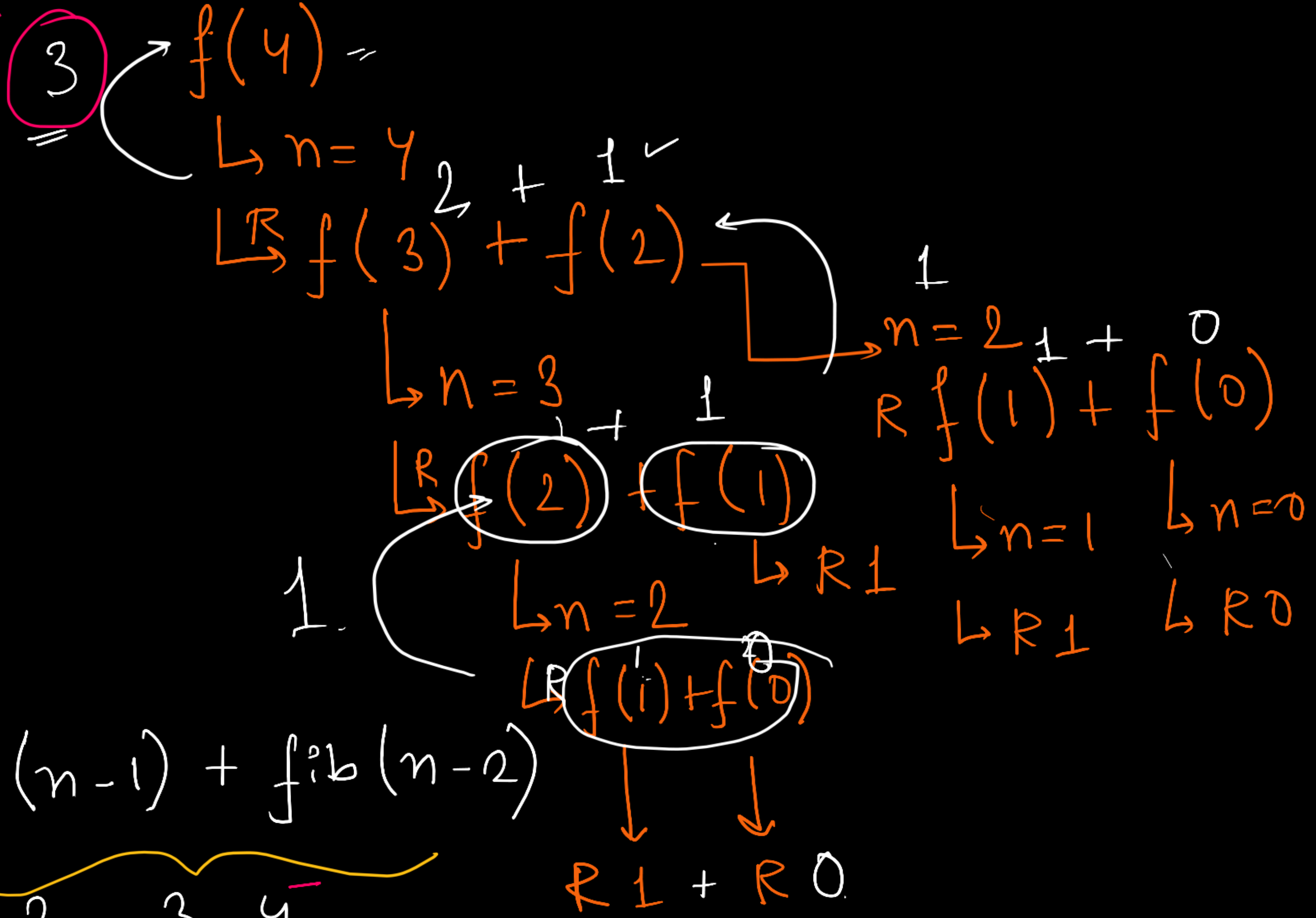
$$\underbrace{1 + 2 + 3 + 4} + \underline{5}$$

$$sum(n-1) + n$$

Q nth term of fib series.

```
Func fib(n)
    if n == 0
        RETURN 0
    else if n == 1
        RETURN 1
    else
        RETURN fib(n-1) + fib(n-2)
```

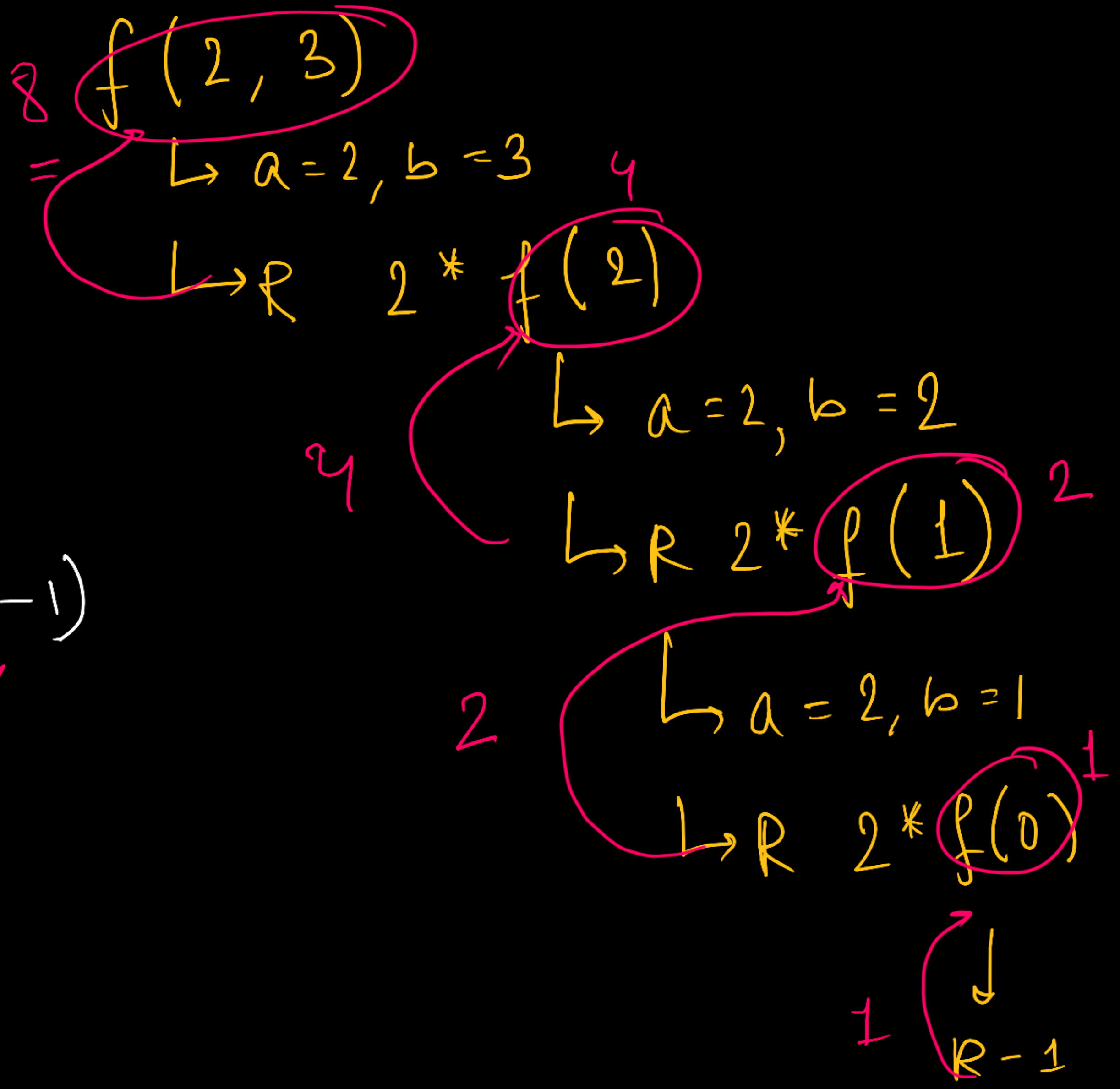
0	1	2	3	4
0	1	1	2	3



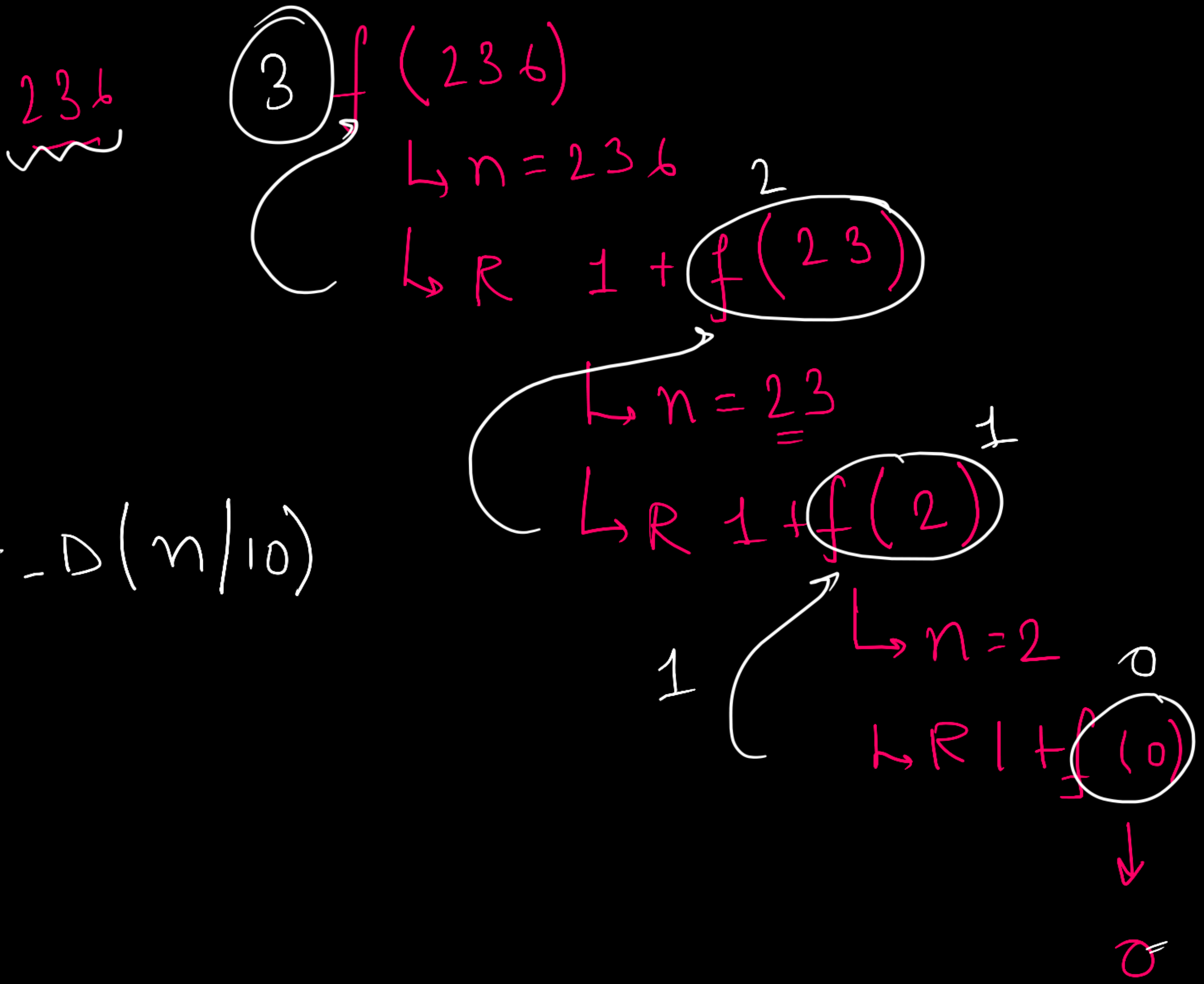
```
Func power(a, b)
    if b == 0
        RETURN 1
    else
        RETURN a * power(a, b-1)
```

$\boxed{a^b}$

2^3



```
Func    count_D(n):
    if    n==0
        RETURN 0
    else
        RETURN 1+ count_D(n/10)
```



Diff. b/w loop & recursion

loop	recursion
(i) Iterative for, while	(i) f^n call
(ii) Lesser memory usage (single stack frame)	(ii) More memory usage (each call uses new stack)
(iii) Faster & Efficient	(iii) slower
(iv) - - - .	For T BT