Array + Recur
+ Hashmap

DSA1

Loop

for, while ✓

40%

Array

String ✓

function ✓

Arraylist

1 hour

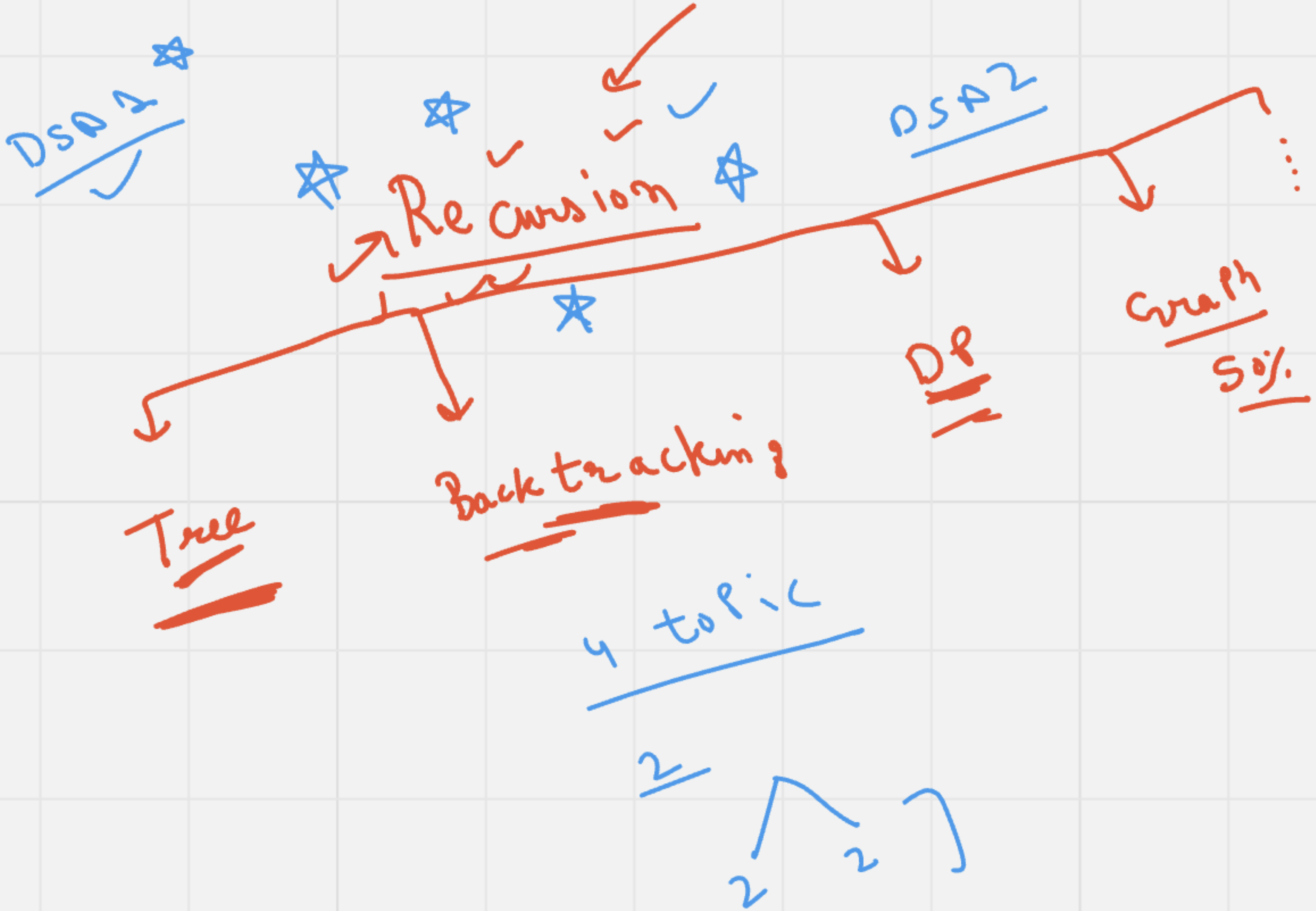
DSA 1 2 3 .. 4

front

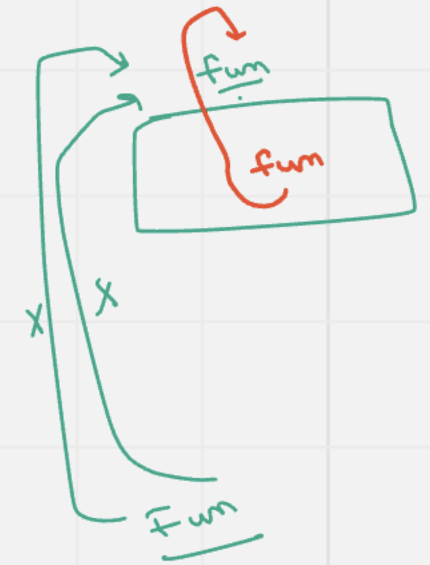
Recursion ✓

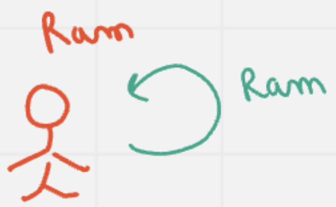
OOPS ✓

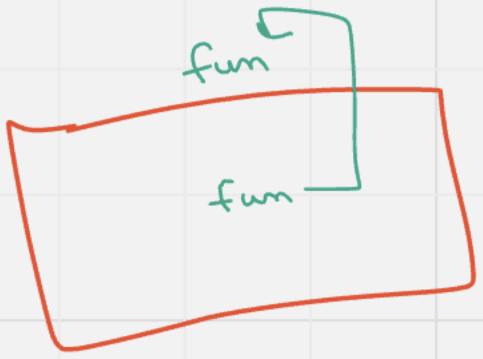
Linked list ✓

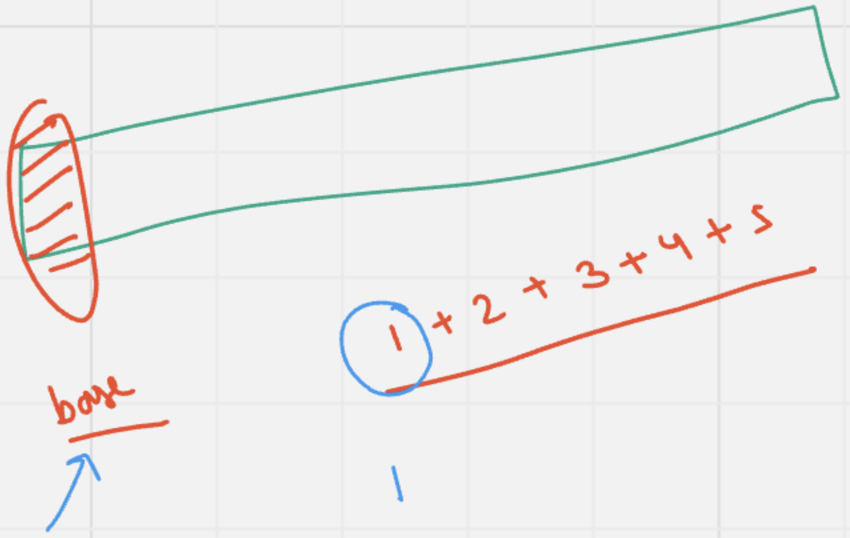

DSA 1
DSA 2
Recursion
Tree
Backtracking
DP
Graph
50%
4 topic
2
2
2

Recursion









$\textcircled{1} + 2 + 3 + 4 + 5$

1

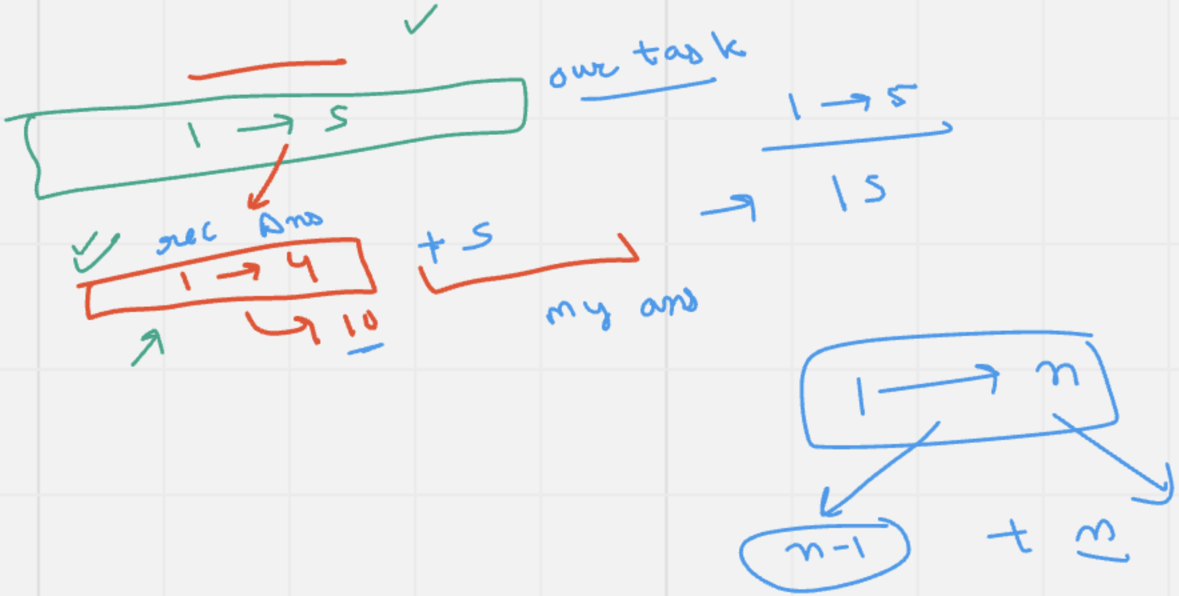
with coding

1. Base Case ✓
2. Rec Ans ✓ — my Ans ✓
3. my Ans ✓ — Rec Ans ✓

$$1 \longrightarrow 5$$

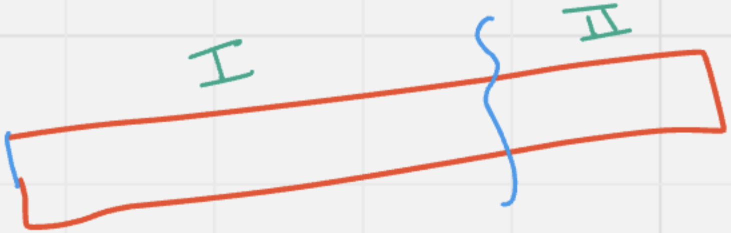
$$\Sigma 5 = \boxed{1 + 2 + 3 + 4} + 5$$

$$\Sigma 5 = \Sigma 4 + 5$$



When ? ?

Divide and con



$1 \longrightarrow 5$

$1 + 2 + 3 + 4 + 5 = 15$

$10 + 5 = 15$

$(1 + 2 + 3) + (4 + 5)$

$6 + 9 = 15$

✓

$\boxed{1 \longrightarrow 10}$

re $\boxed{1 \longrightarrow 9}$ $\overset{\text{my}}{+ 10}$

$\boxed{1 \rightarrow 10}$

$1 * 2 * 3 * 4 * 5$

$\prod \rightarrow 1 \longrightarrow 5$

re $\boxed{1 \longrightarrow 4}$ $\overset{\text{my}}{* 5}$] 120 ✓

24

1 * 2 * 3 * 4 * 5

$\boxed{1 \to n}$

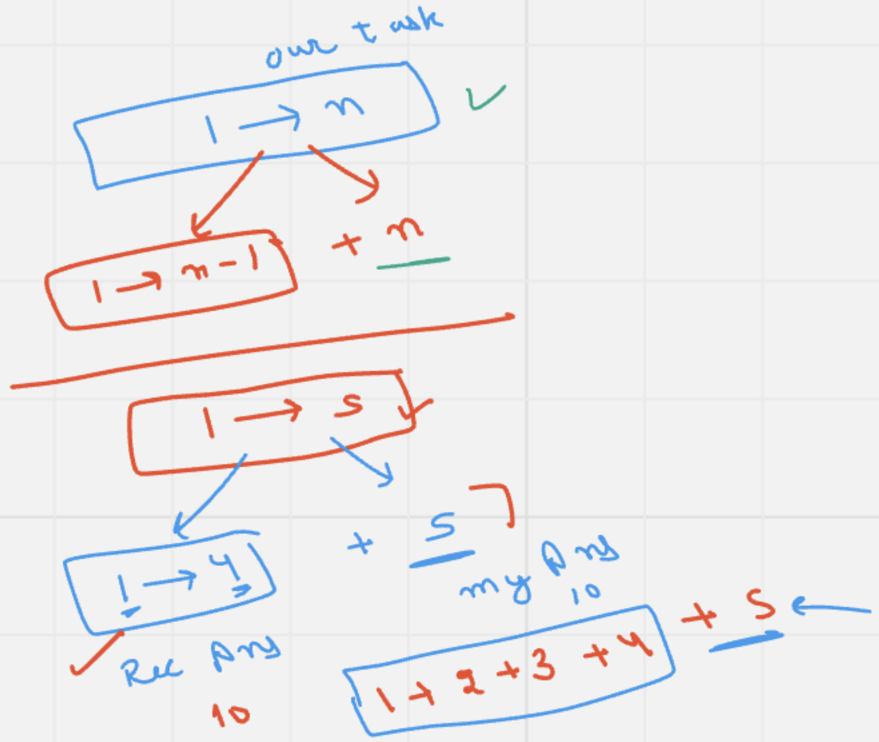
Break

Reason

Why ?!

How ?!

```
// 1 to n
    static int sumOf1ToN(int n) {
        // base case
        if(n == 1) {
            return 1;
        }

        // rec Ans
        int recAns = sumOf1ToN(n-1);

        // my Ans
        int myAns = recAns + n;
        return myAns;

    }
```



Overflow

O(1)

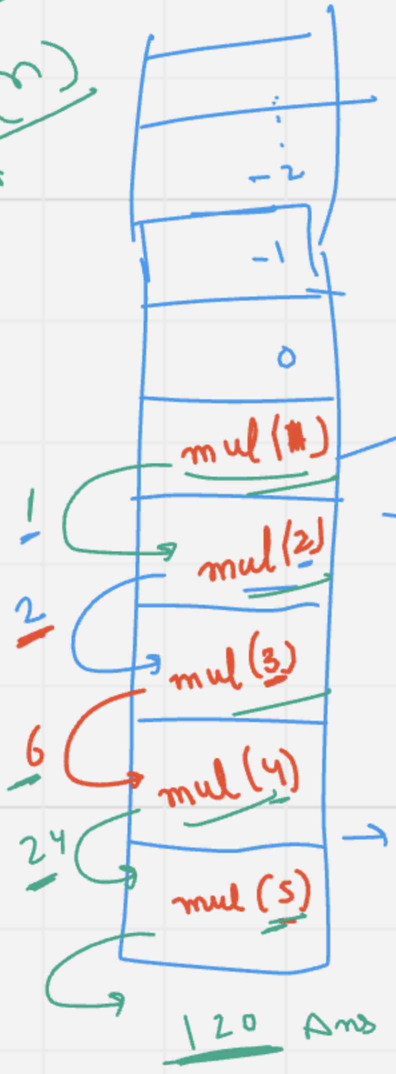
space Com

O(n)

5

```
static int multiply1ToN(int n) {

    if(n == 1) {
        return 1;
    }

    int recAns = multiply1ToN(n-1);

    int myAns = recAns * n;

    return myAns;

}
```

n = 1 ✓



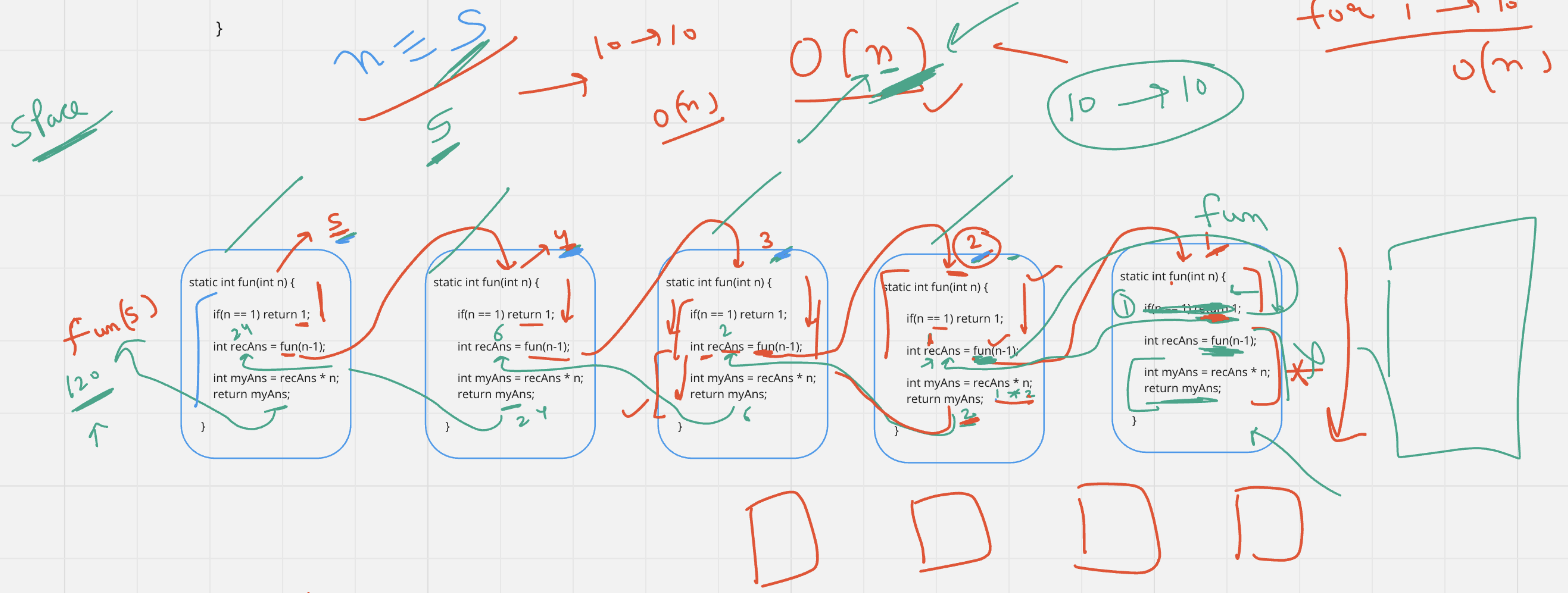
Stack overflow

Stack

fixed size

recursive stack

```
        int recAns = fun(n-1);

        int myAns = recAns * n;
        return myAns;

    }
```



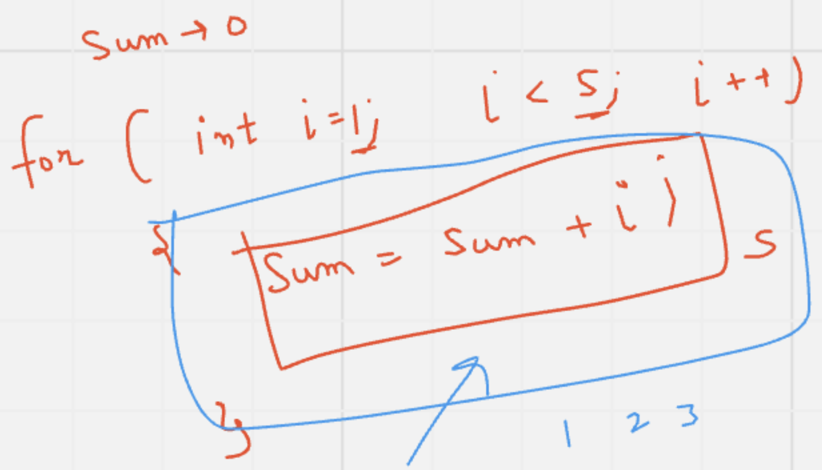
```
static int fun(int n) {

    if(n == 1) return 1;

    int recAns = fun(n-1);

    int myAns = recAns * n;
    return myAns;
}
```

```
static int fun(int n) {

    if(n == 1) return 1;

    int recAns = fun(n-1);

    int myAns = recAns * n;
    return myAns;
}
```

```
static int fun(int n) {

    if(n == 1) return 1;

    int recAns = fun(n-1);

    int myAns = recAns * n;
    return myAns;
}
```

```
static int fun(int n) {

    if(n == 1) return 1;

    int recAns = fun(n-1);

    int myAns = recAns * n;
    return myAns;
}
```

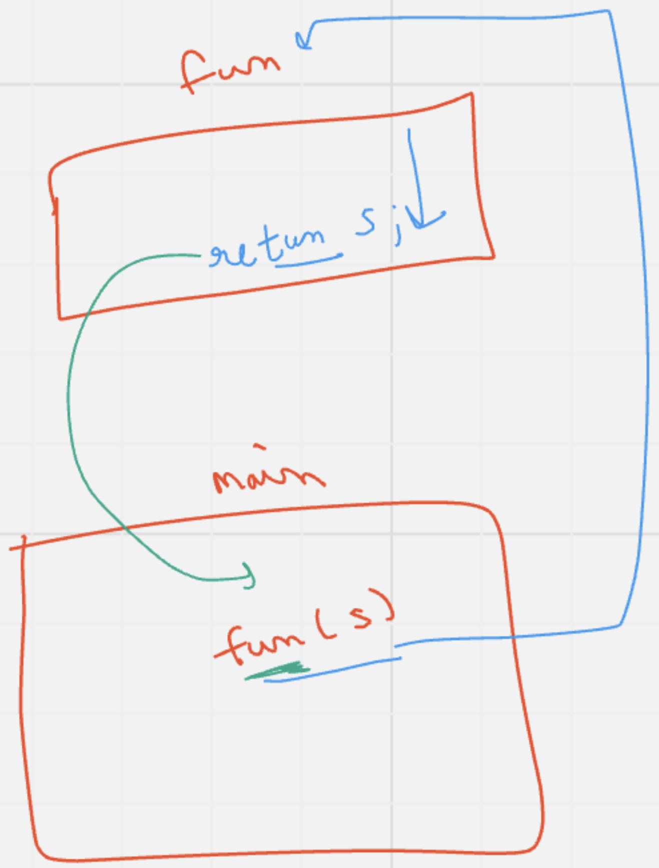
```
static int fun(int n) {

    if(n == 1) return 1;

    int recAns = fun(n-1);

    int myAns = recAns * n;
    return myAns;
}
```

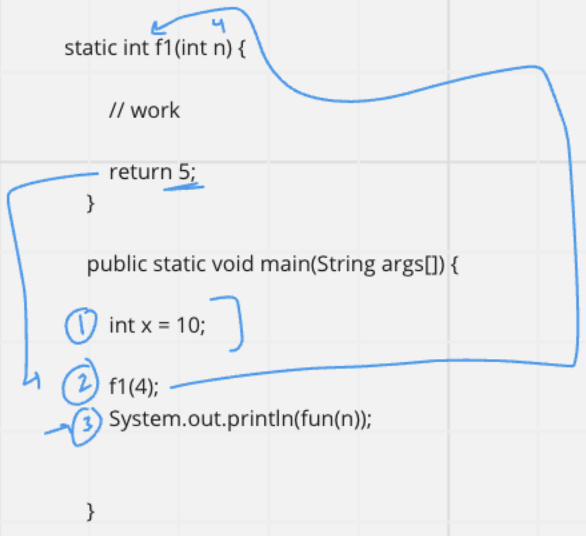


```
for ( int i=1;   i < 5;   i++)
{
   Sum = Sum + i;
}
```

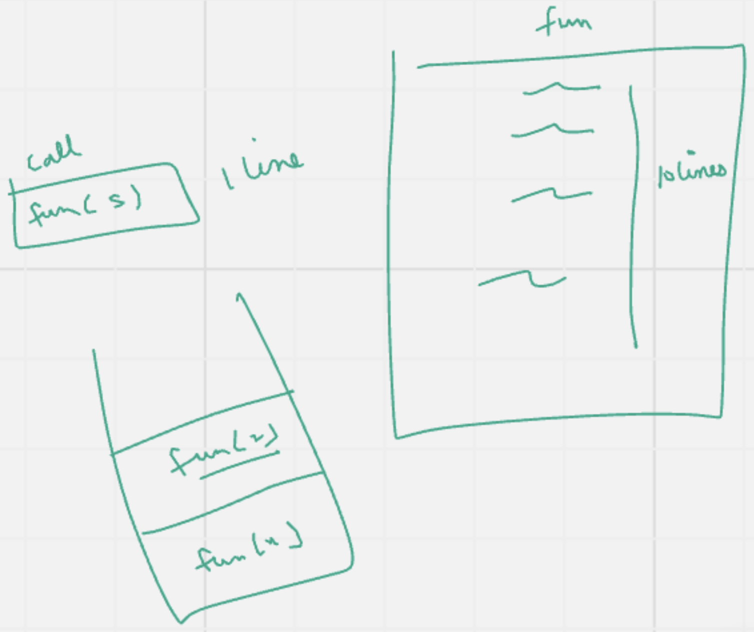
$1 \longrightarrow 5$

$1 \longrightarrow 5$

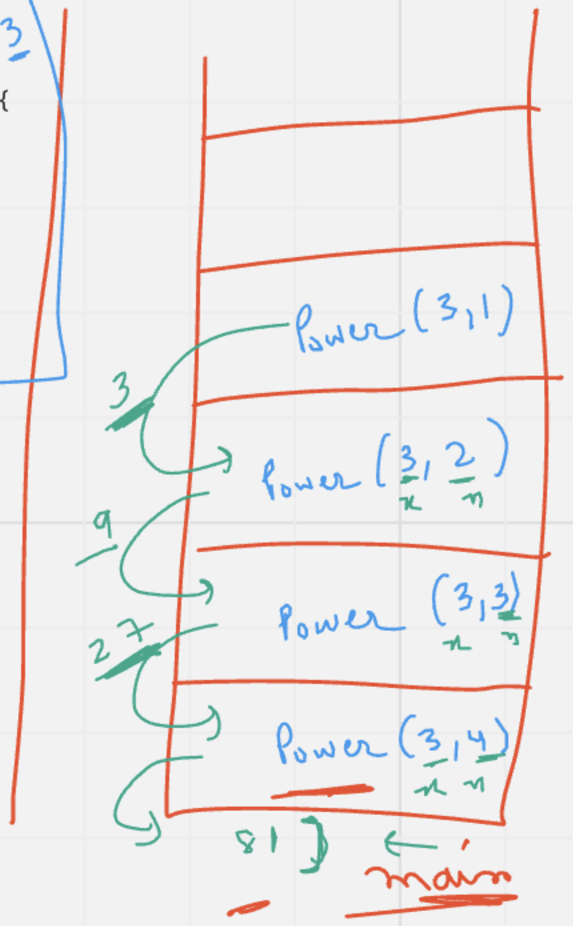




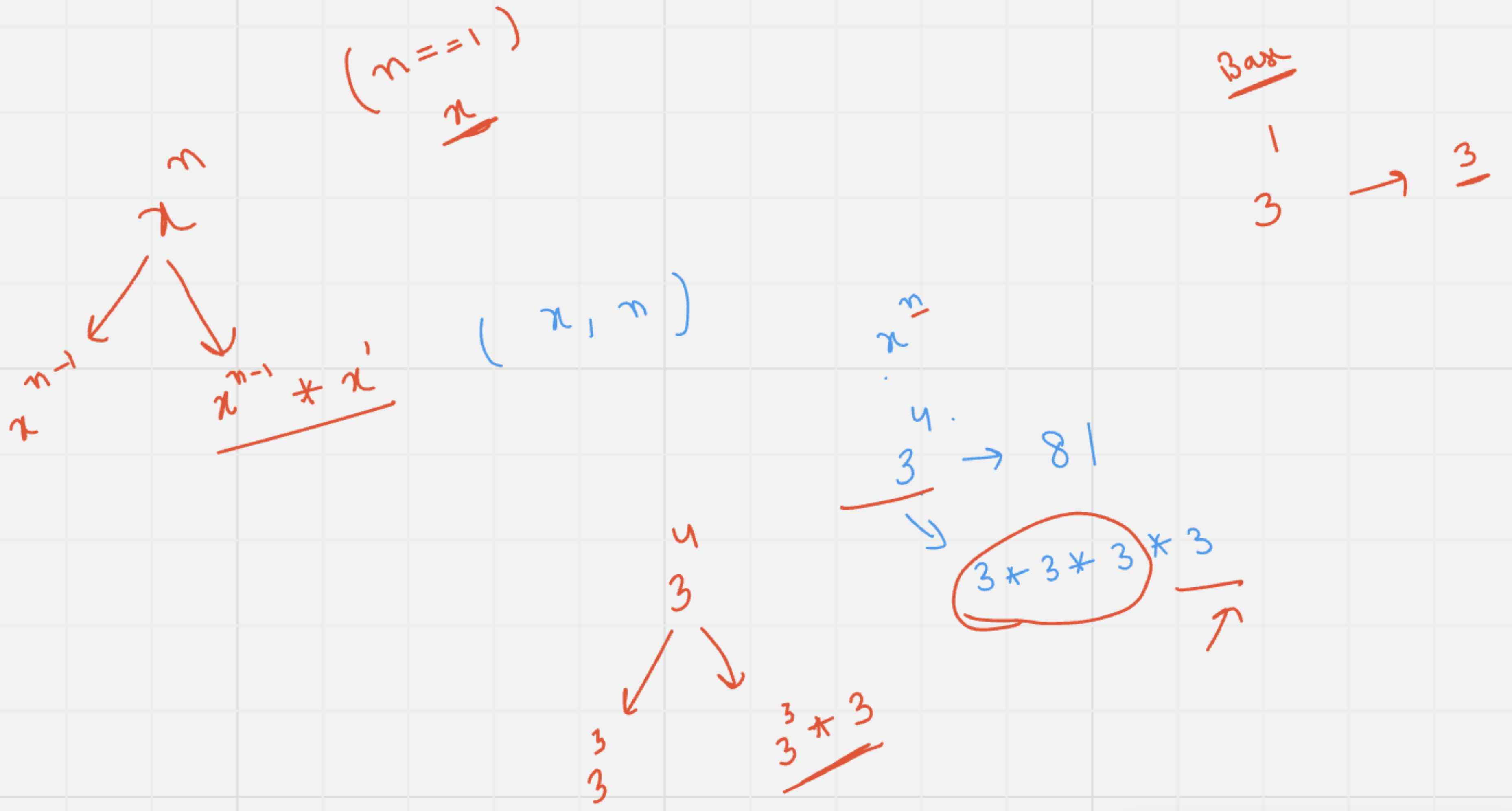
```
static int f1(int n) {

    // work

    return 5;
}

  public static void main(String args[]) {

    int x = 10;

    f1(4);
    System.out.println(fun(n));


  }
```



```
public static long xPowerN(int x, int n){
    //write code here

    if(n == 1) {
        return x;
    }

    long recAns = xPowerN(x, n-1);

    long myAns = recAns * x;

    return myAns;

}
```



Power (3,1)
3
Power (3,2)
9
Power (3,3)
27
Power (3,4)
81
main



(n==1)
x
x^n
x^{n-1}
x^{n-1} * x^1
(x, n)
Base
1
3 → 3
x^n
3^4 → 81
3 * 3 * 3 * 3
3^4
3^3
3^3 * 3