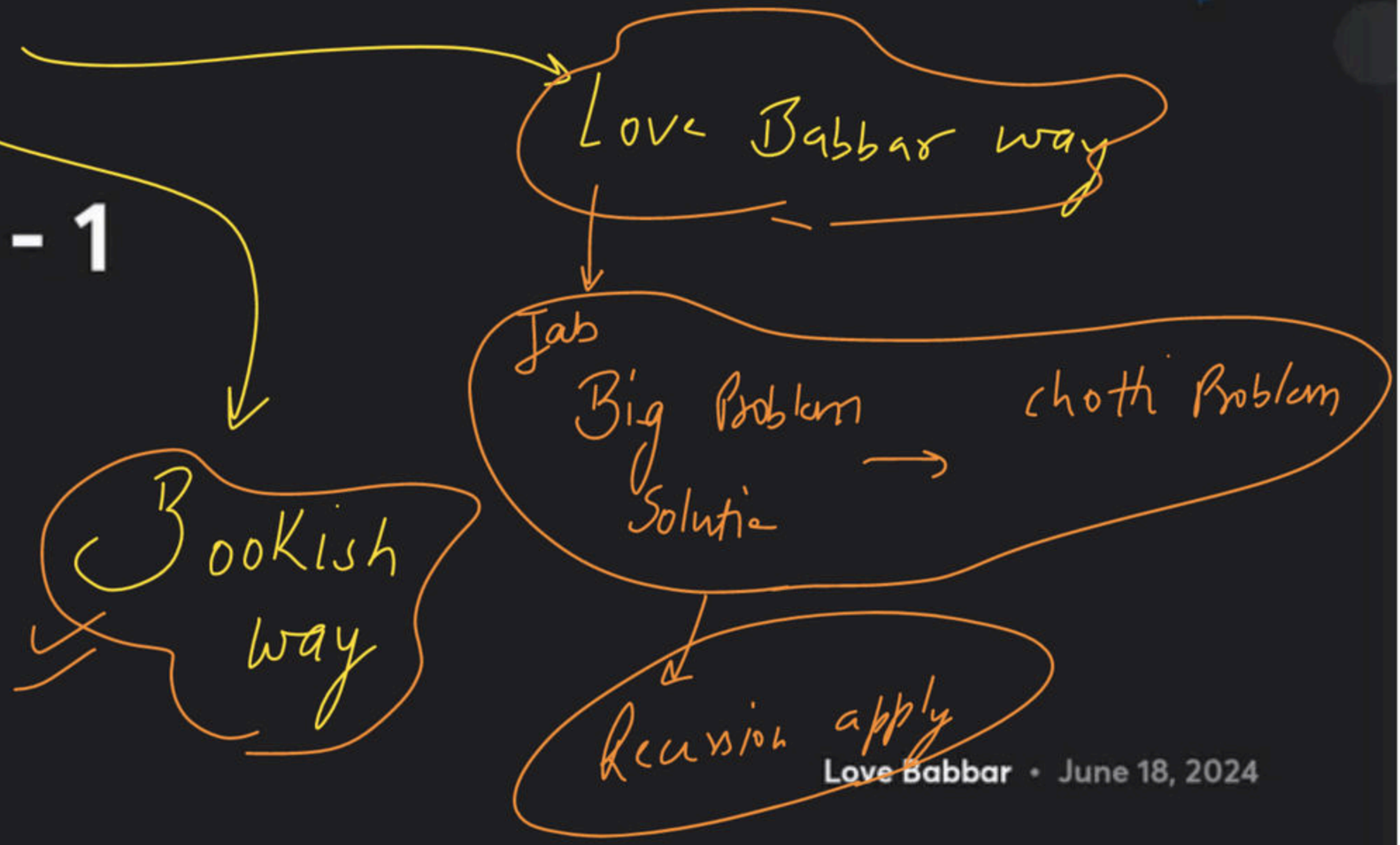
Recursion Class - 1

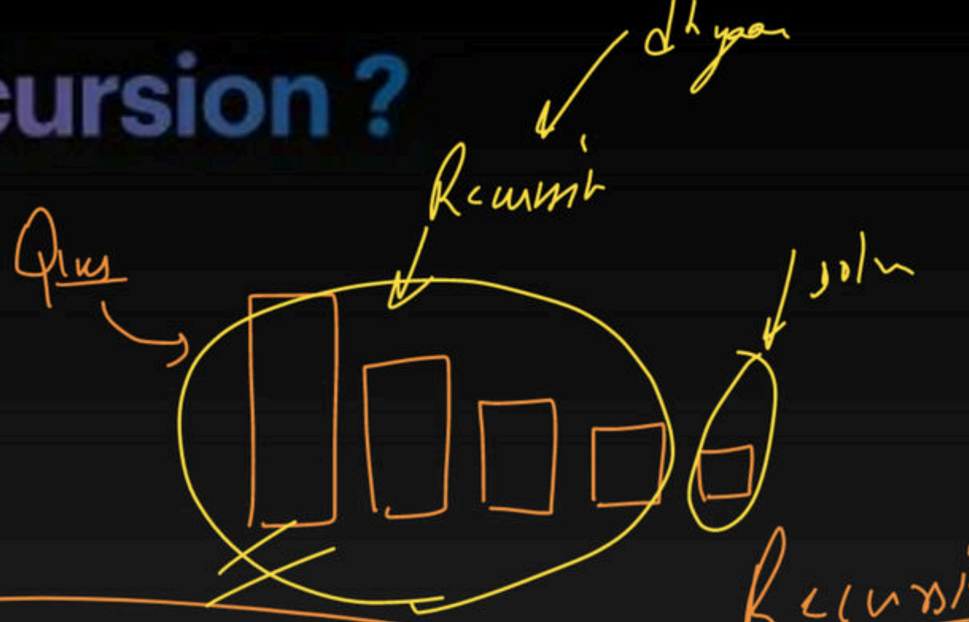
Special class



Love Babbar • June 18, 2024

What is Recursion ?

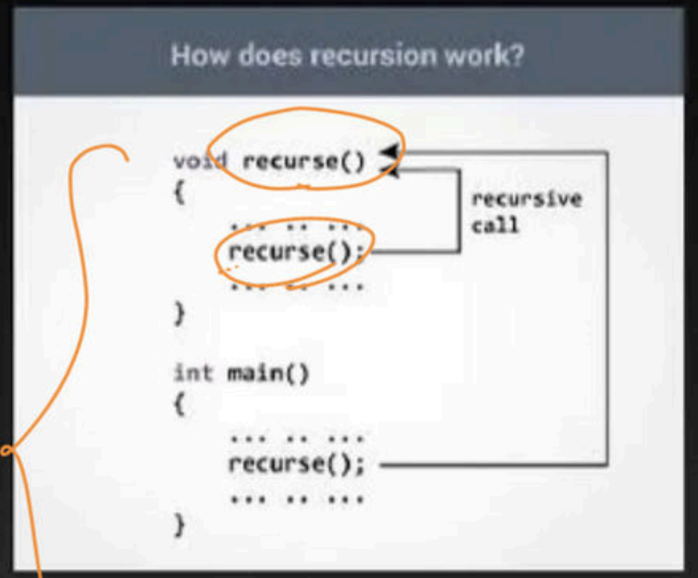
TRUST



Magical Line :-

1 case hum solve karenge,

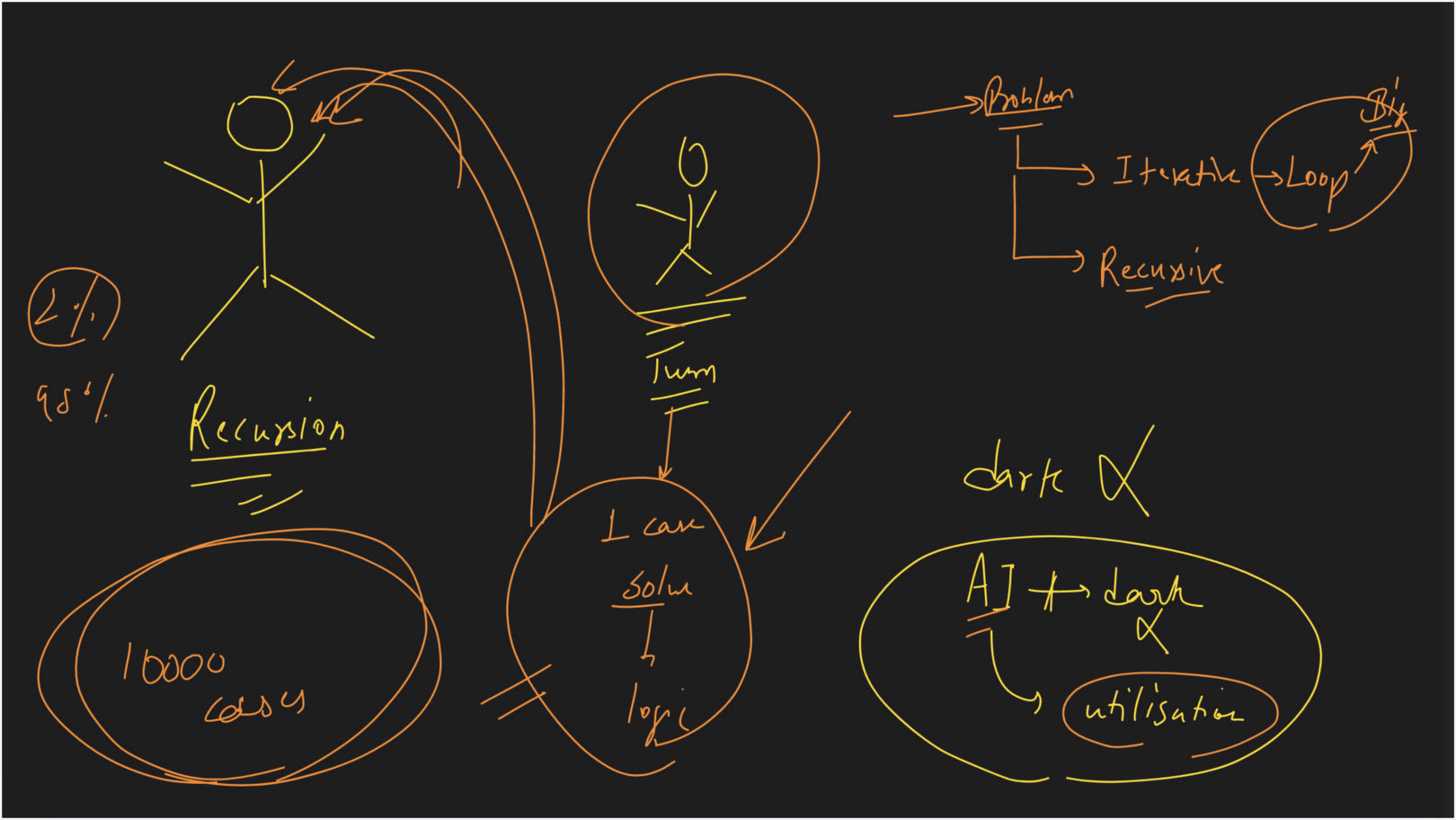
Baaki Recursion sambhal lega

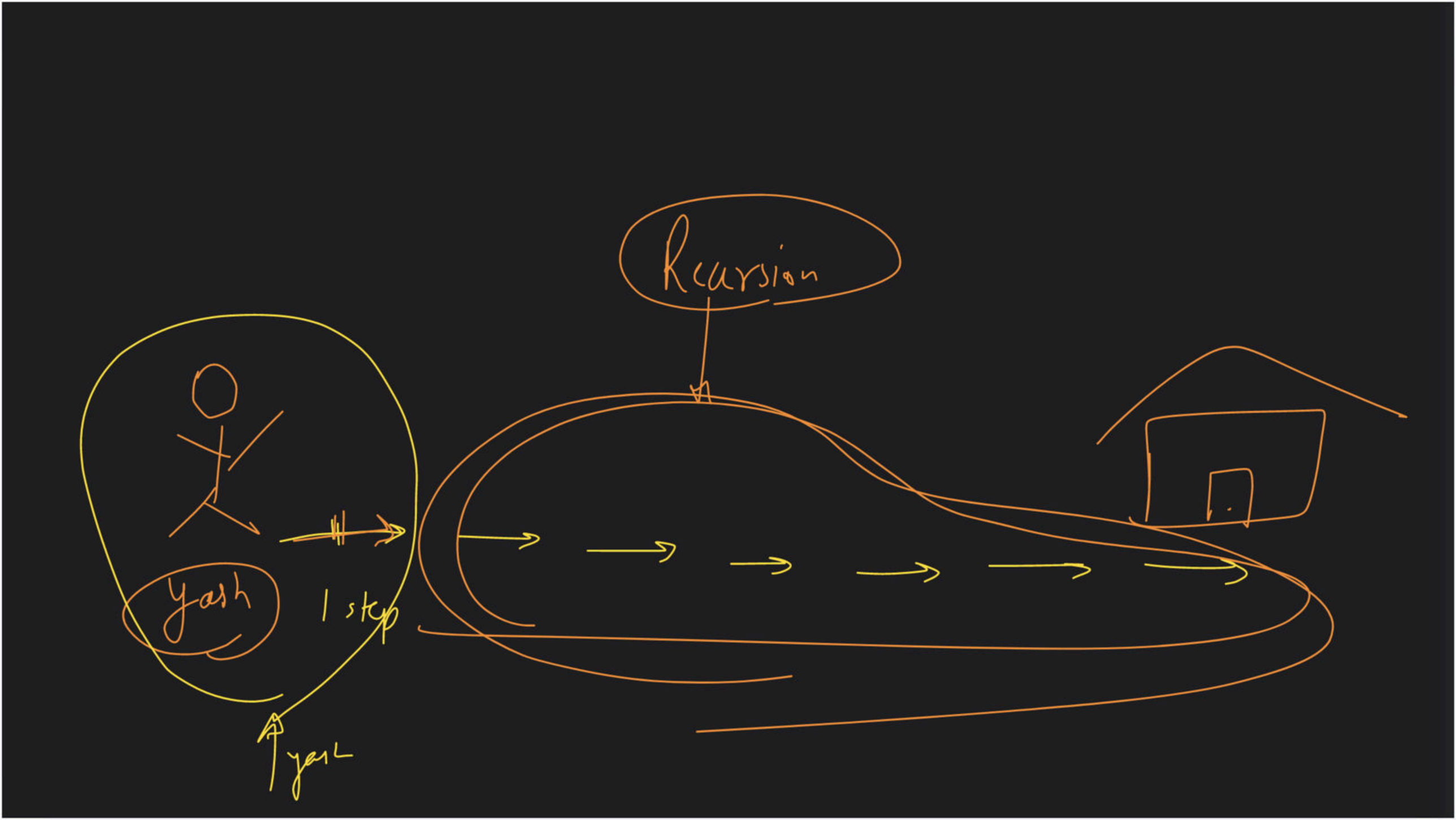


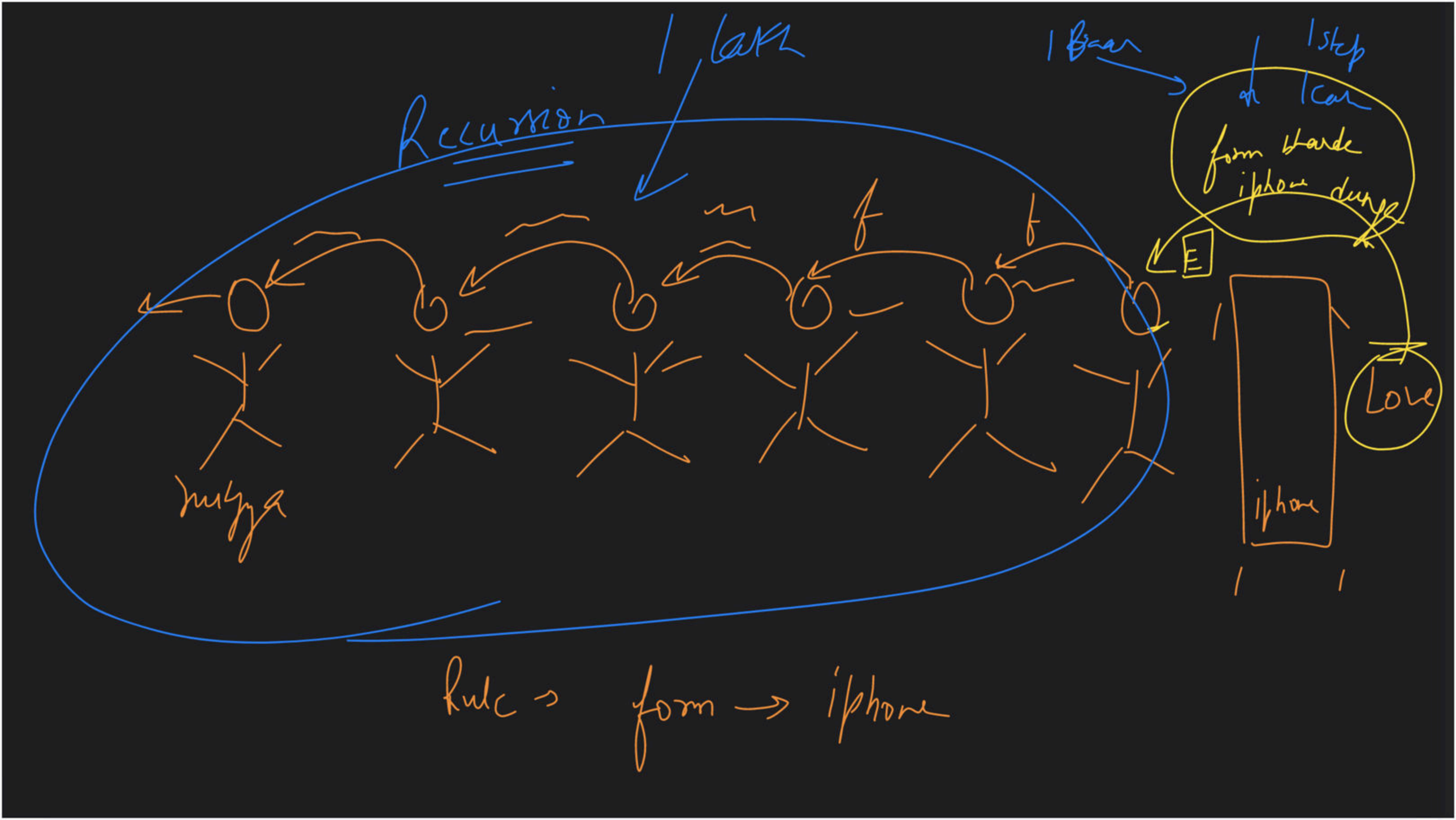
- What ?
- When to apply ?
- How to apply ?

Recursion

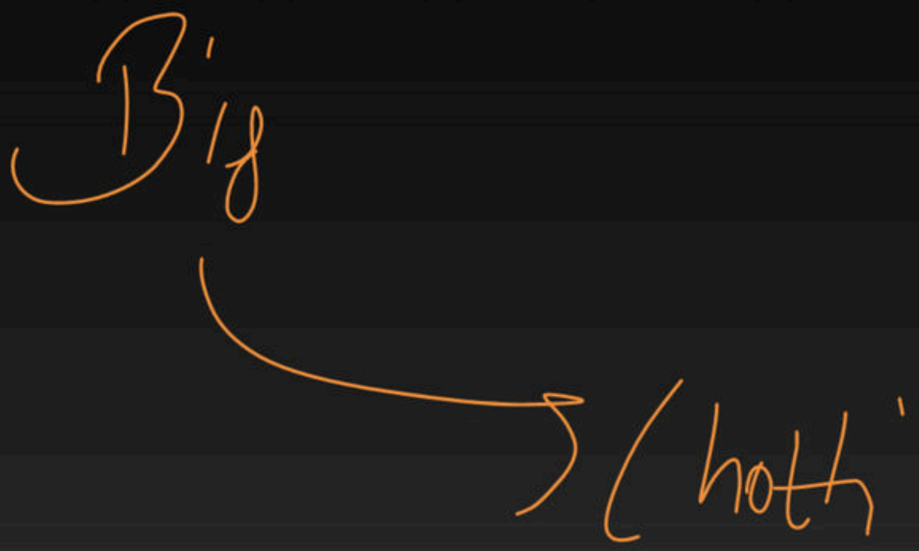
dry run

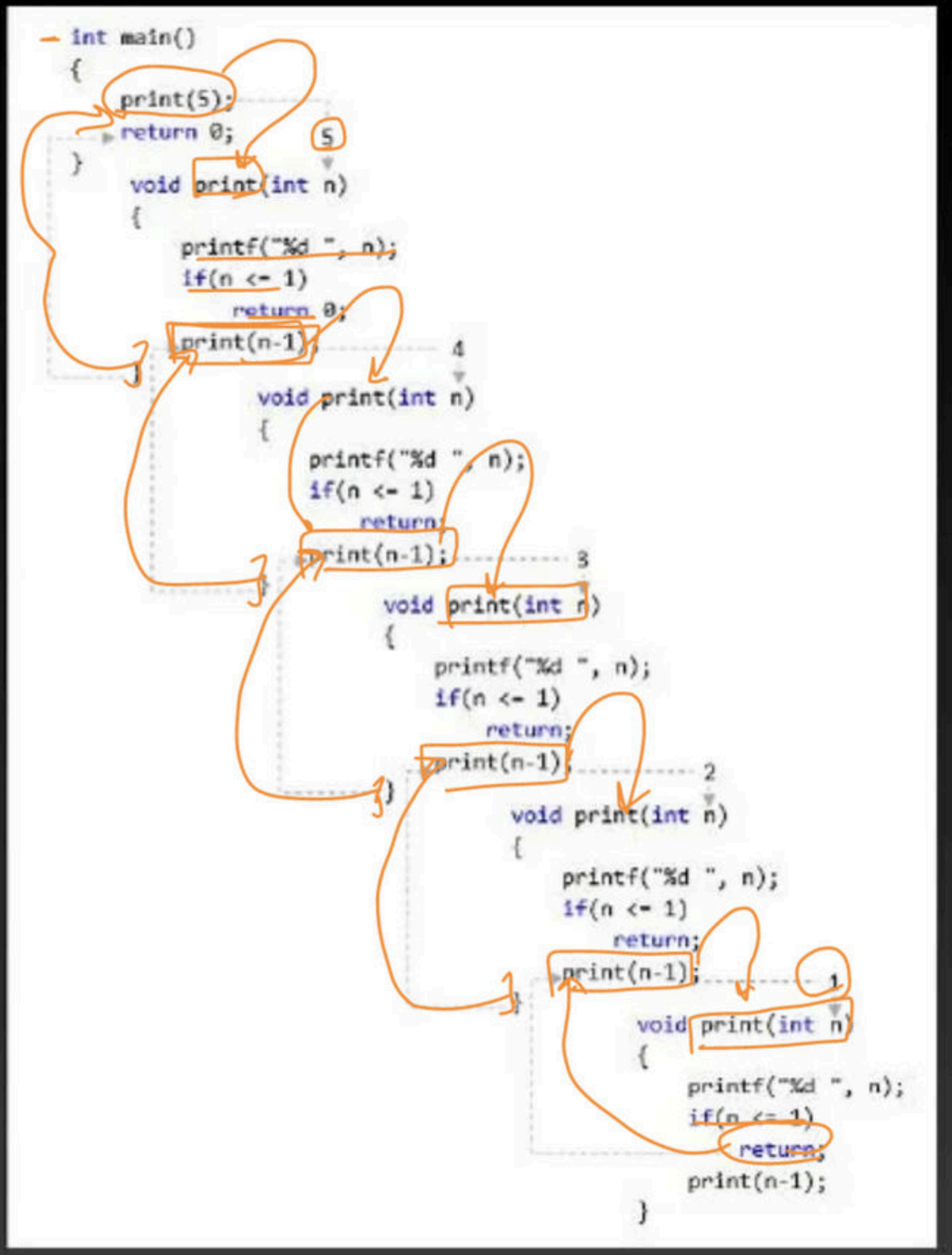

2%
98%
Recursion
10000
cases
Turn
I can
solve
logic
Problem
Iterative
Loop
Big
Recursive
dark
AI
dark
utilisation


Recursion
yash
1 step
yash


Recursion
f
f
E
Love
iphone
Rule → form → iphone

What is Recursion ?





factorial

$5!$

$5! = 5 \times 4 \times 3 \times 2 \times 1$

$4! = 4 \times 3 \times 2 \times 1$

$9! = 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$

$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$

$6!$

Big

$7! = 7 \times 6!$

Small

$n = 7$

factorial(7) = 7 * factorial(6)

factorial(n) = n * factorial(n-1)

Power 2^4

$$2^3 = 2 \times 2 \times 2$$

$$2^7 = 2 \times \underbrace{2 \times 2 \times 2 \times 2 \times 2 \times 2}_{2^6}$$

$$2^7 = 2 \times 2^6$$

power(7) = 2 x power(6)

6 × 5 × 4 × 3 × 2

Recursive / Recurrence relation

factorial(7) = 7 * (720) ⇒

factorial(7) = 7 * factorial(6)

factorial(n) = n * factorial(n-1)

Bij: factorial(n)

maine Kiya: n

chotti Recursion: factorial(n-1)

power(7) = 2 * power(6)

chotti

Recursion

$2^6 \rightarrow$

= 2 * 64

= 128

10.1.

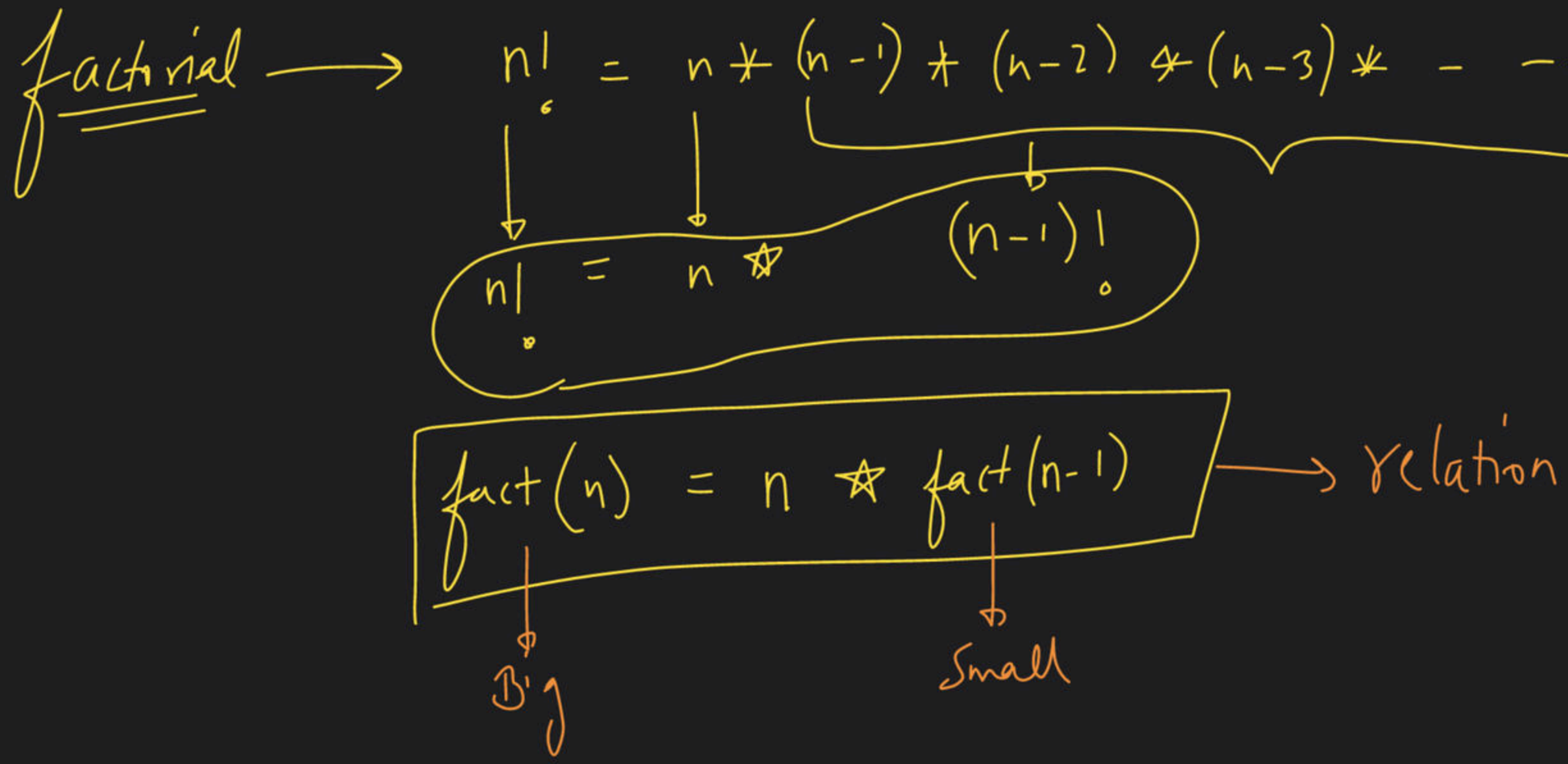
90

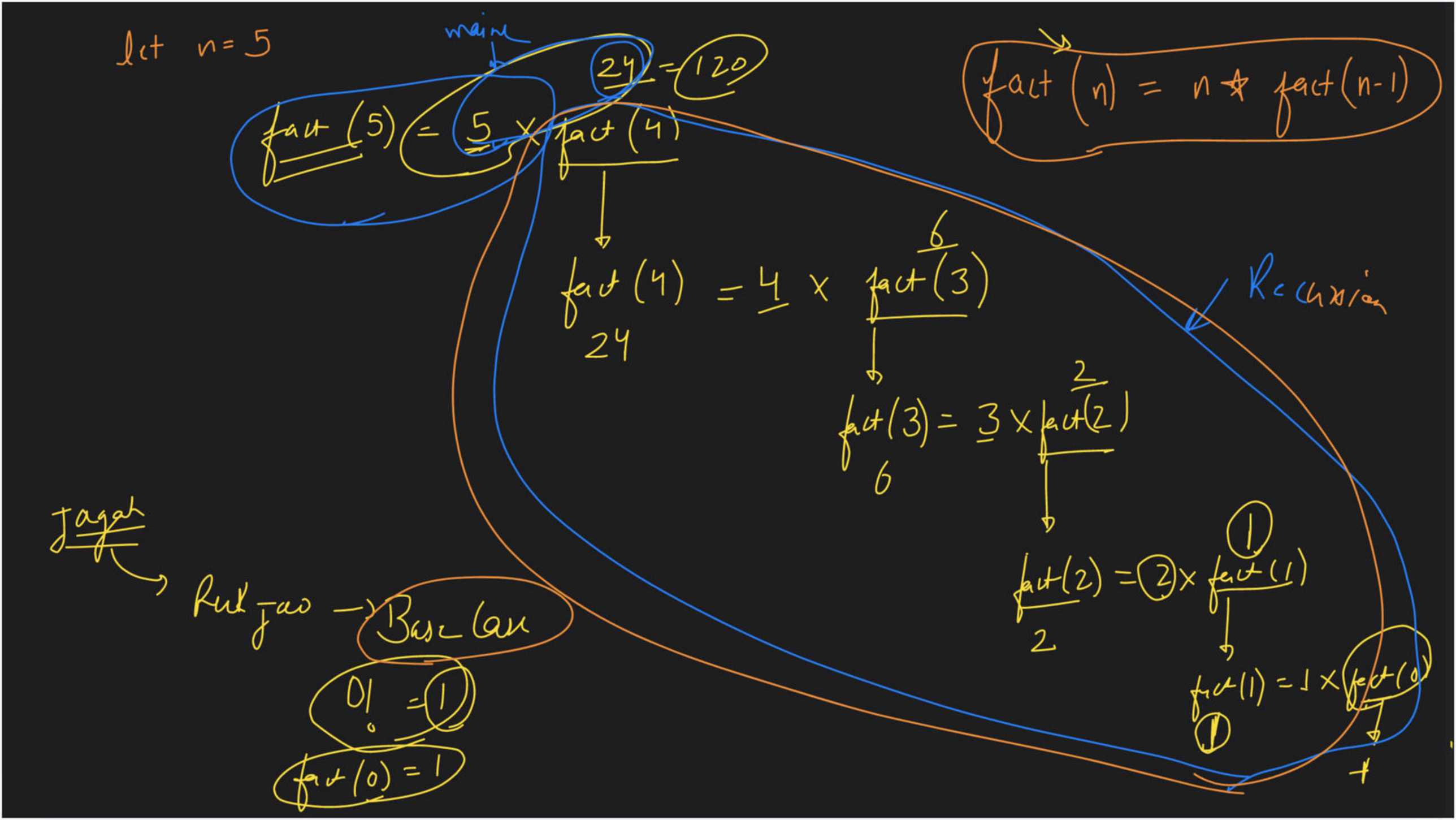
Recursion Components :-

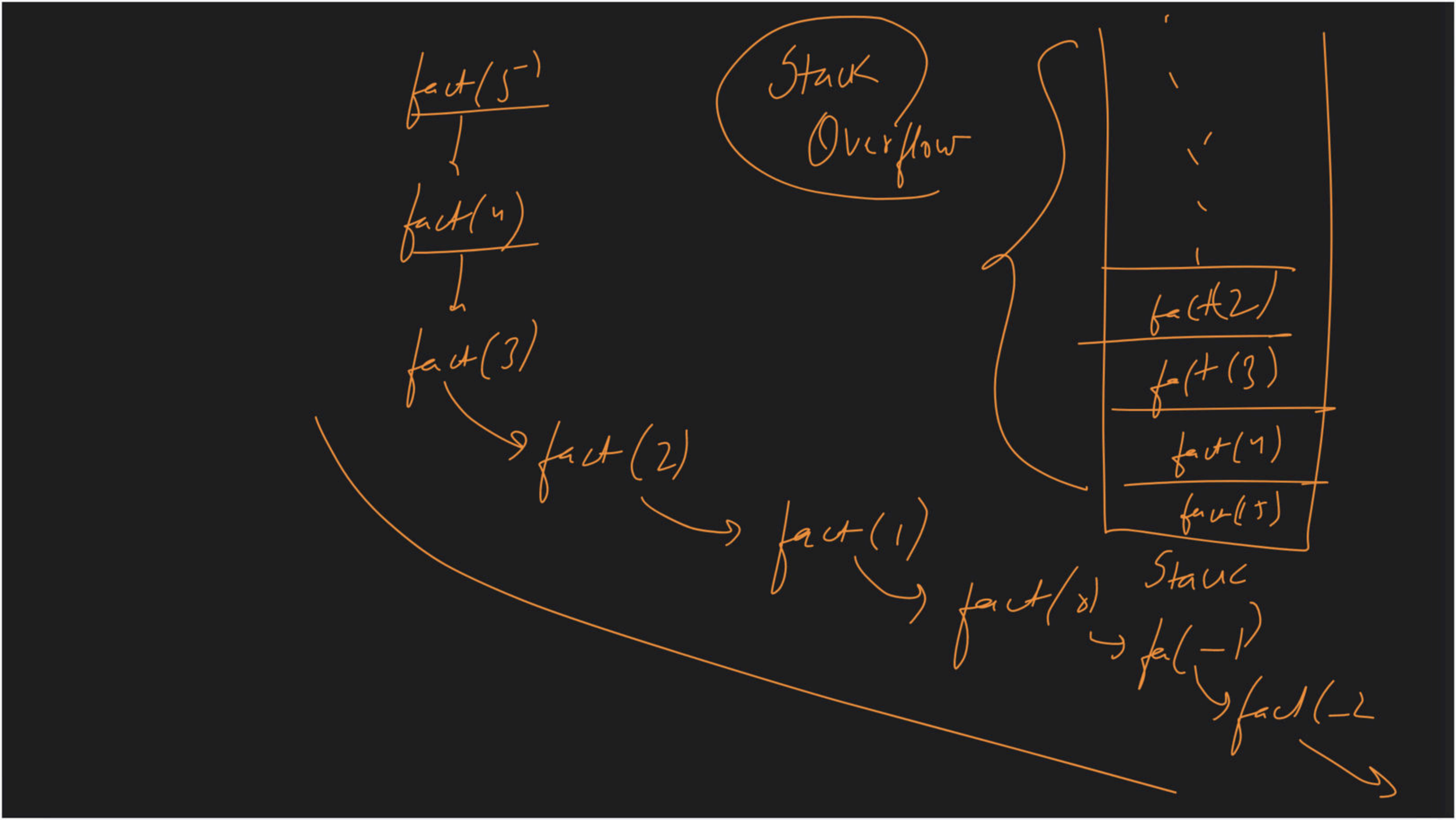
3

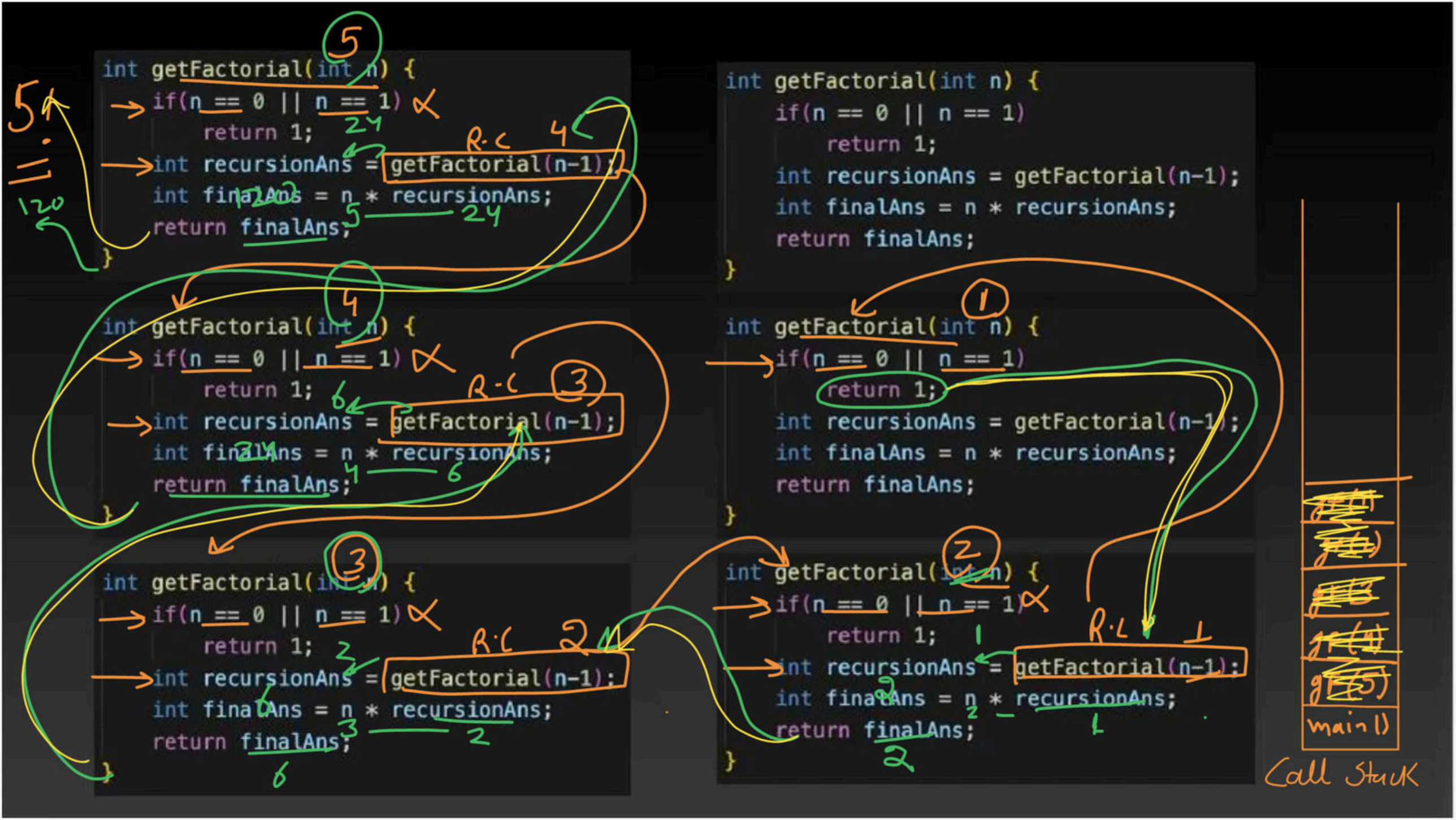
- Basic Case / Condition
  - Ruknaa kab hai
  - Mandatory
- Recursive Call
  - Mandatory
  - function call
- Processing
  - Optional

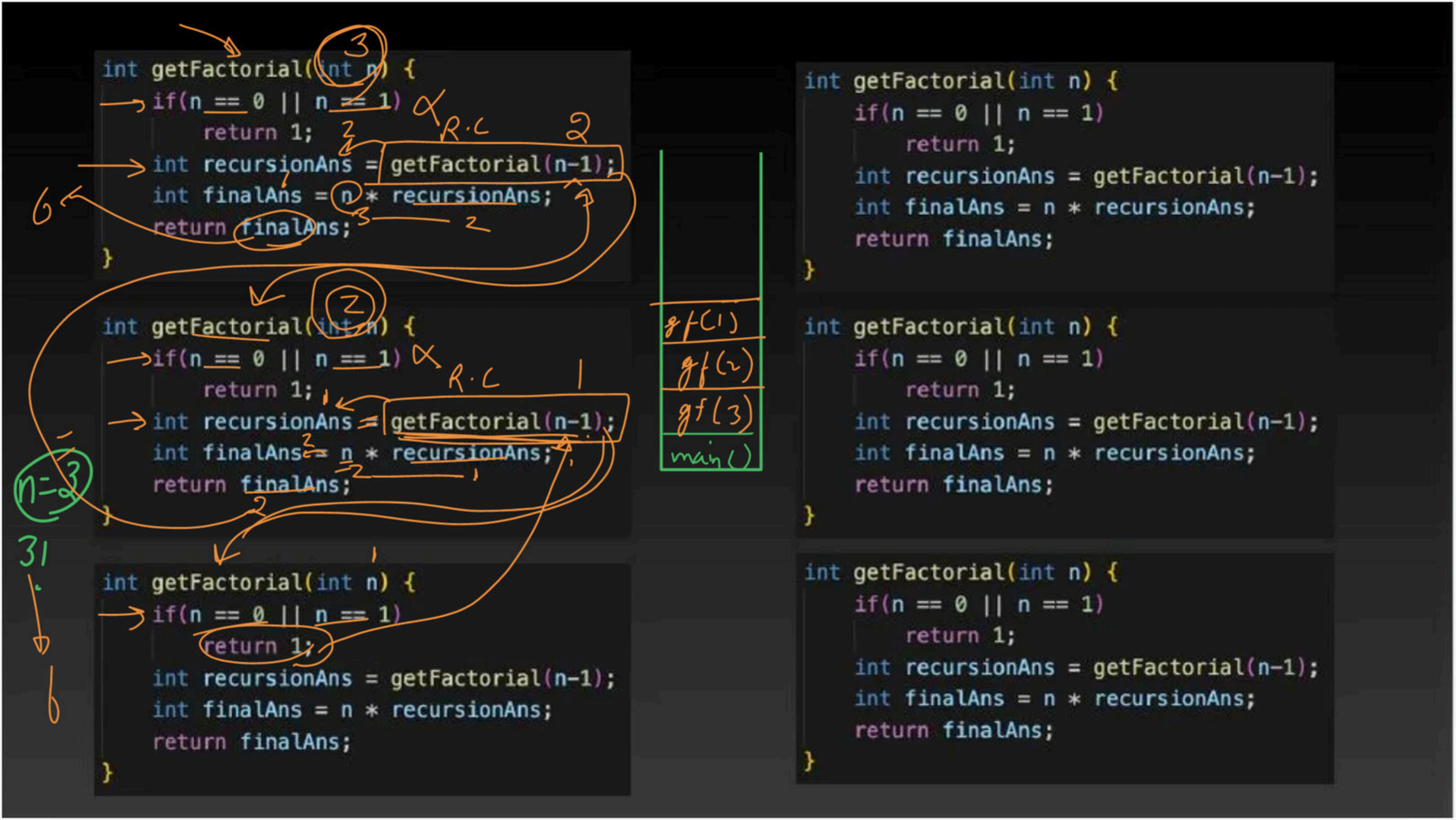
Sort (v- ~ -)


factorial
n! = n * (n-1) * (n-2) * (n-3) * - -- 3 * 2 * 1
n! = n * (n-1)!
fact(n) = n * fact(n-1)
relation
Big
Small


let n = 5
main
fact(5) = 5 x fact(4)
24 = 120
fact(n) = n * fact(n-1)
fact(4) = 4 x fact(3)
24
6
fact(3) = 3 x fact(2)
6
2
fact(2) = 2 x fact(1)
2
1
fact(1) = 1 x fact(0)
1
1
Recursion
Jagah
Kuch jaao → Base Case
0! = 1
fact(0) = 1


fact(5^-1)
fact(4)
fact(3)
fact(2)
fact(1)
fact(0)
fact(-1)
fact(-2
Stack Overflow
fact(2)
fact(3)
fact(4)
fact(5)
Static


int getFactorial(int n) {
if(n == 0 || n == 1)
return 1;
int recursionAns = getFactorial(n-1);
int finalAns = n * recursionAns;
return finalAns;
}
R.C
R.L
5
4
3
2
1
120
24
6
2
1
main()
Call Stack




```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

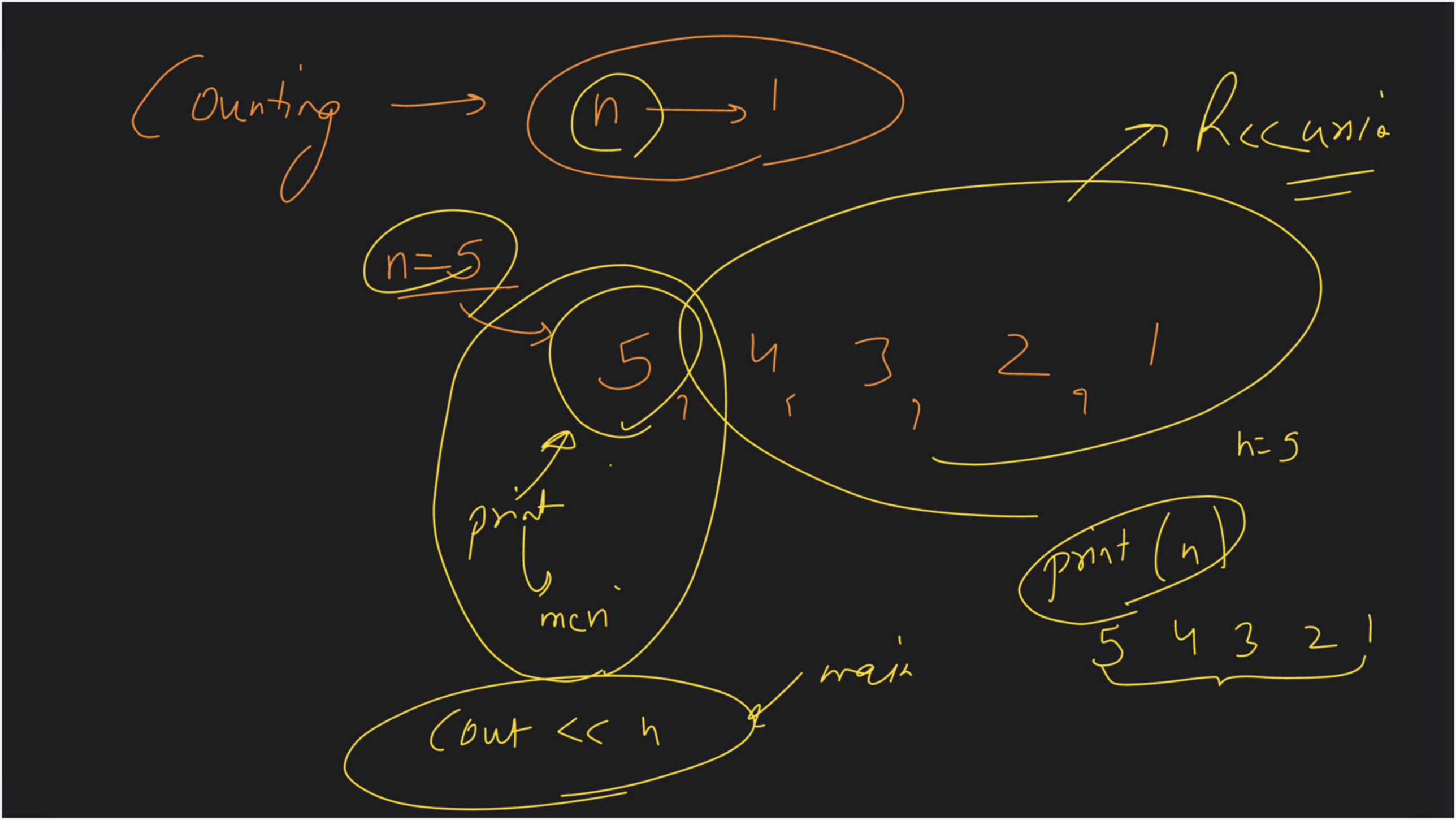
```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

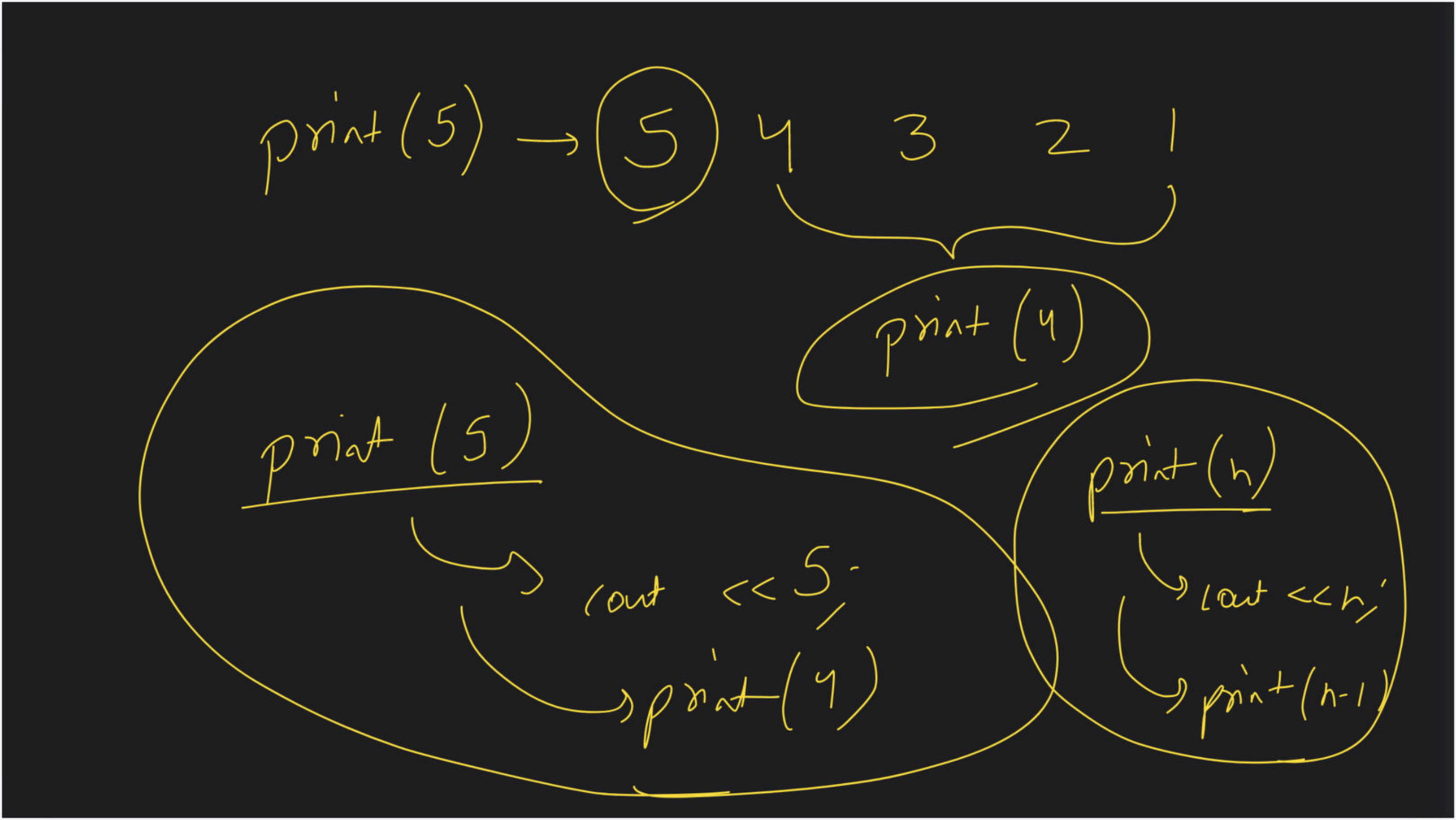
```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

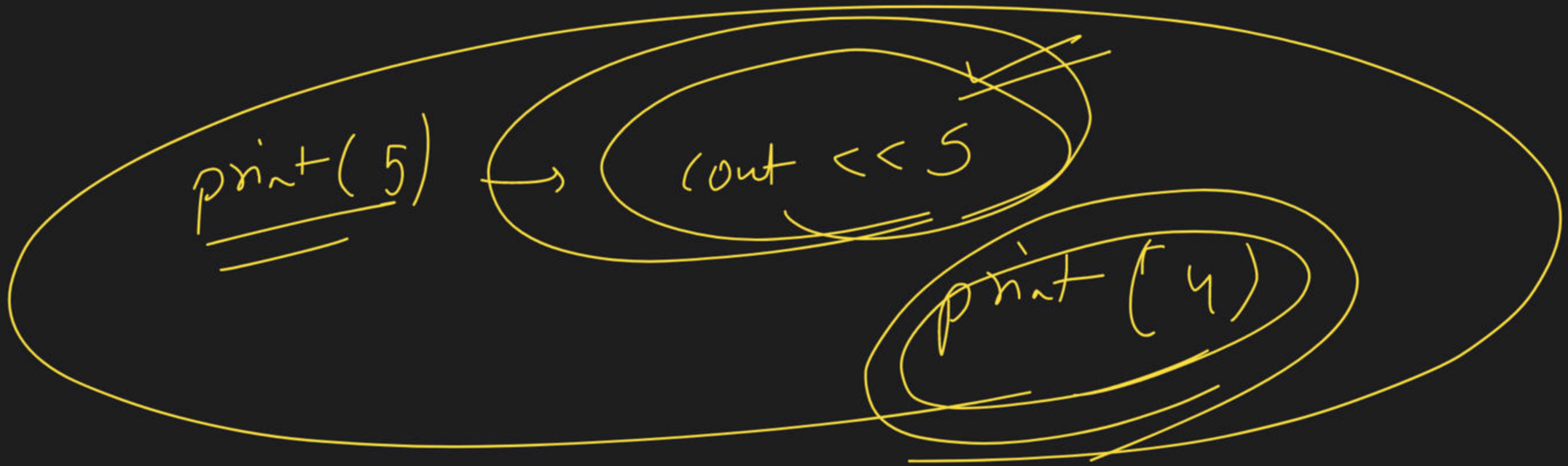
```
int getFactorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    int recursionAns = getFactorial(n-1);
    int finalAns = n * recursionAns;
    return finalAns;
}
```

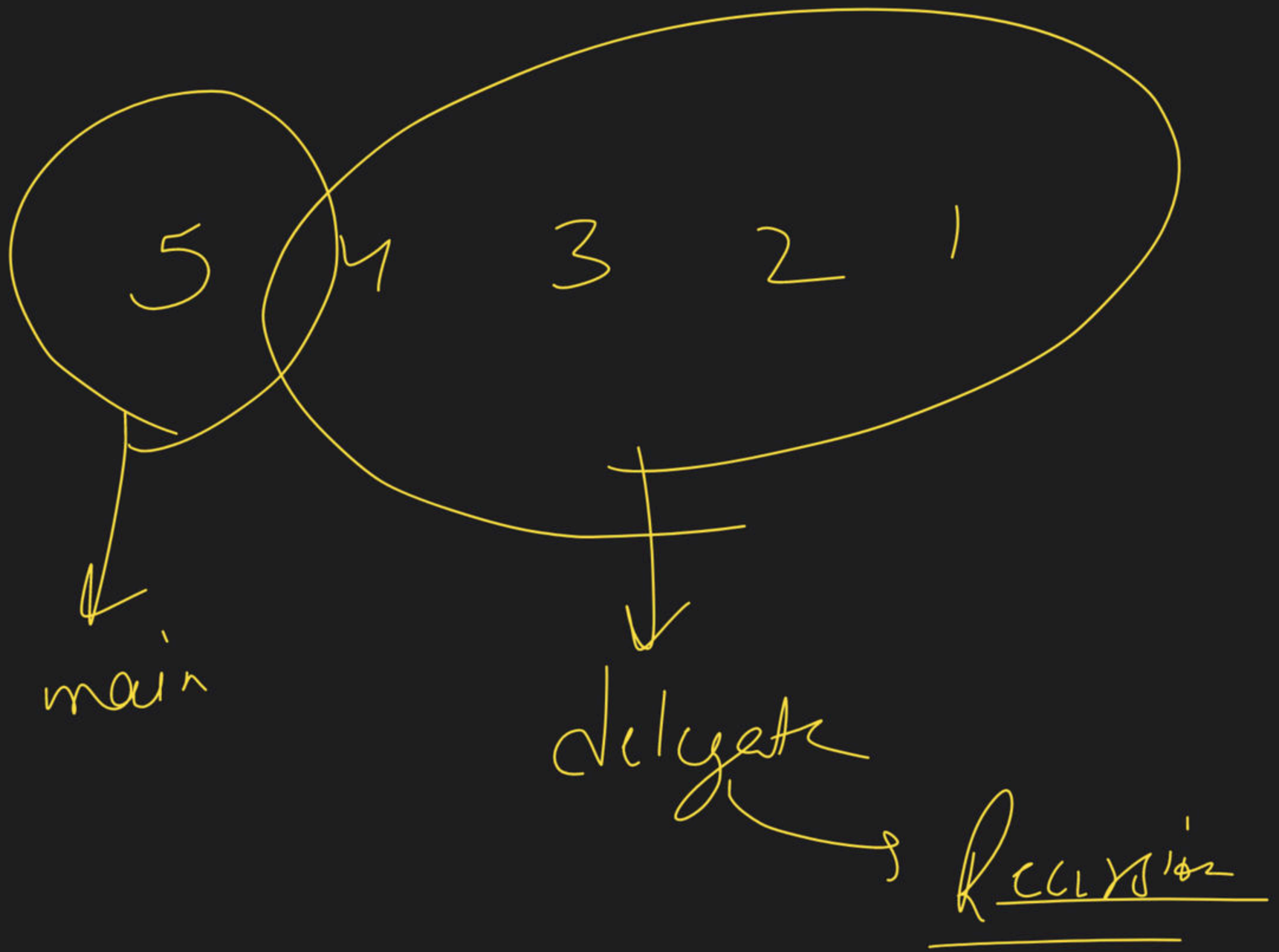

Counting
n
1
Recursio
n=5
5
4
3
2
1
print
n=5
print (n)
5 4 3 2 1
main
cout << n

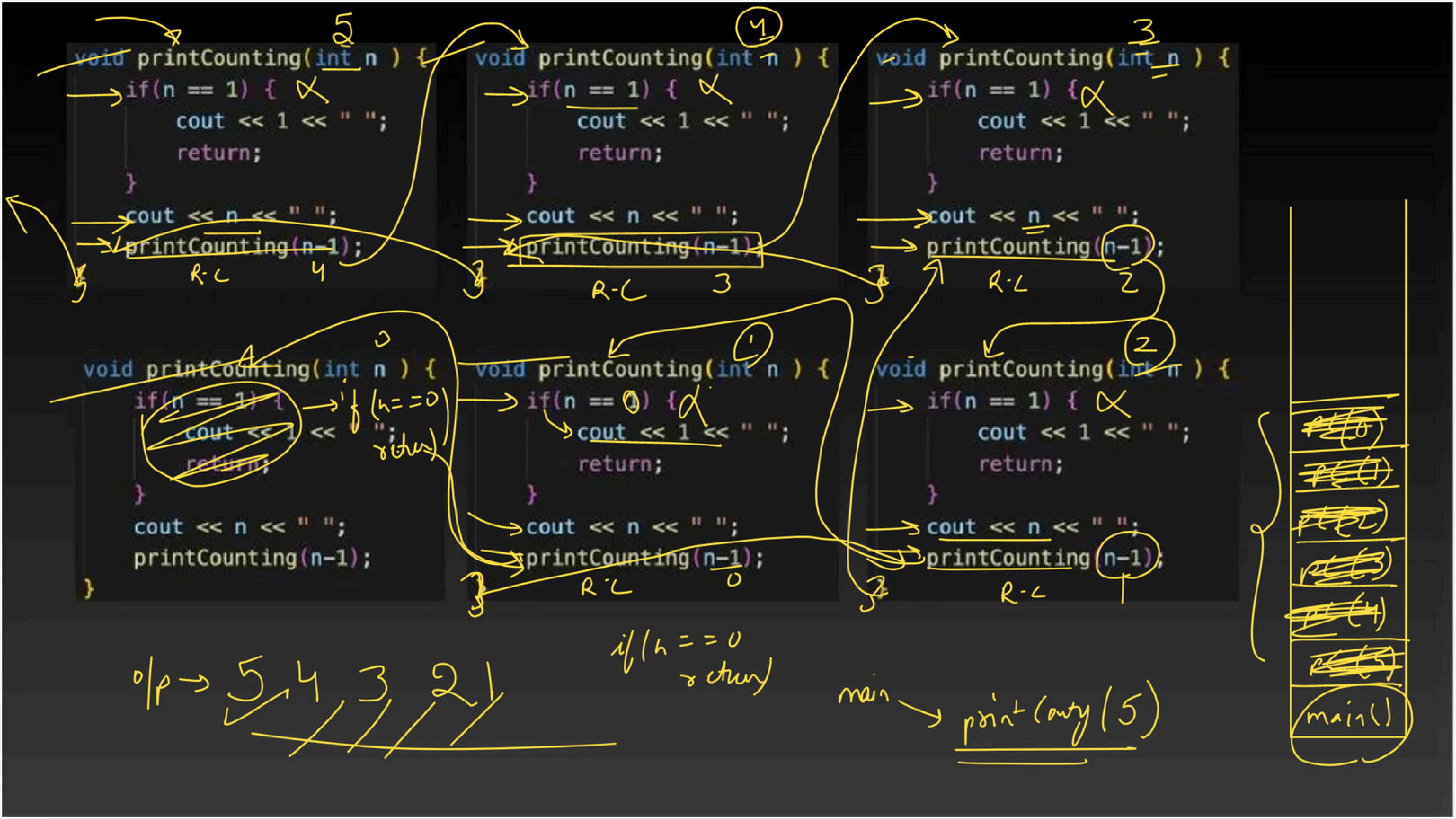

print(5) → 5 4 3 2 1
print(4)
print(5)
cout << 5;
print(4)
print(n)
cout << n;
print(n-1)

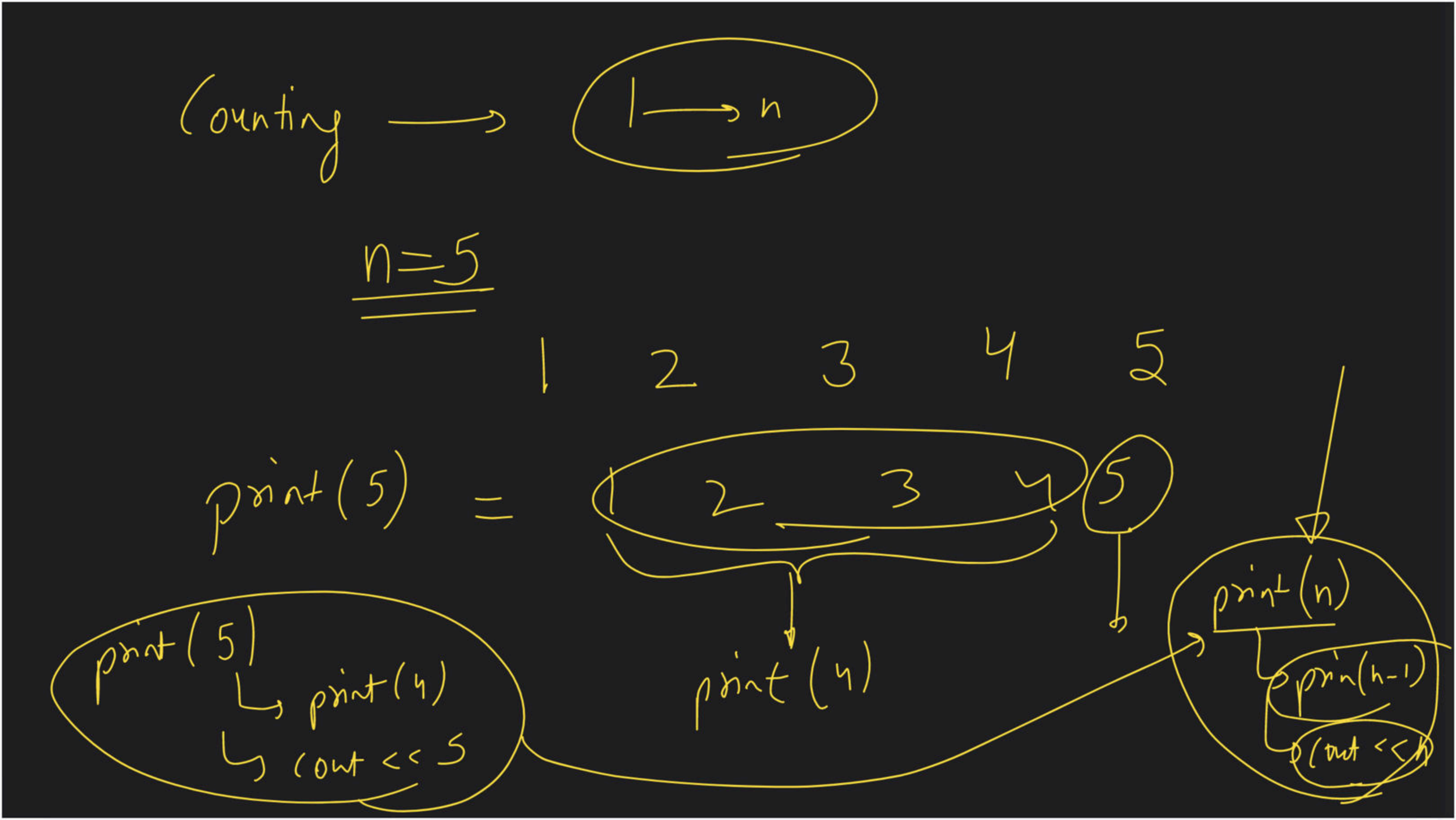
print(5) → 5 4 3 2 1

print(4) → 4 3 2 1 → Rec




5
4
3
2
1
main
delgate
Recursion


void printCounting(int n) {
if(n == 1) {
cout << 1 << " ";
return;
}
cout << n << " ";
printCounting(n-1);
}
R-C
if(n==0) return
o/p → 5 4 3 2 1
main → print Counting(5)
main()


Counting
1 → n
n=5
1 2 3 4 5
print(5) =
1 2 3 4
5
print(4)
print(5)
print(4)
cout << 5
print(n)
print(n-1)
cout << n

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```


```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    cout << n << " ";
    printCounting(n-1);
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```


```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

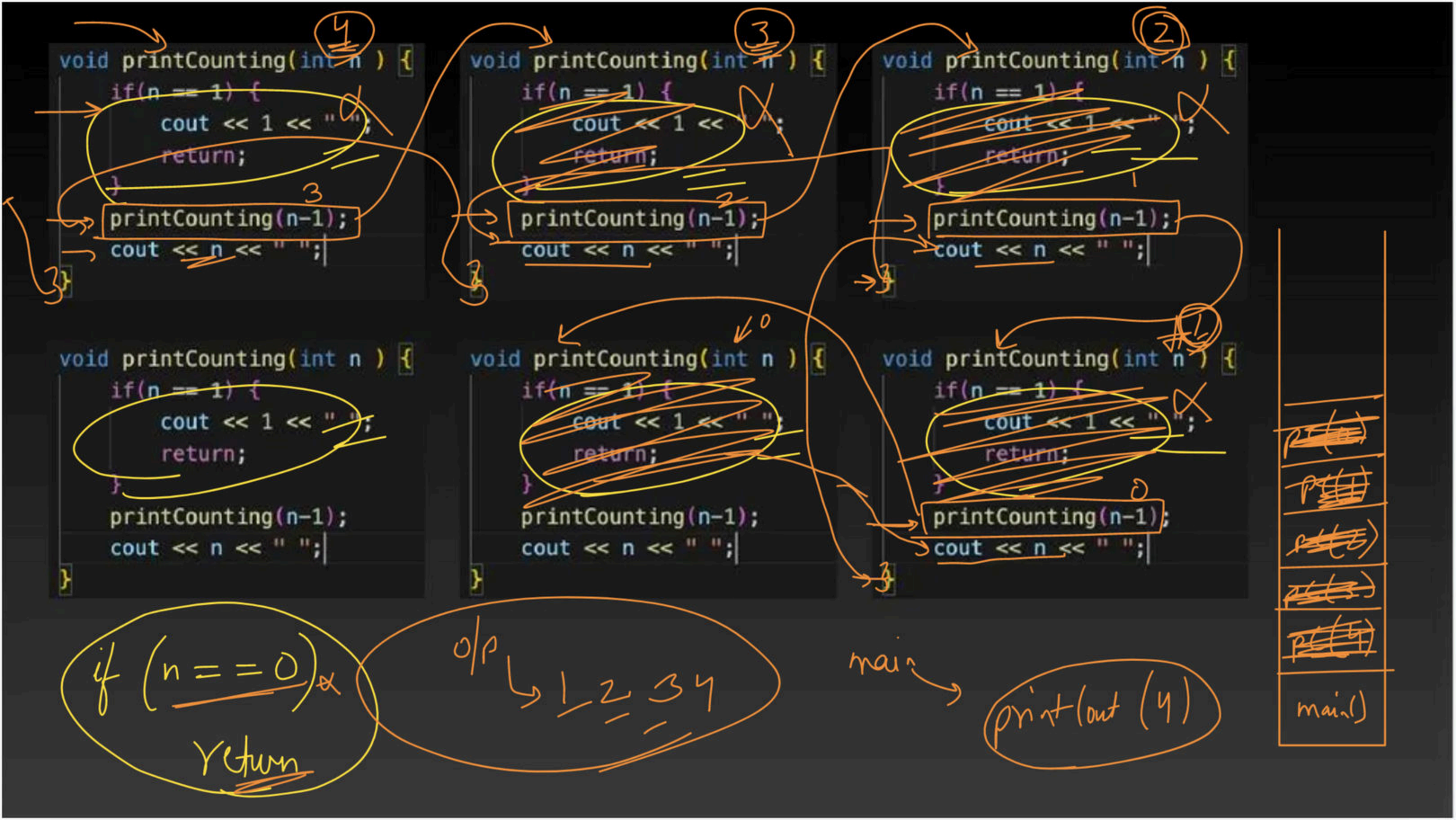

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```

```
void printCounting(int n ) {
    if(n == 1) {
        cout << 1 << " ";
        return;
    }
    printCounting(n-1);
    cout << n << " ";
}
```


void printCounting(int n) {
if(n == 1) {
cout << 1 << " ";
return;
}
printCounting(n-1);
cout << n << " ";
}
if (n==0)
return
o/p
1 2 3 4
main
print(out (4)
main()

$n = 5$

$2^5 = 2 \times 2 \times 2 \times 2 \times 2$ (2^4)

$2^5 = 2 \times 2^4$

pow(5) = 2 × pow(4)

pow(n) = 2 × pow(n−1)

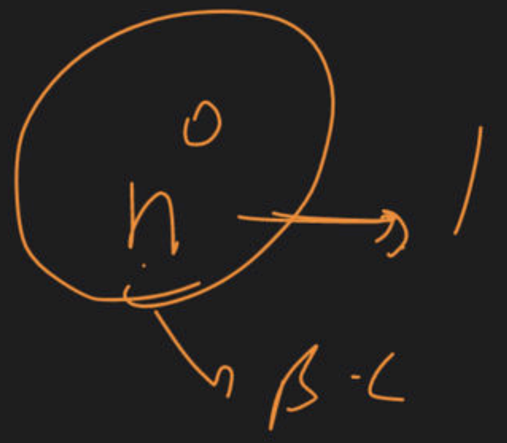
if(n == 0)

if(n == 1)

2^n

$2^1 \rightarrow 2$

pow (n) = 2 * pow(n-1)



pow(5)

pow(4)

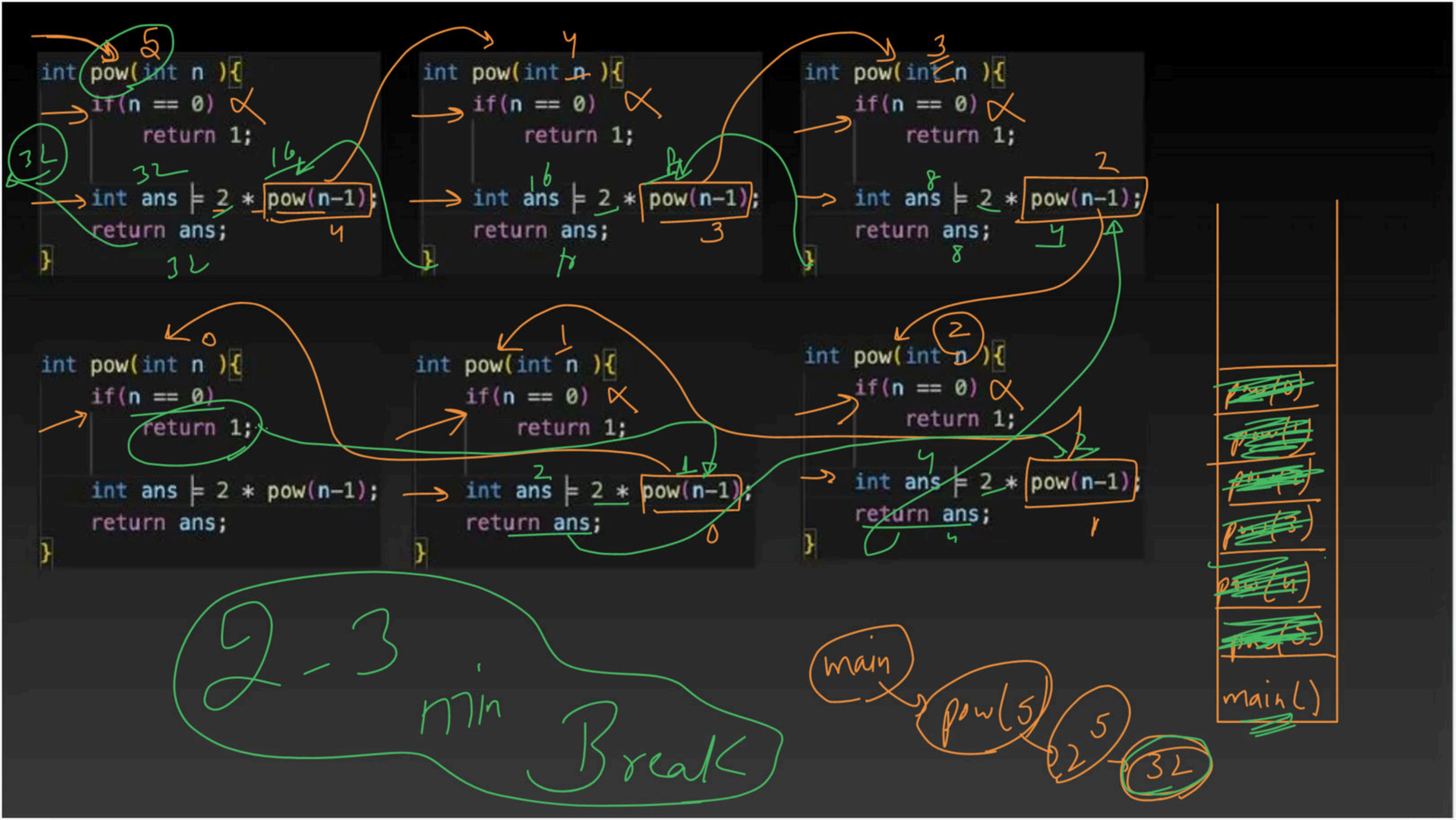
pow(3)

pow(2)

pow(1)

pow(0)

pow(-1)


int pow(int n){
if(n == 0)
return 1;
int ans = 2 * pow(n-1);
return ans;
}
2-3 min Break
main
pow(5)
5
32
main()

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```


```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```


```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

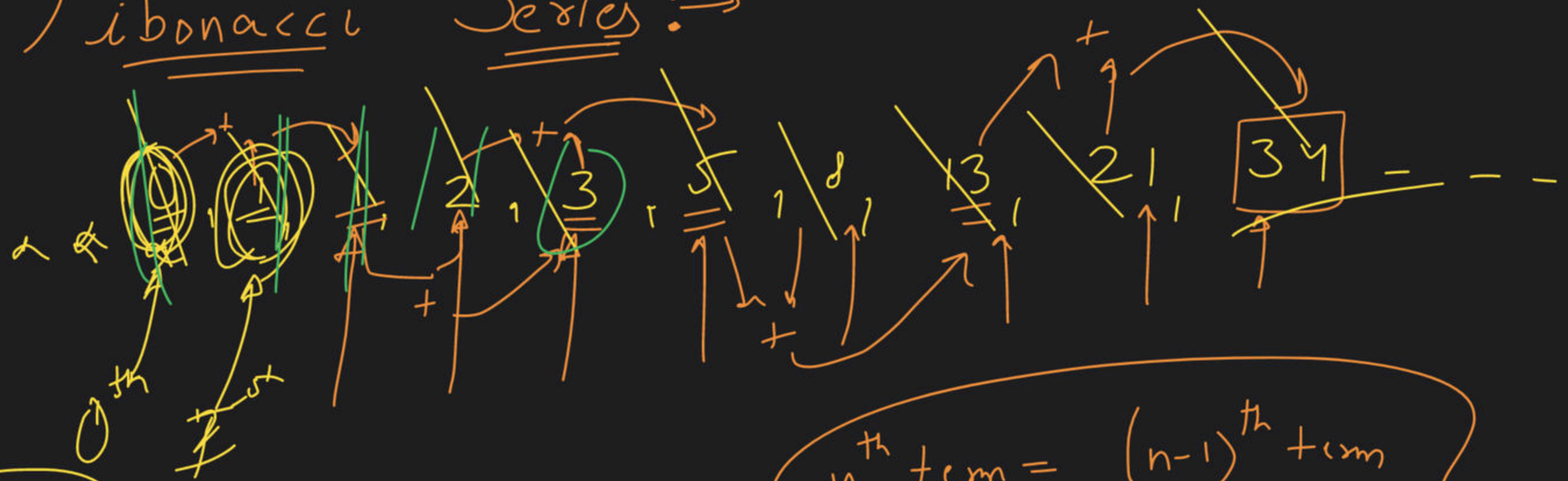
```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

```
int pow(int n ){
    if(n == 0)
        return 1;

    int ans = 2 * pow(n-1);
    return ans;
}
```

→ Fibonacci Series :→



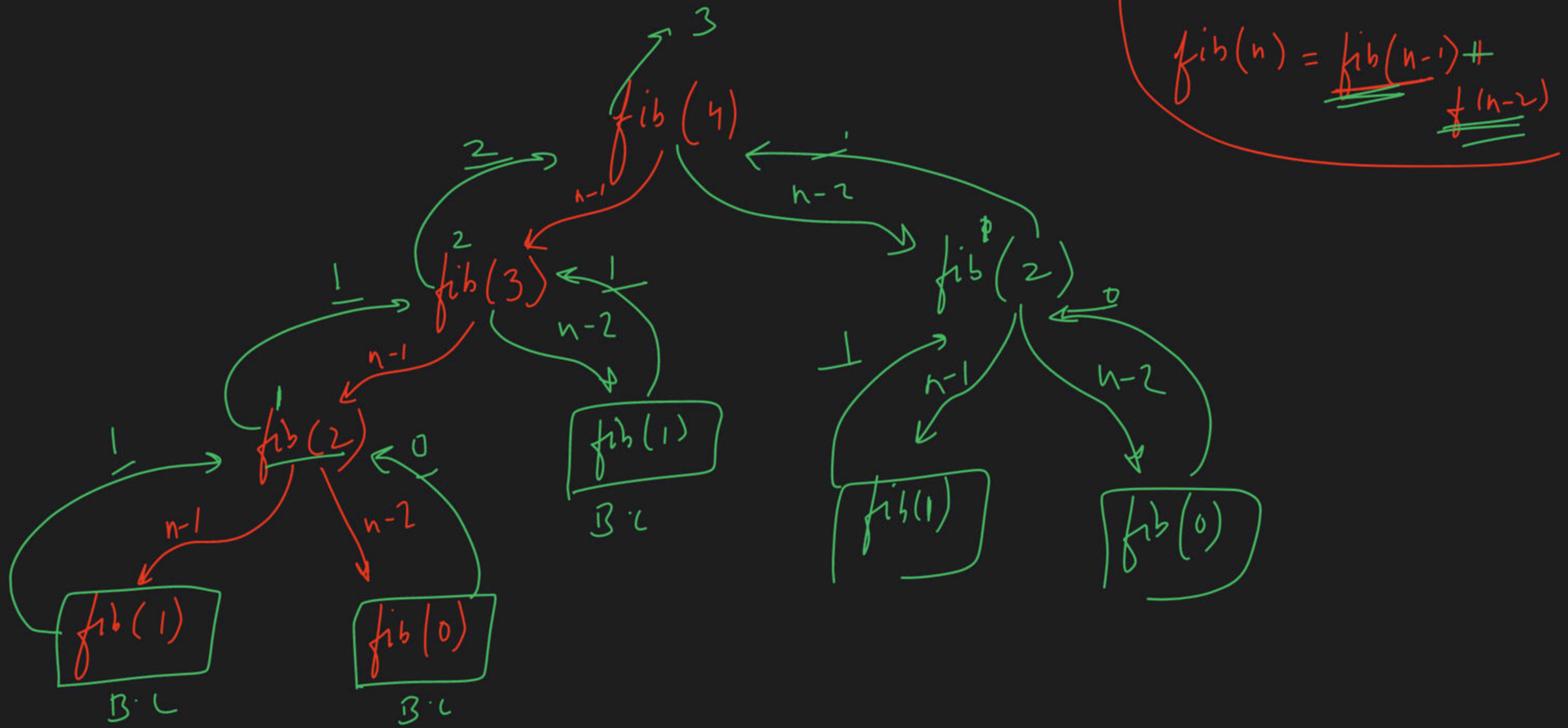
3. (

n == 0
↳ 0th term → 0

n == 1
↳ 1st term → 1

n^{th} term = $(n-1)^{th}$ term + $(n-2)^{th}$ term

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


fib(n) = fib(n-1) + fib(n-2)
fib(4)
3
2
1
n-1
n-2
fib(3)
2
1
1
n-1
n-2
fib(2)
1
0
1
n-1
n-2
fib(1)
B.C
fib(1)
B.C
fib(2)
1
0
n-1
n-2
fib(1)
fib(0)
B.C
B.C

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

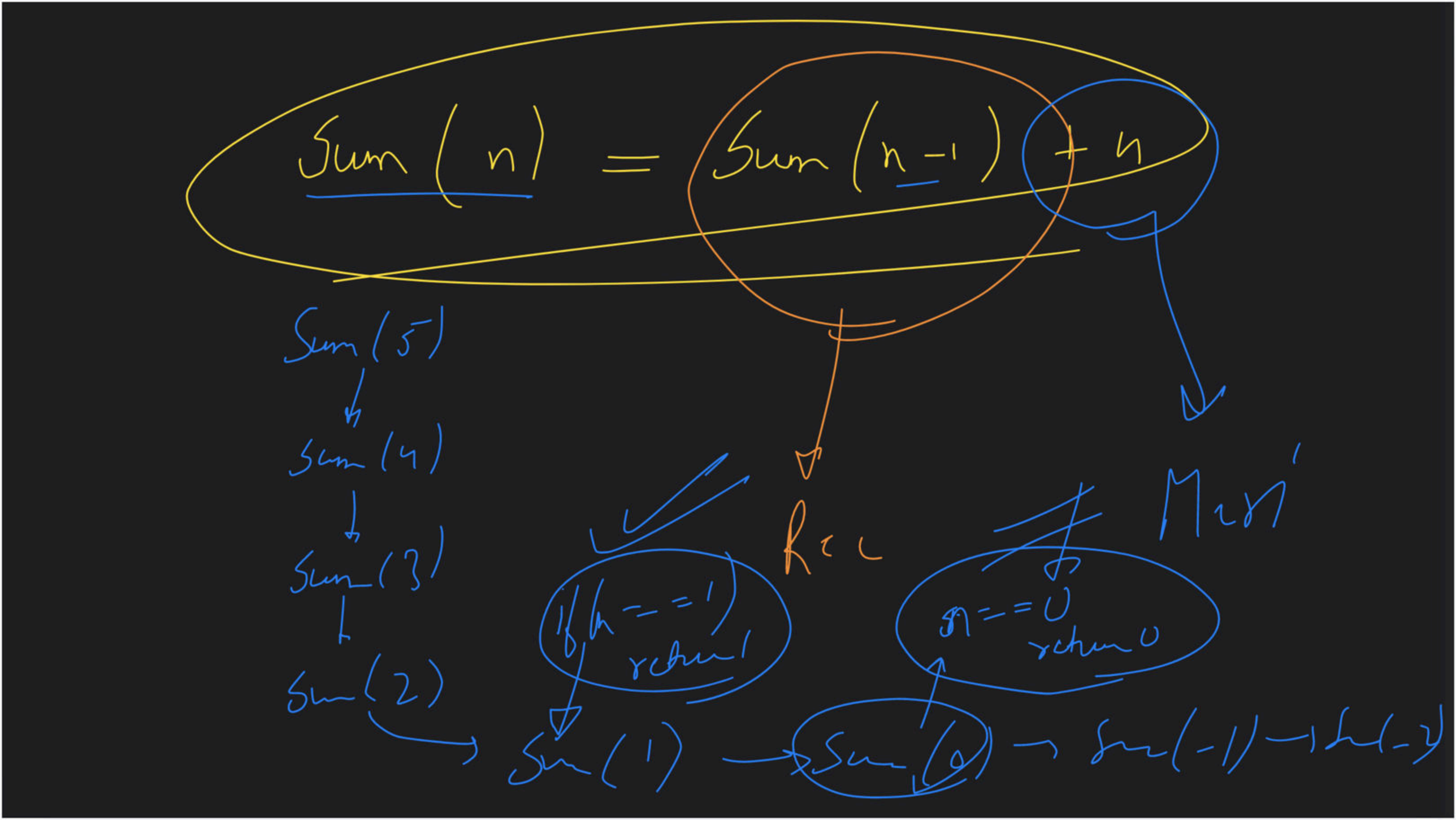
93%

getSum(5) → 1 + 2 + 3 + 4 + 5

getSum(4)

getSum(5) = getSum(4) + 5

getSum(n) = getSum(n-1) + n

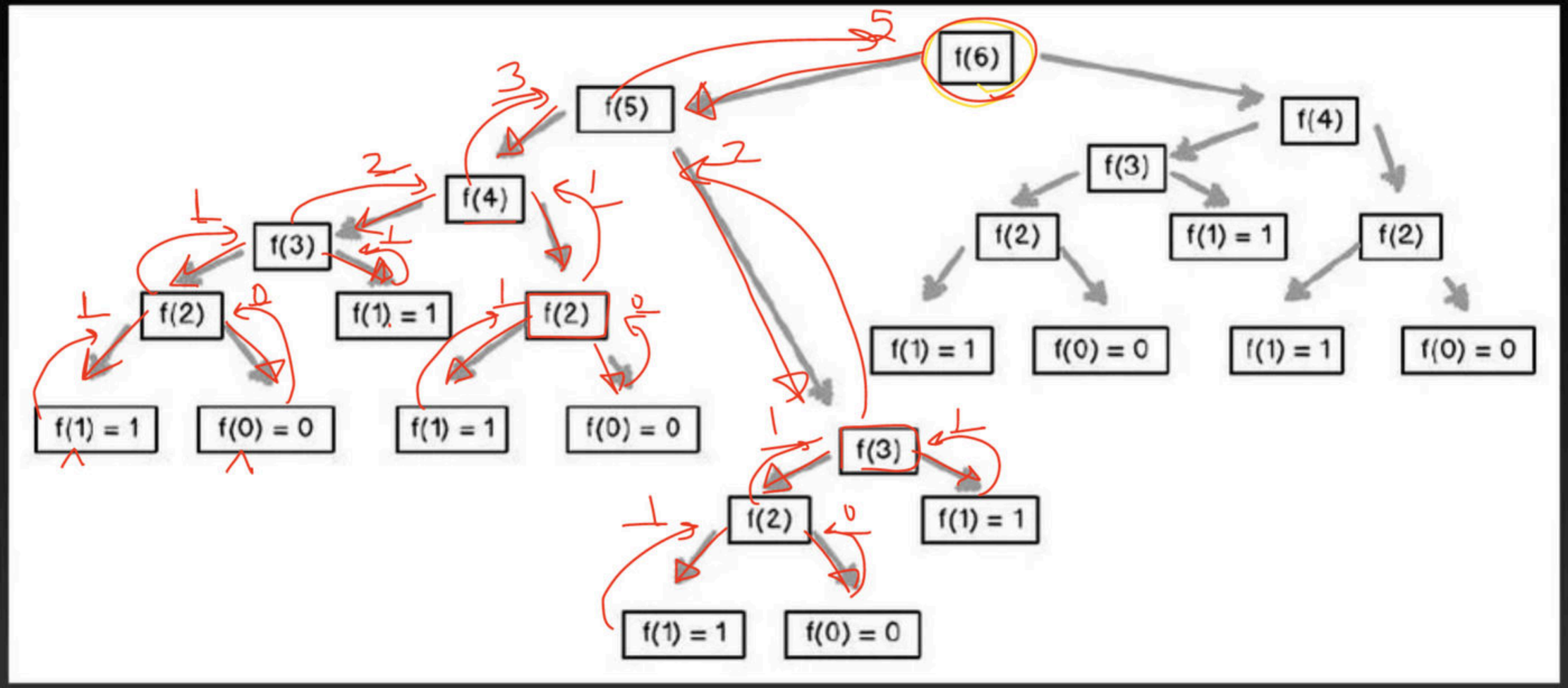

Sum(n) = Sum(n-1) + n
Sum(5)
Sum(4)
Sum(3)
Sum(2)
Sum(1)
Sum(0)
Rec
Main
if(n==1)
return 1
if(n==0)
return 0

n → 6

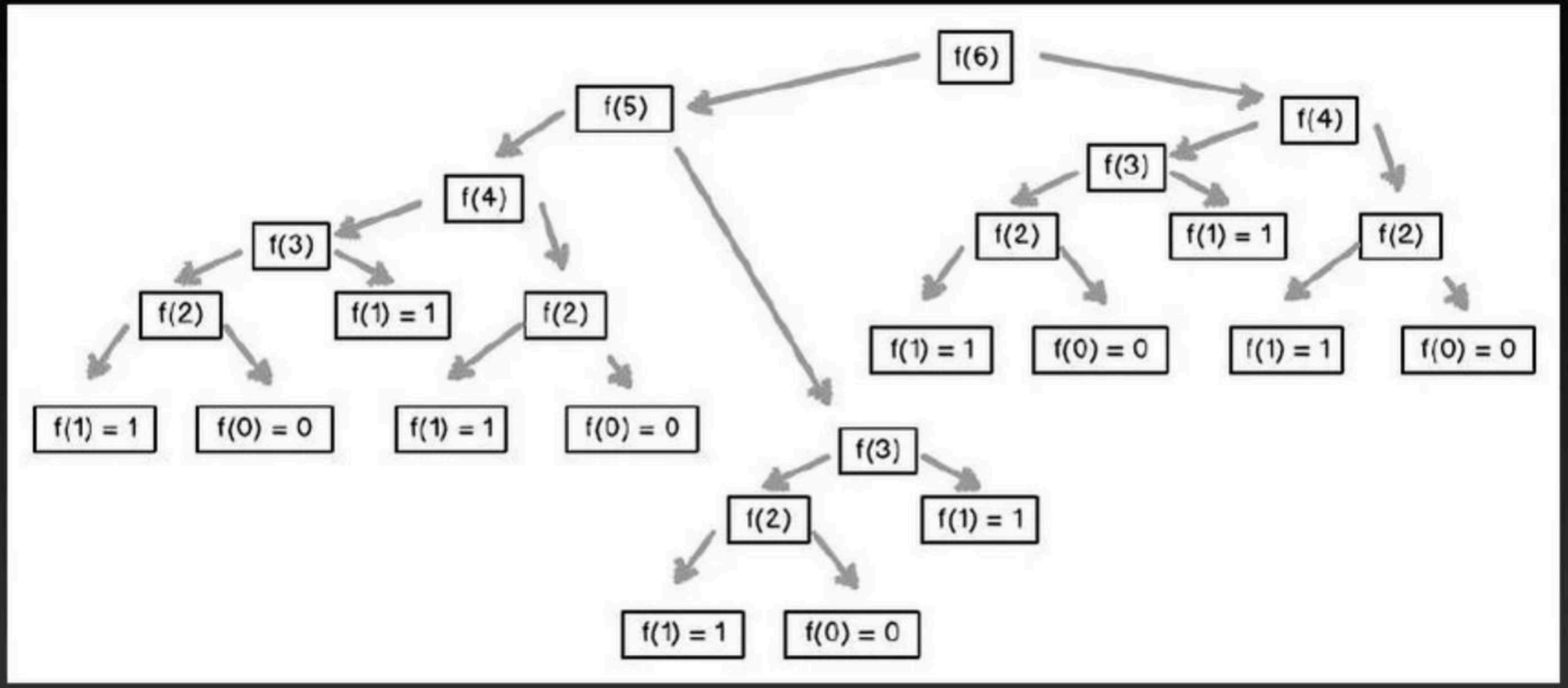
→ if (n == 0 || n == 1) return n

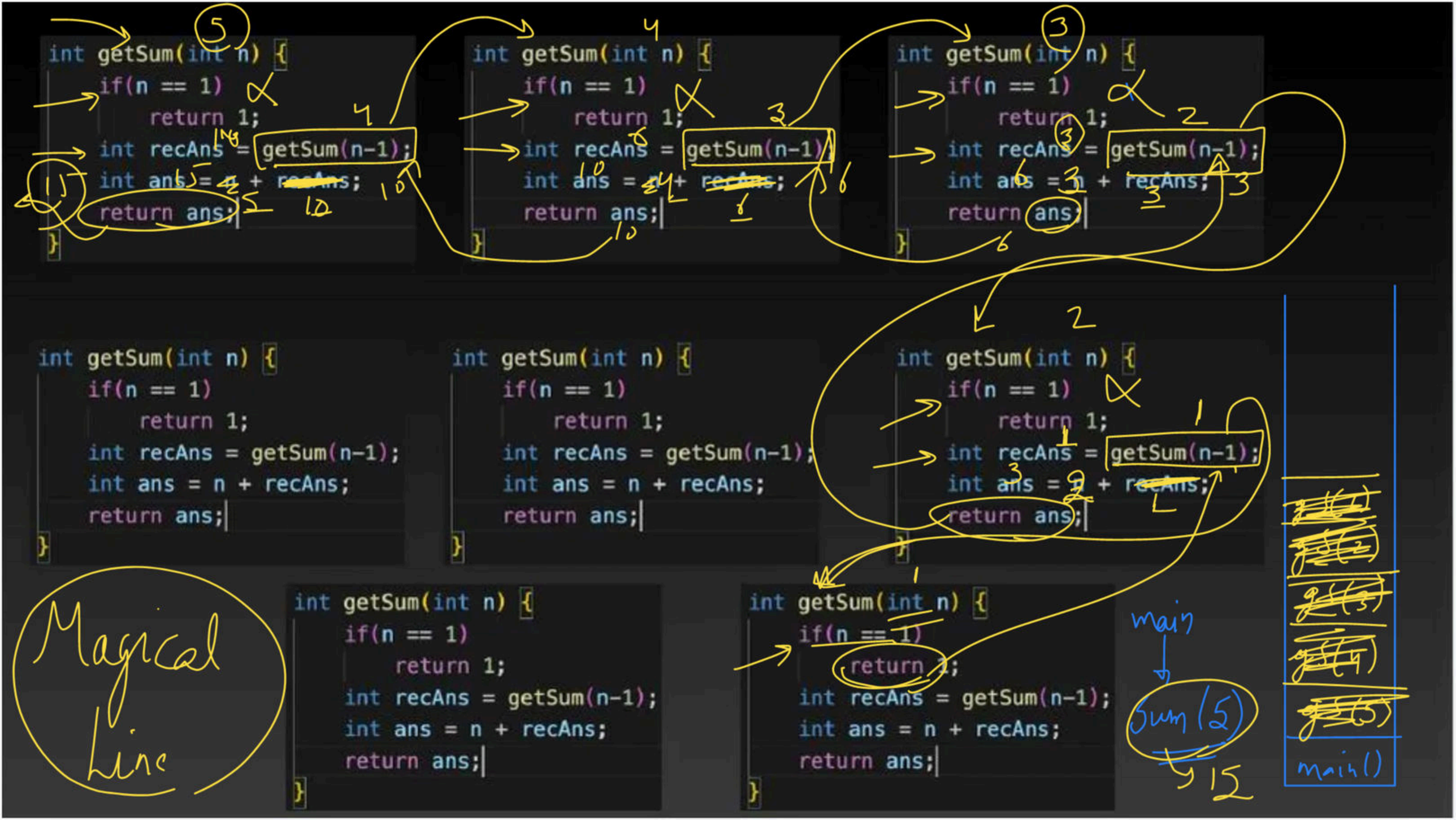
→ int ans = fib(n-1) + fib(n-2)

→ return ans



fib(0) = 0
fib(1) = 1
} → B.C


f(6)
f(5)
f(4)
f(4)
f(3)
f(3)
f(2)
f(1) = 1
f(2)
f(2)
f(1) = 1
f(0) = 0
f(1) = 1
f(0) = 0
f(2)
f(1) = 1
f(1) = 1
f(0) = 0
f(1) = 1
f(0) = 0
f(3)
f(2)
f(1) = 1
f(1) = 1
f(0) = 0


int getSum(int n) {
if(n == 1)
return 1;
int recAns = getSum(n-1);
int ans = n + recAns;
return ans;
}
Magical Line
main
Sum(5)
15
main()

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

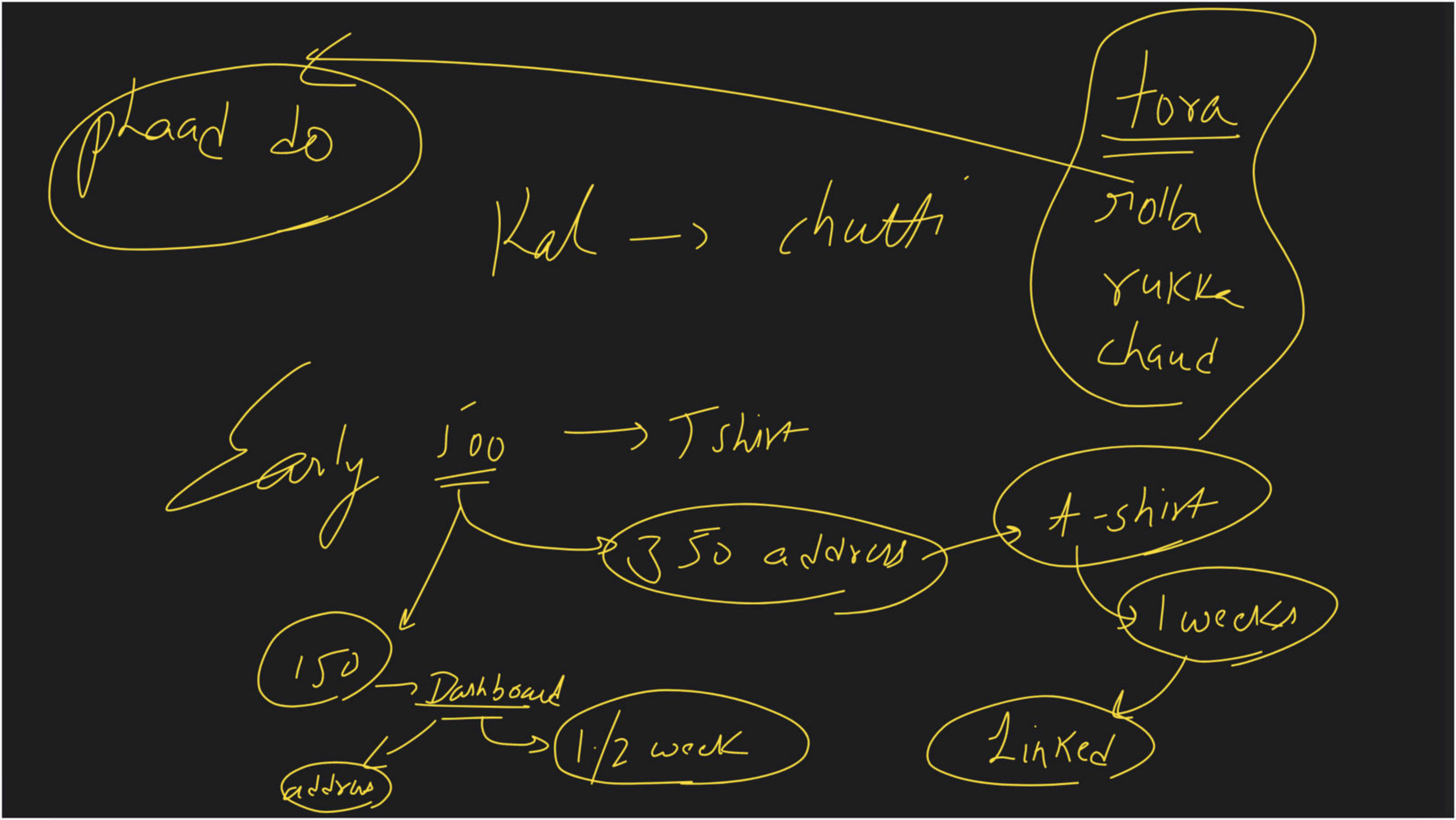
```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

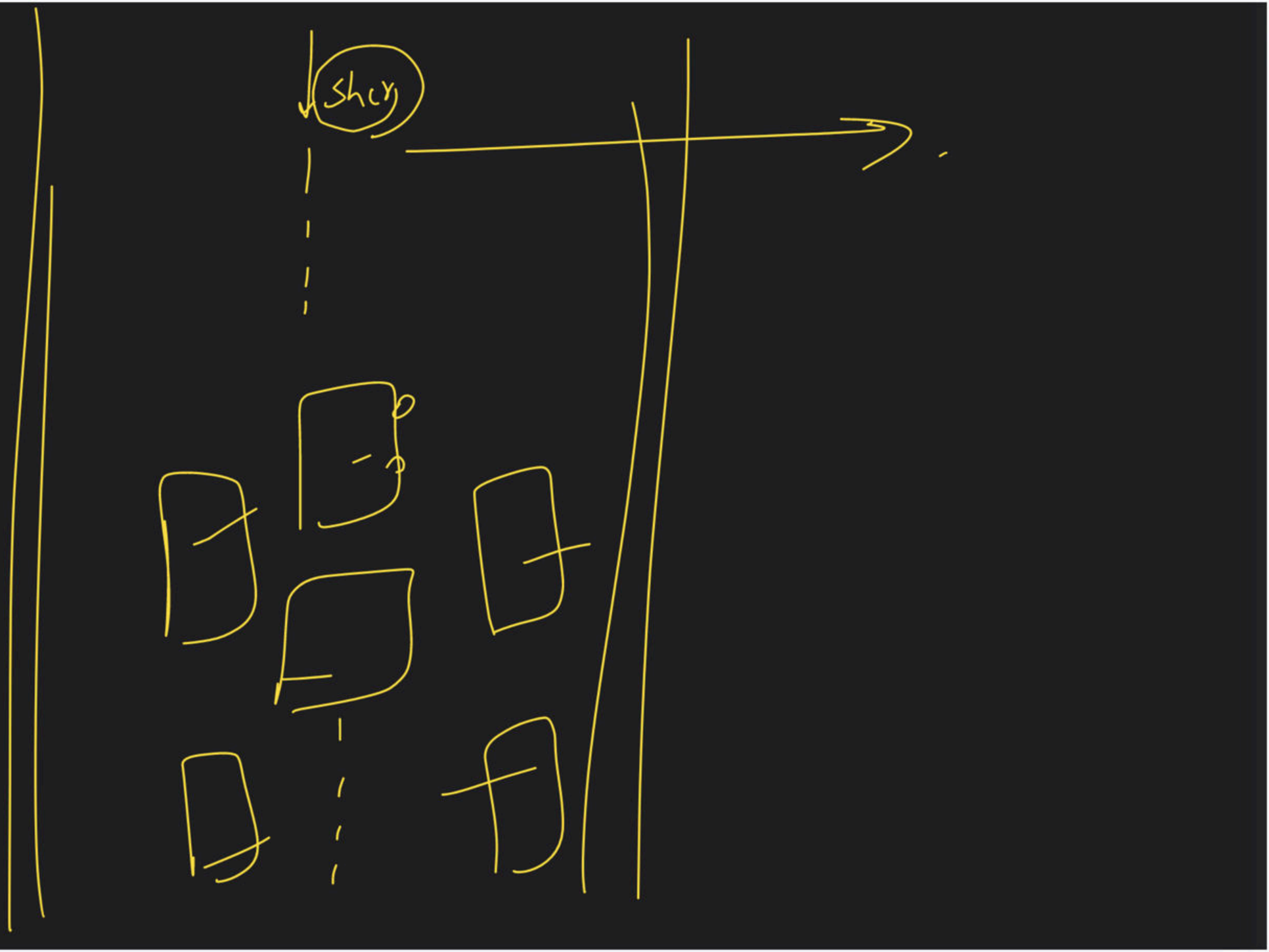
```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```

```
int getSum(int n) {
    if(n == 1)
        return 1;
    int recAns = getSum(n-1);
    int ans = n + recAns;
    return ans;
}
```


Plaad do
Kal → chutti
tora
solla
yukka
chaud
Early
100 → Tshirt
350 address
A-shirt
1 weeks
150
Dashboard
1/2 week
address
Linked


Shop