Recursion

- Harsh Gupta

Goal

To understand:

- Recursion
- Recursive **Call Stack**
- Recursion Tree

Recap on Functions

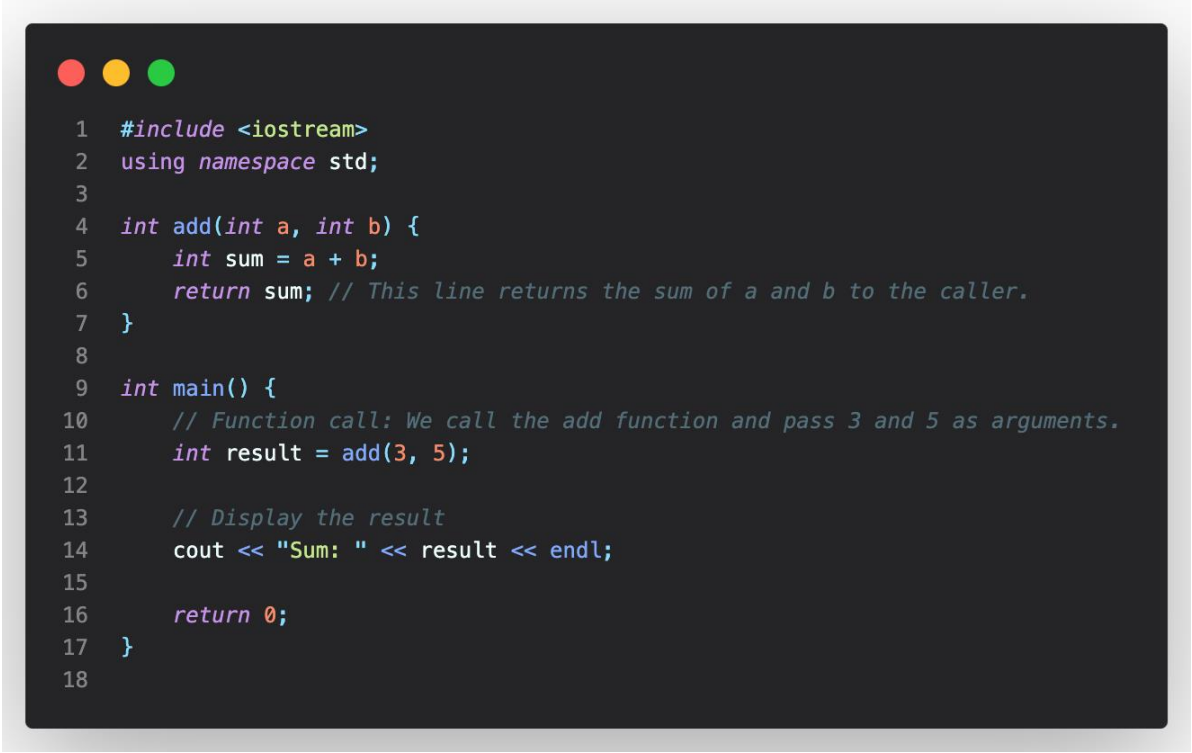
Functions are **reusable blocks of code** that can be run whenever called.

They can take in parameters (input) and return a value (output).

Syntax:

```
return_type name(d1 param1, d2 param2, …) {
    // result must be same as return_type
    return result;
}
```

Recap on Functions (Contd…)



```cpp
#include <iostream>
using namespace std;

int add(int a, int b) {
    int sum = a + b;
    return sum; // This line returns the sum of a and b to the caller.
}

int main() {
    // Function call: We call the add function and pass 3 and 5 as arguments.
    int result = add(3, 5);

    // Display the result
    cout << "Sum: " << result << endl;

    return 0;
}
```

What is Recursion?

- Recursion happens when a **function calls itself** on a different set of input parameters.
- Used when the solution for current problem involves first solving a **smaller sub-problem**.

Example - Sum of First N Natural numbers

If we want to find sum of first **N natural numbers** , we can easily get that if we know the sum of first **(N-1) natural numbers**.

So finding sum of first (N-1) natural numbers become a **smaller subtask for us**.

Similar thing happens with recursion. By **using the answer from smaller subproblem we can solve bigger problems**.

Sum of first 40 Natural Numbers.

$N \rightarrow (N)(N+1)/2$

1+2 → 3

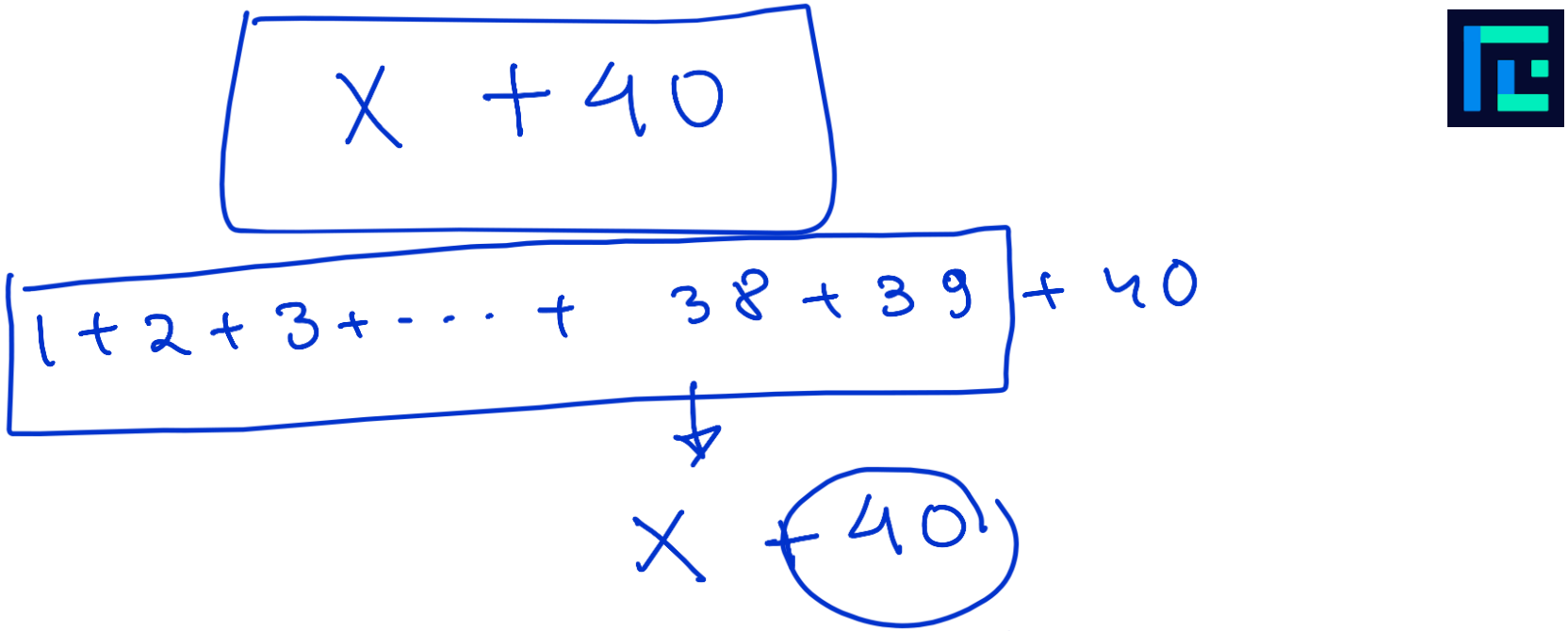
1 + 2 + 3 + 4 + 5 + ... + 40

Impossible

friend Recursion

40 → 39 ?

39 Natural Numbers. = X

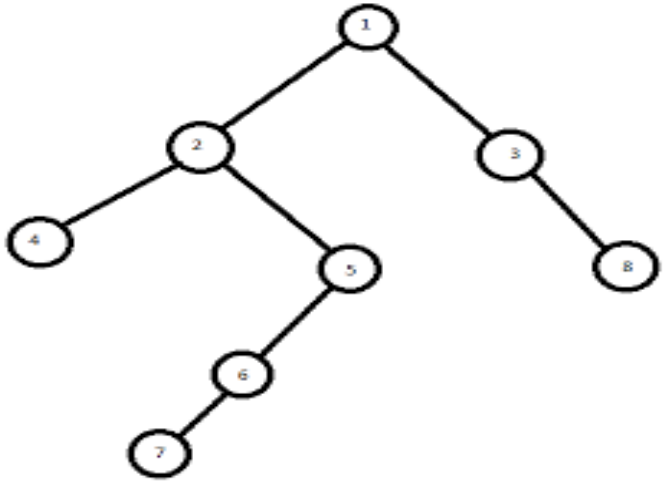


It helps us to solve bigger problem by solving smaller subproblems.

Recursion Tree

- A recursive tree is similar to a **“mind map”** of the function call.
- Recursive trees are useful to help us understand how the function acts.

Example:



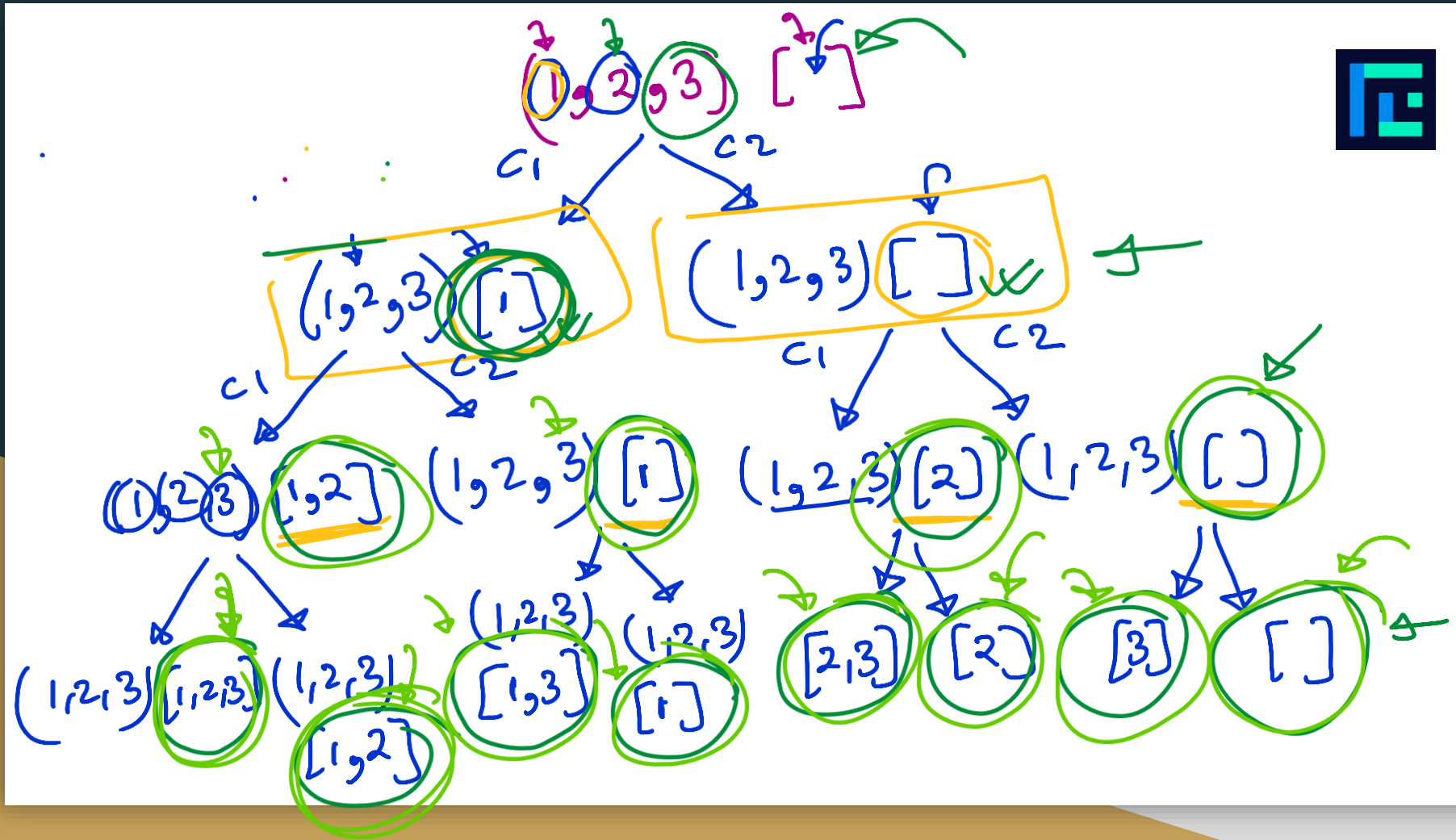
Example - Recursion Tree

Question: Generate all the **subsets** of an array [1,2,3] →

[ ], [1] , [2] , [3] , [1,2] , [2,3] , [1,3] , [1,2,3]

Example - Recursion Tree

Solution: For all the elements in the array , I have two choices.
Either I include them in the subset (choice C1) or I don’t include them (choice C2).


(1,2,3) [ ]
c1
c2
(1,2,3) [1]
(1,2,3) [ ]
(1,2,3) [1,2]
(1,2,3) [1]
(1,2,3) [2]
(1,2,3) [ ]
(1,2,3) [1,2,3]
(1,2,3) [1,2]
(1,2,3) [1,3]
(1,2,3) [1]
[2,3]
[2]
[3]
[ ]

Basic Structure of Recursive Function

- Parameters to start the function
- Appropriate **base case(s)** to end the recursion (Why ??)
- Recursively solve the **sub-problems**
- Process the result and return the value

Example - Nth Fibonacci Number

The Fibonacci sequence is the series of numbers where each number is the **sum of the two preceding numbers**.

For example,

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, …

Mathematically we can describe this as:

$F_n = F_{n-1} + F_{n-2}$, for all $n >= 3$

5th Fibonacci

$$F_n = F_{n-1} + F_{n-2}$$

$n \geq 3$

0 1

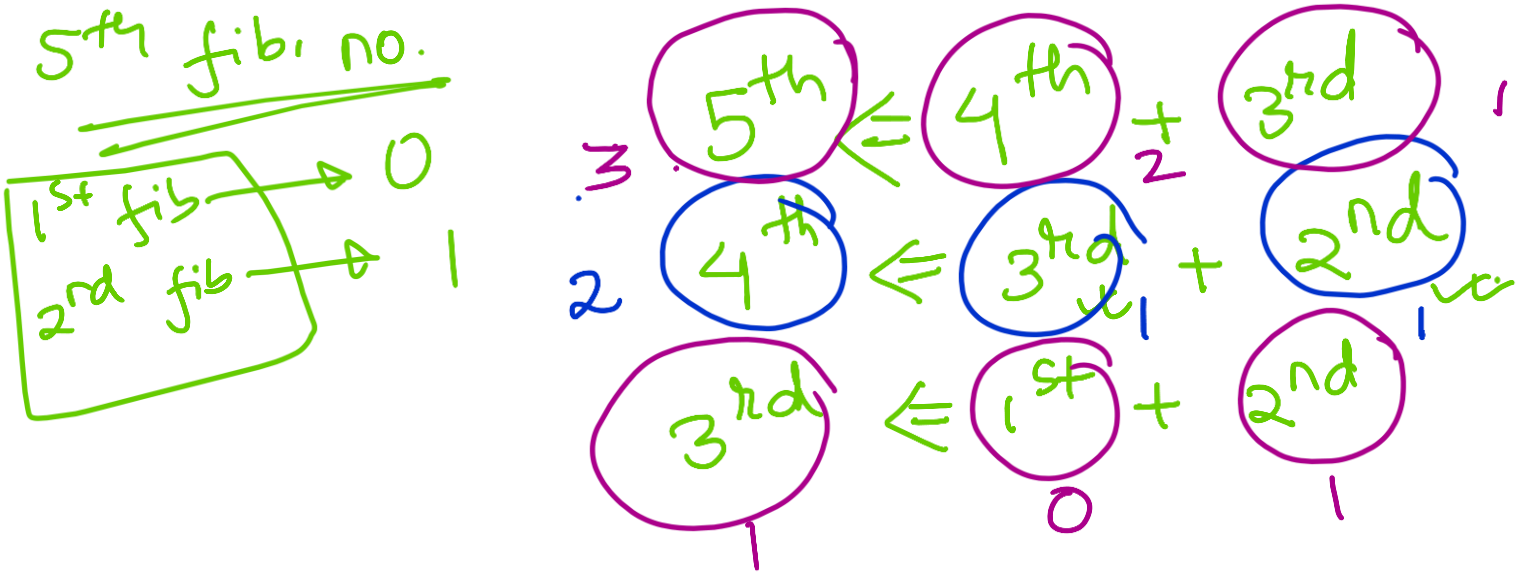
Example - Nth Fibonacci Number

Code →

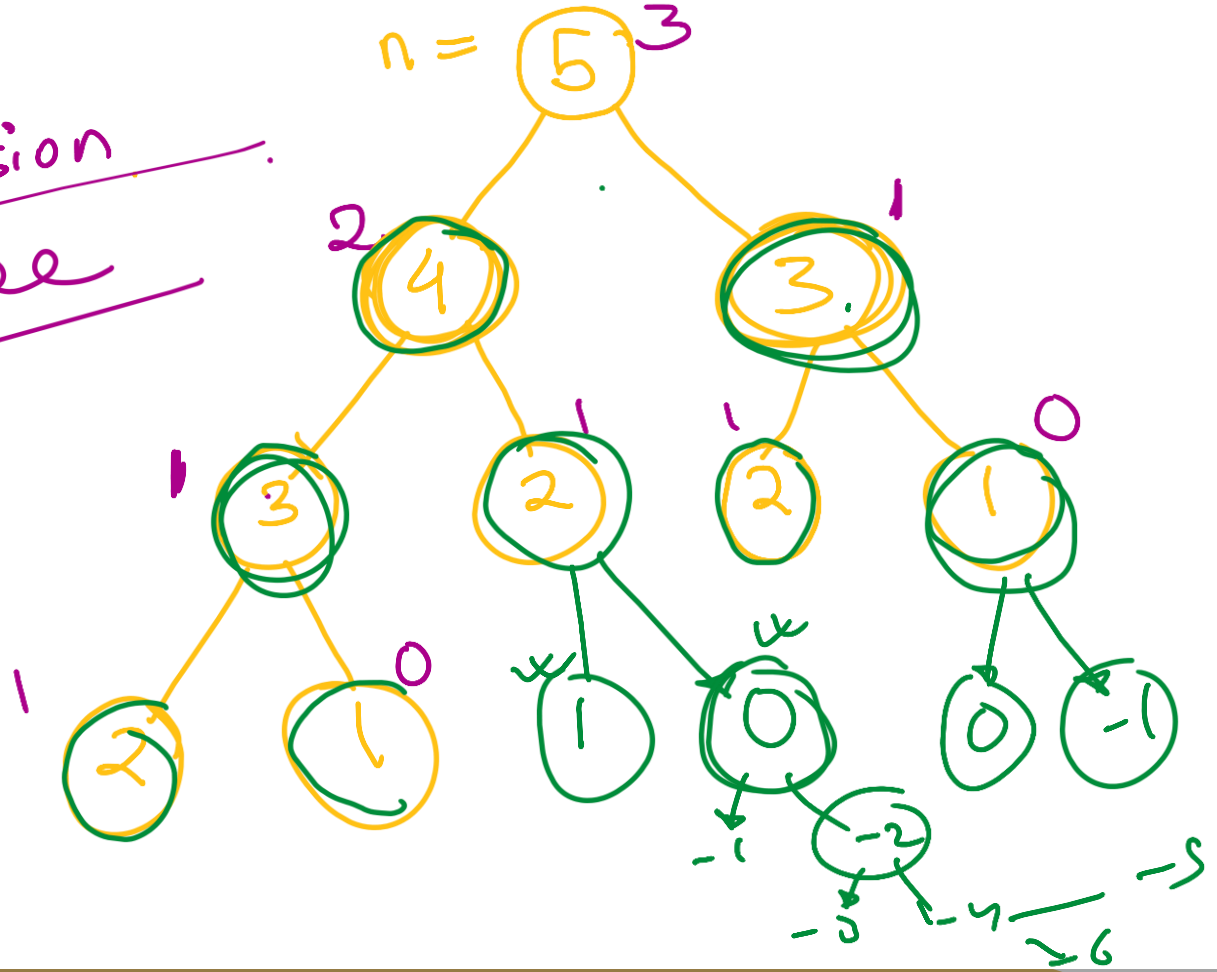
$f(n) = f(n-1) + f(n-2)$

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

base cases

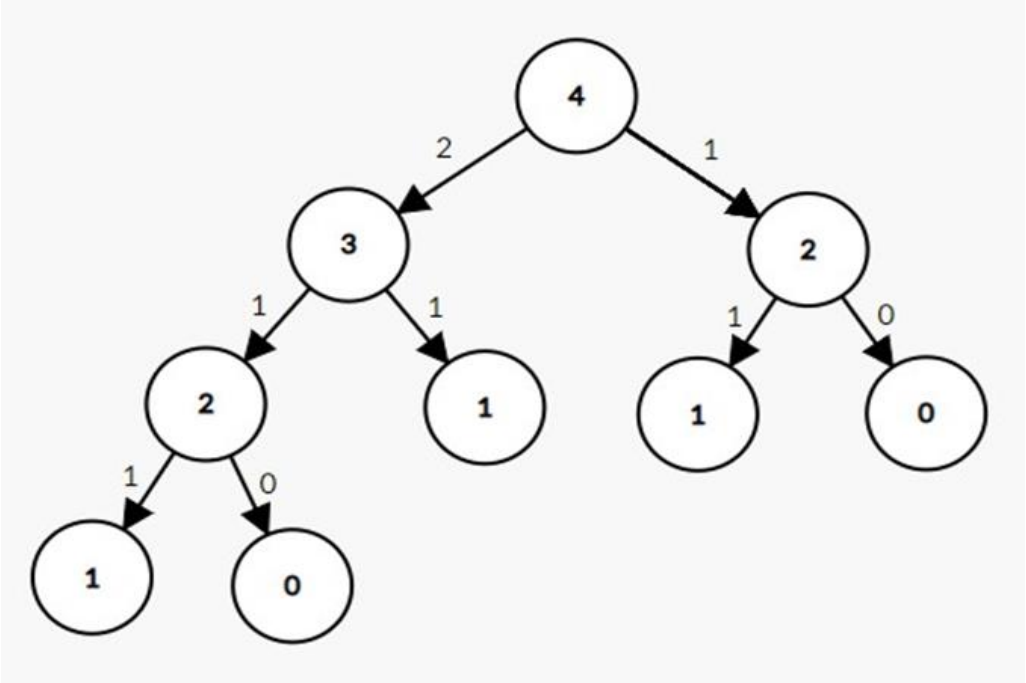

5th fib. no.
1st fib
2nd fib
0
1
5th
4th
3rd
4th
3rd
2nd
3rd
1st
2nd
3
2
1
1
1
1
0
1

Recursion Tree



Example - Nth Fibonacci Number (Contd…)

Recursion Tree for finding the 4th Fibonacci Number



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

N

N = 5

N = 4

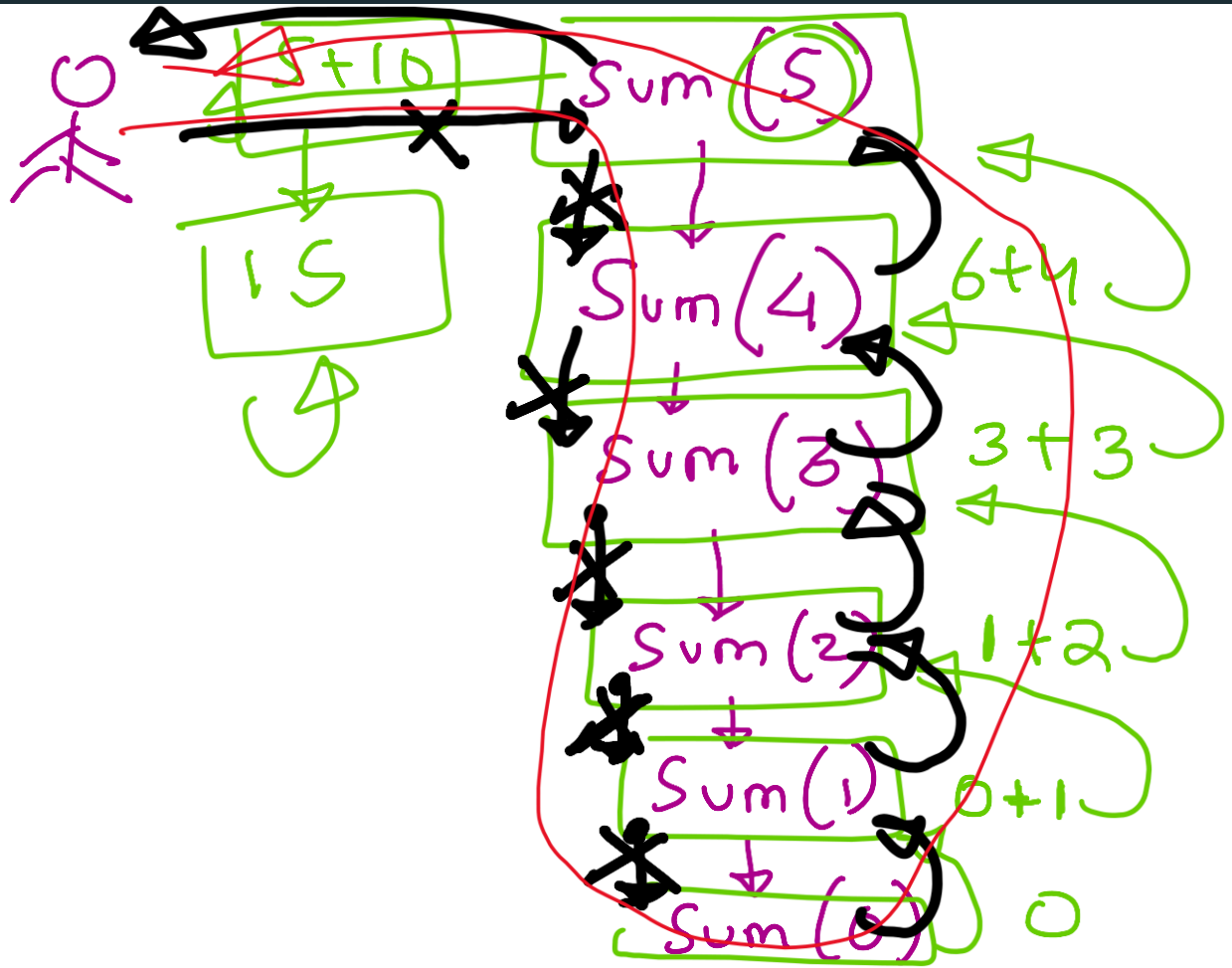
N = 3

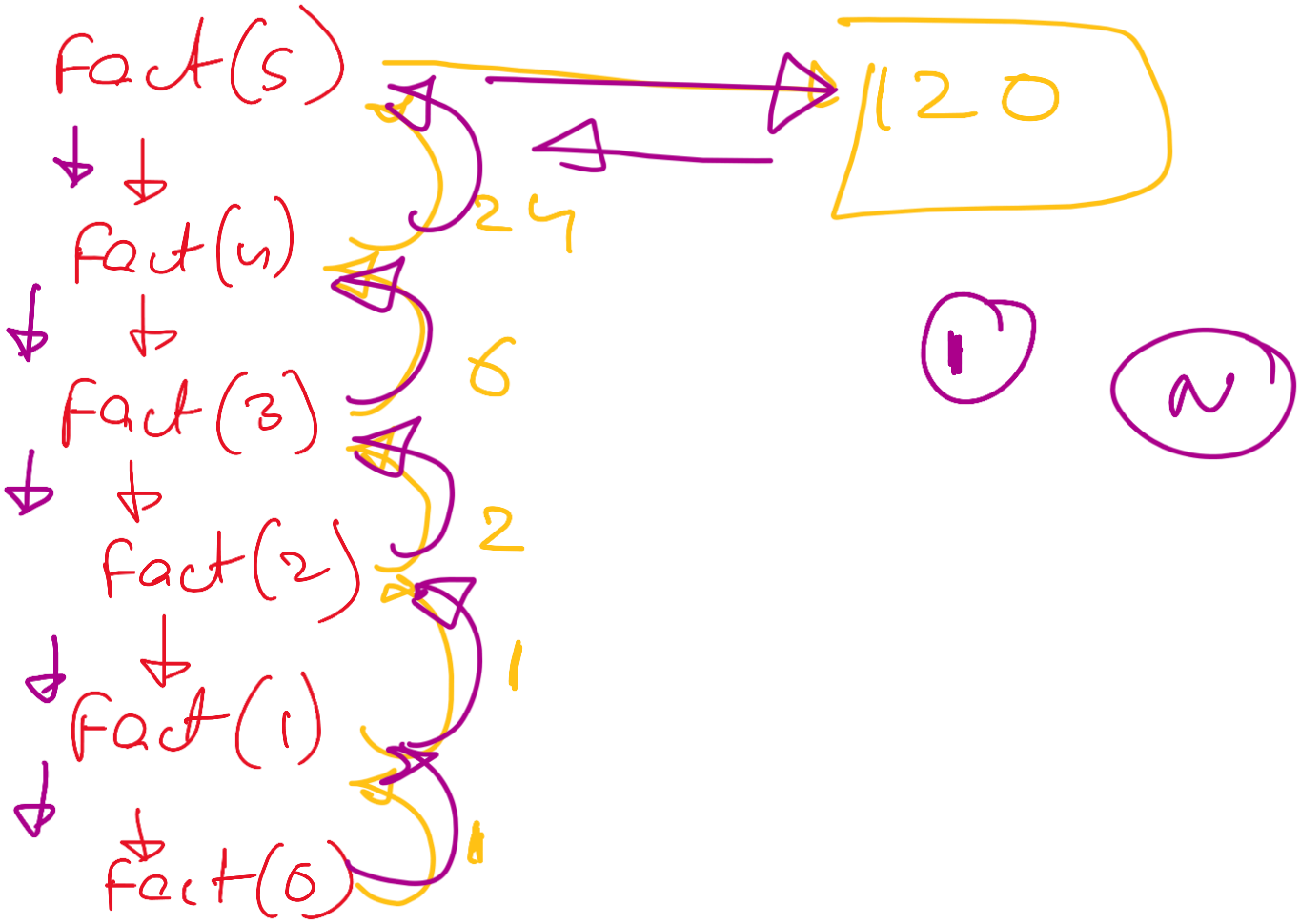
N = 2

N = 1

N = 0

0


5+10
Sum(5)
15
Sum(4)
6+4
Sum(3)
3+3
Sum(2)
1+2
Sum(1)
0+1
Sum(0)
0


fact(5)
fact(4)
fact(3)
fact(2)
fact(1)
fact(0)
120
24
6
2
1
1
1
n

Recursion Call Stack

- When a recursion is executed, the function calls are placed in a **stack**.
- The **first call is at the bottom of the stack**, and the **last call is at the top**.
- After a call is executed, it is popped from the stack, and the **returned value is passed to the current top element.**

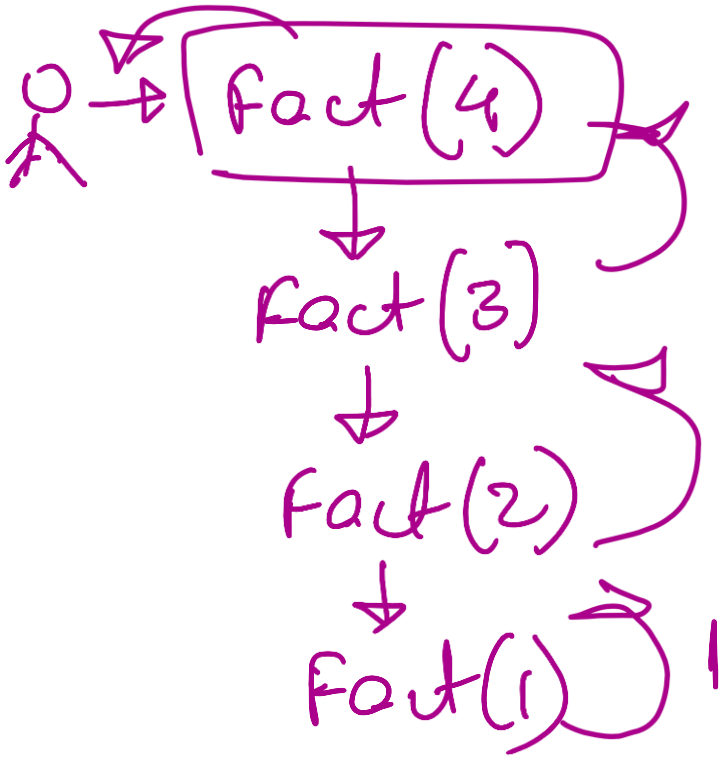
Recursion Call Stack

Question: Print factorial of a number n →

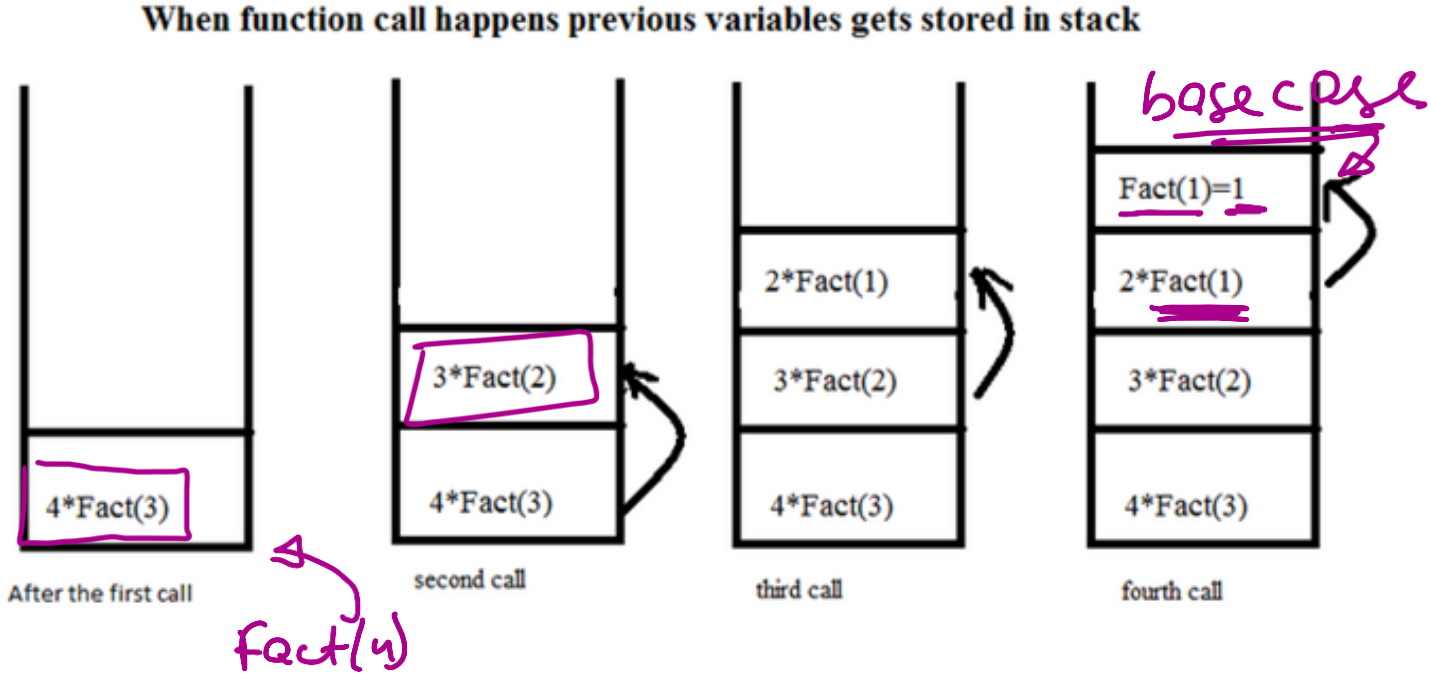
For n = 4, print 1*2*3*4 = 24

Code →

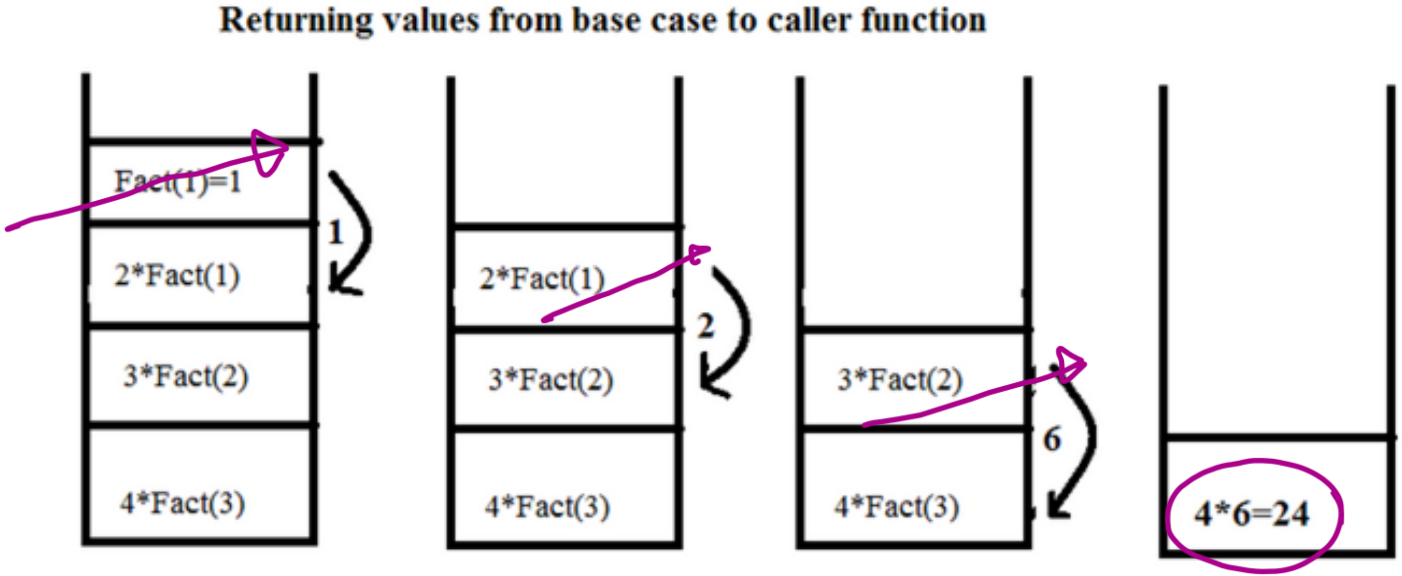
```
int Fact (int n){
    if(n<=1){
        return 1;
    }
    return n*Fact(n-1);
}
```


fact(4)
fact(3)
fact(2)
fact(1)
1

Recursion Call Stack



Recursion Call Stack



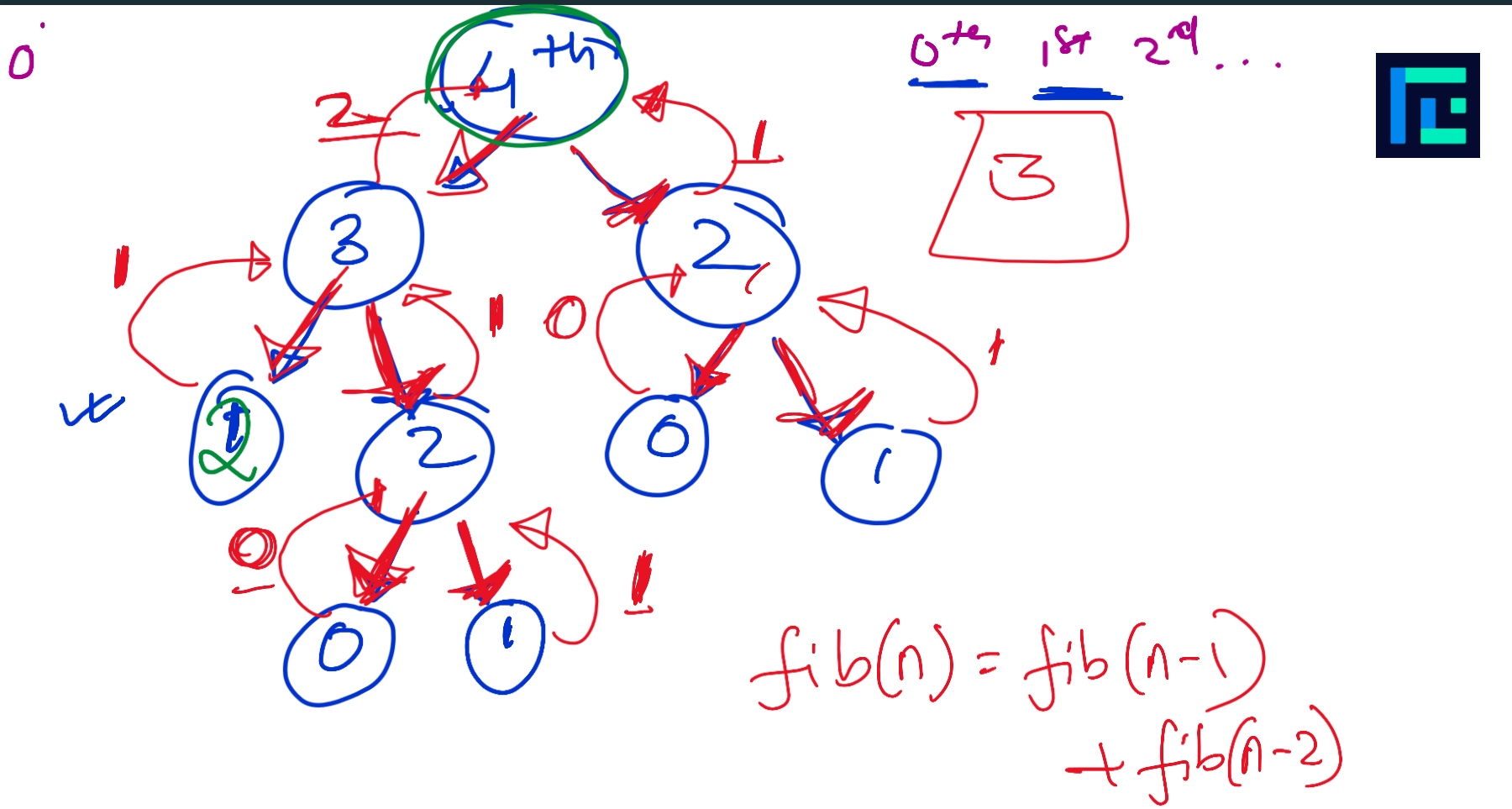
Recursion Call Stack

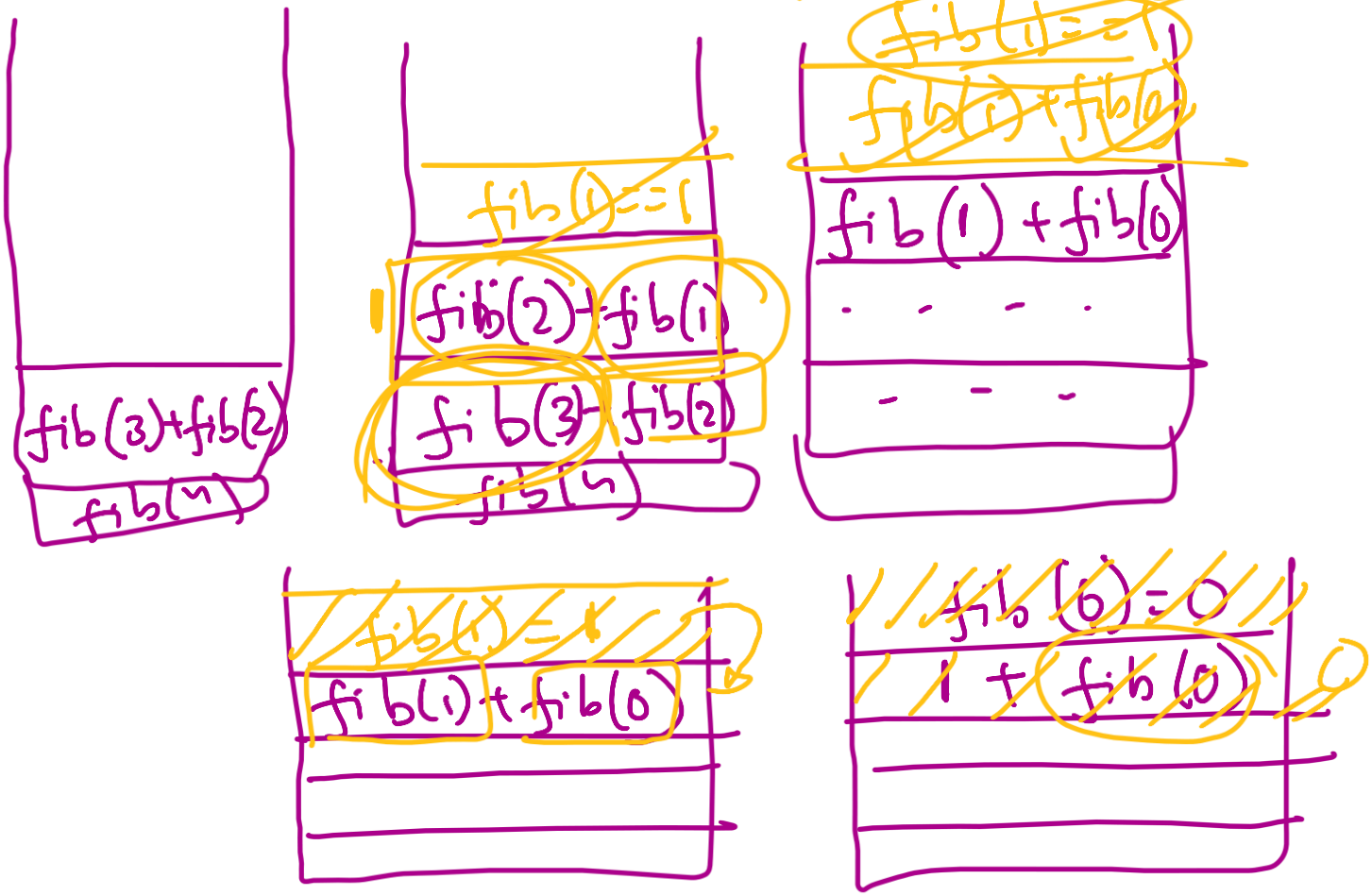
Question: Find the Nth fibonacci number

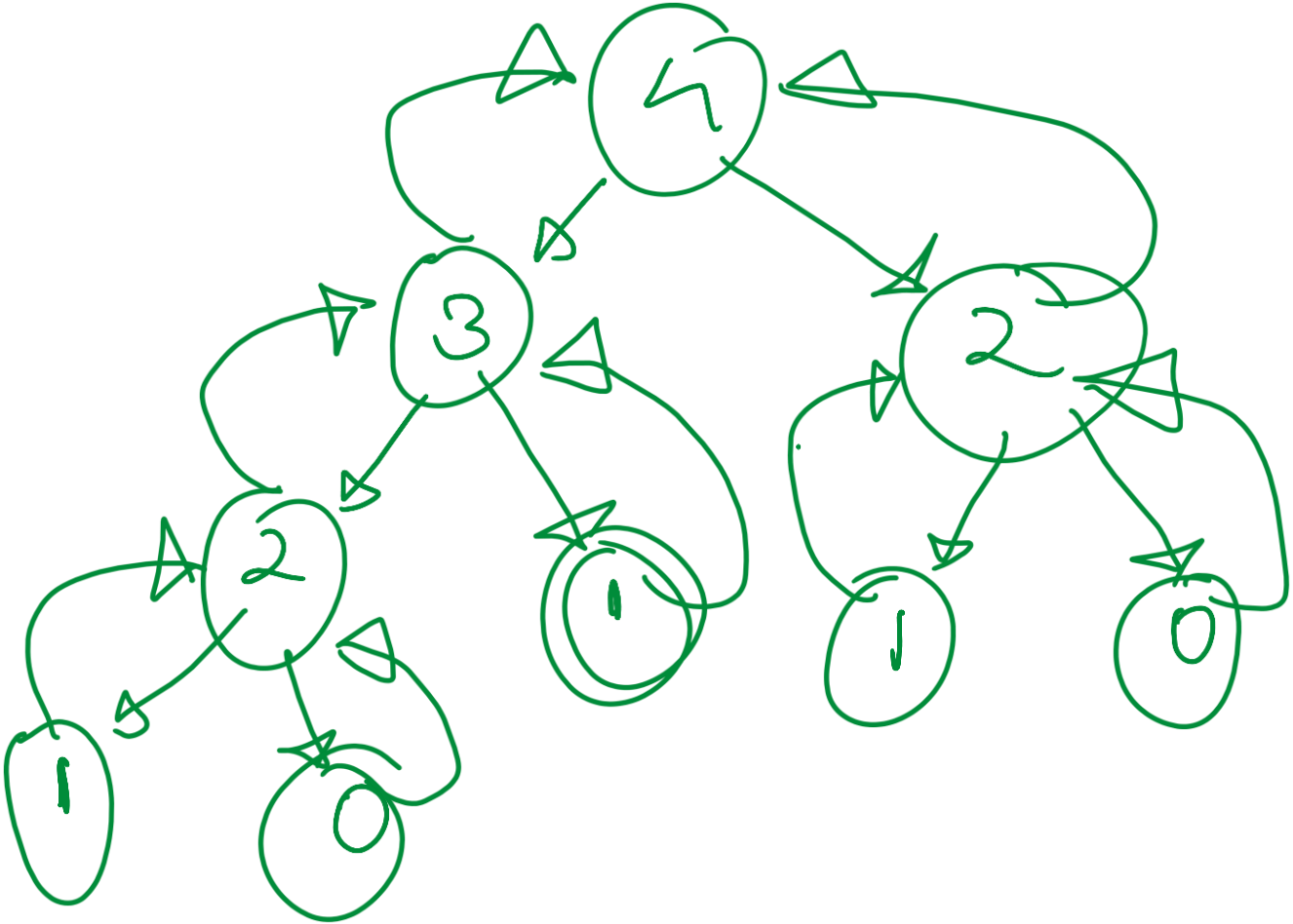
For n = 5, print 3

Code-

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


0
4th
2
3
2
1
1
0
1
2
2
0
1
0
1
0th 1st 2nd ...
3
fib(n) = fib(n-1)
+ fib(n-2)


fib(3)+fib(2)
fib(4)
fib(1)==1
fib(2)+fib(1)
fib(3)+fib(2)
fib(4)
fib(1)==1
fib(1)+fib(0)
fib(1)+fib(0)
fib(1)==1
fib(1)+fib(0)
fib(0)==0
1+fib(0)


4
3
2
2
1
1
0
1
0