



$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

→ The mathematical representation of the recursive function is called as recurrence relation

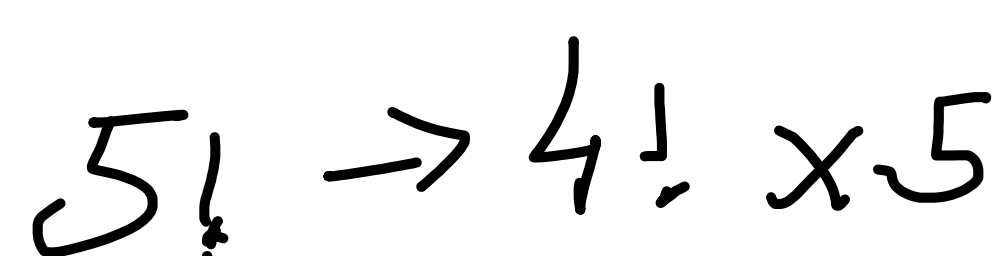
Find height of buildings \rightarrow heights

recursion 



Iterative

Heriberto → 5! 4!



$$\Rightarrow \boxed{n!} \Rightarrow (n-1)! \times n$$

$$\boxed{f(n) \Rightarrow f(n-1) \text{ u } n}$$

function a()

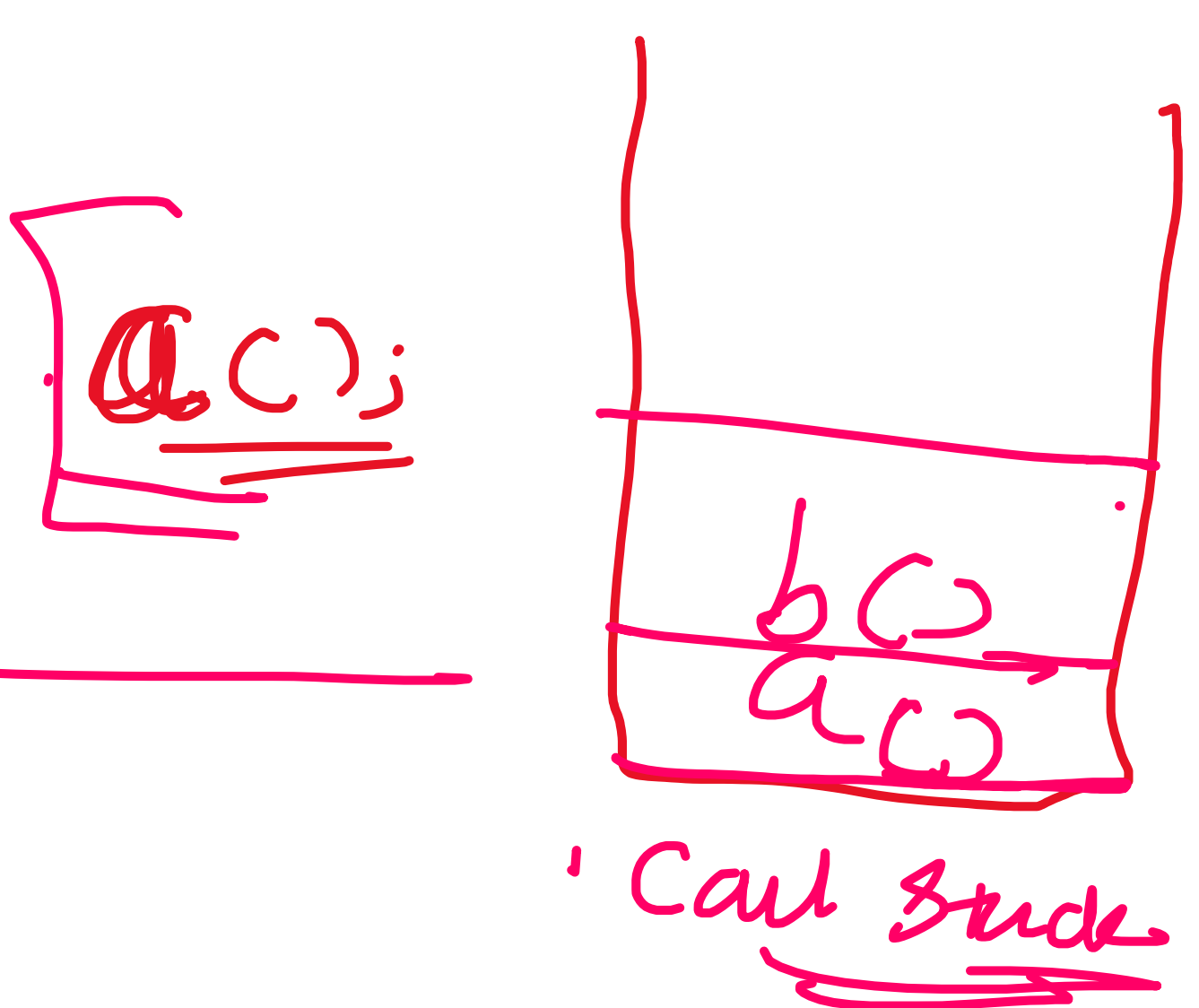
function $c(c)$

→ b c)

Cor. log (H⁴)₂

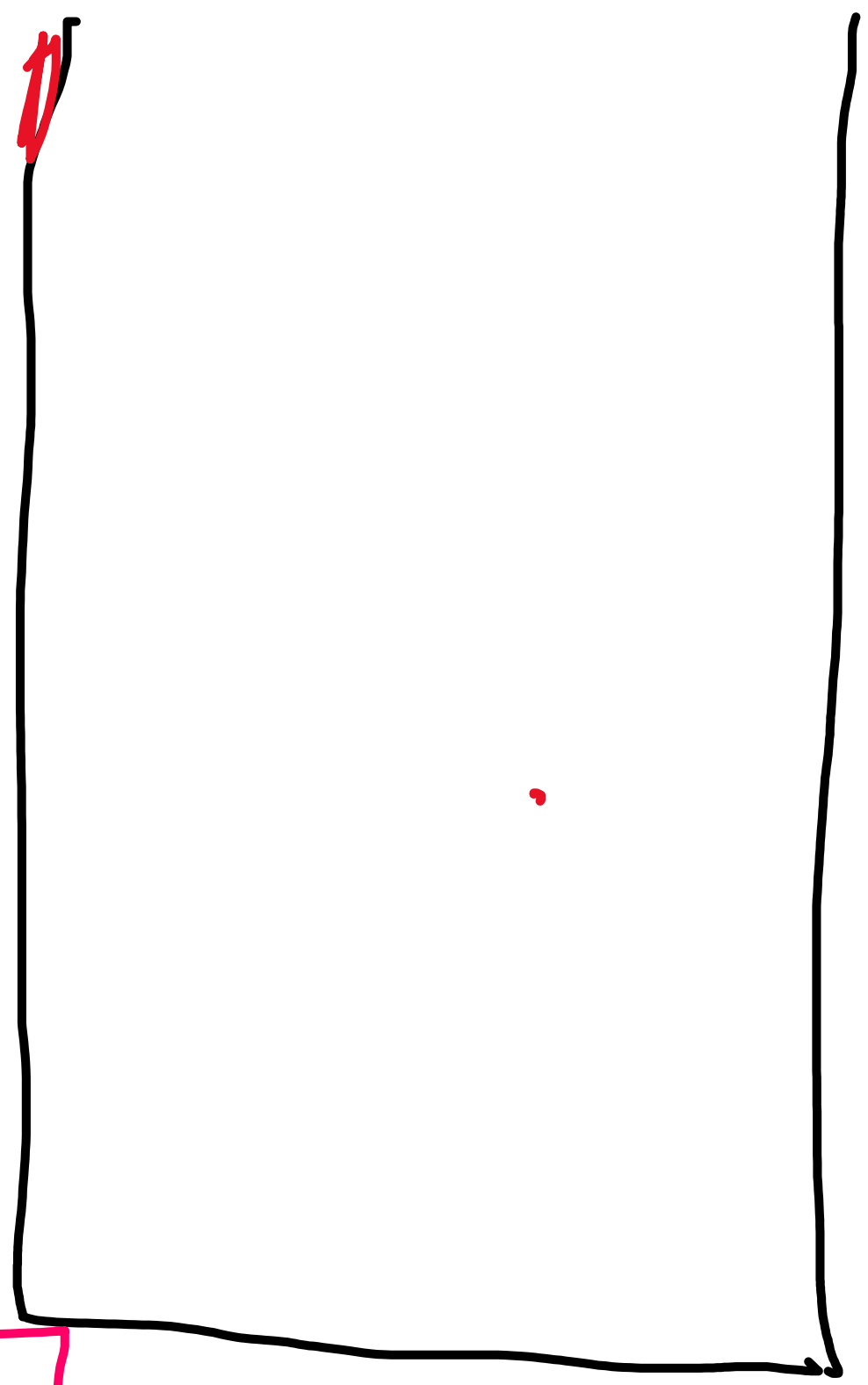
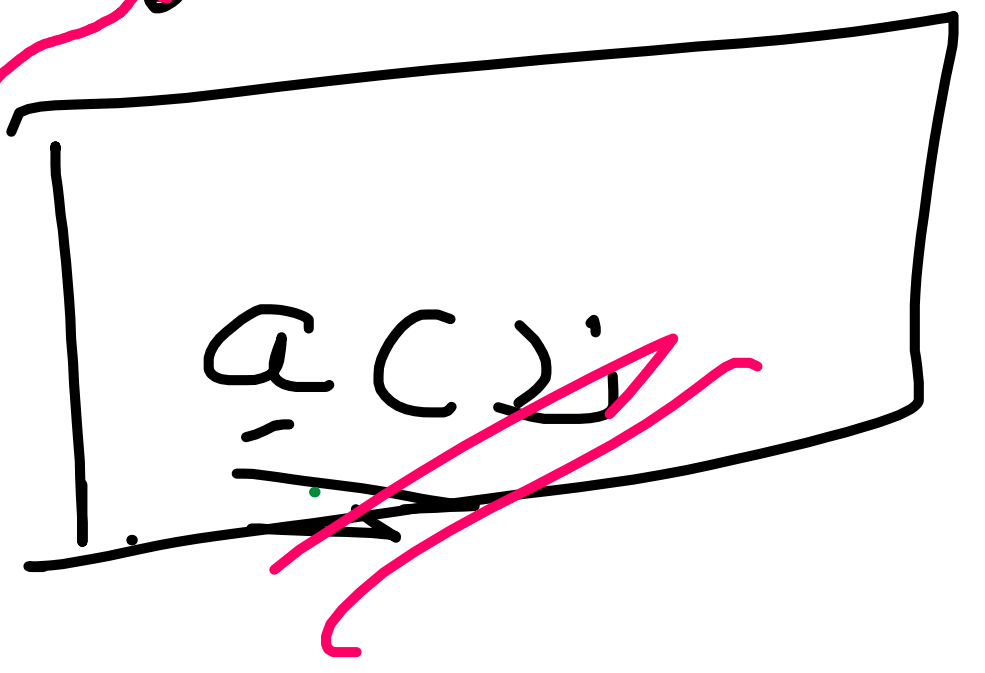
Consider. log C("a")

```
function b(c)
{
  c(c);
}
```

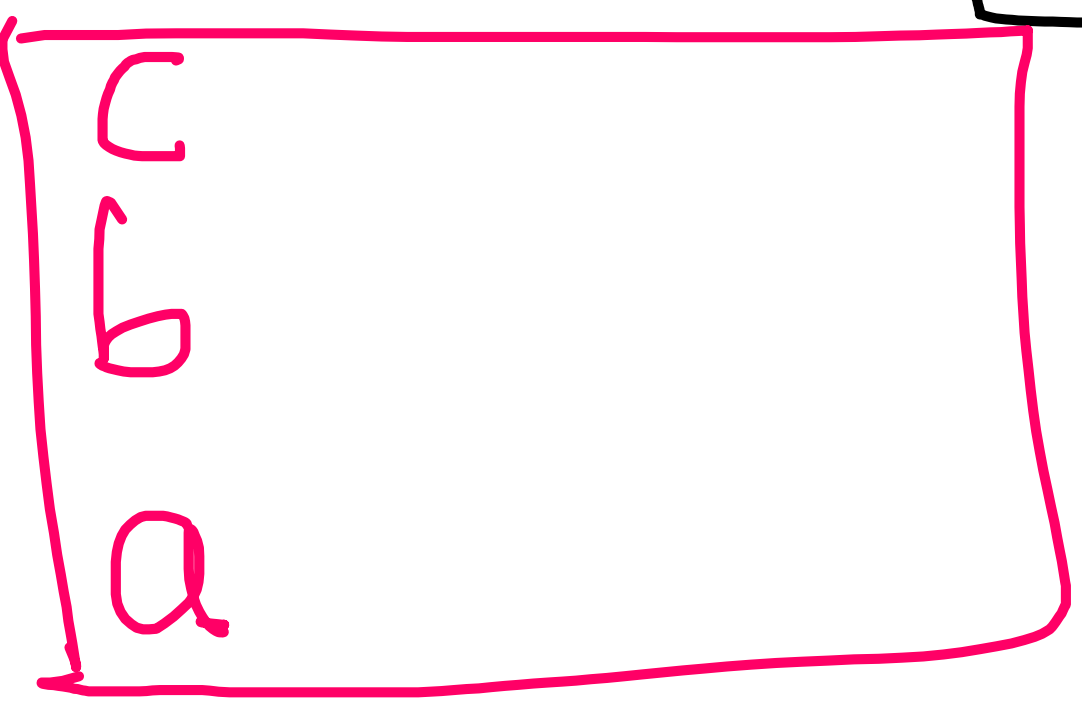


```
function a(c)
{
  b(c);
  c(c);
  c.l("b");
}
```

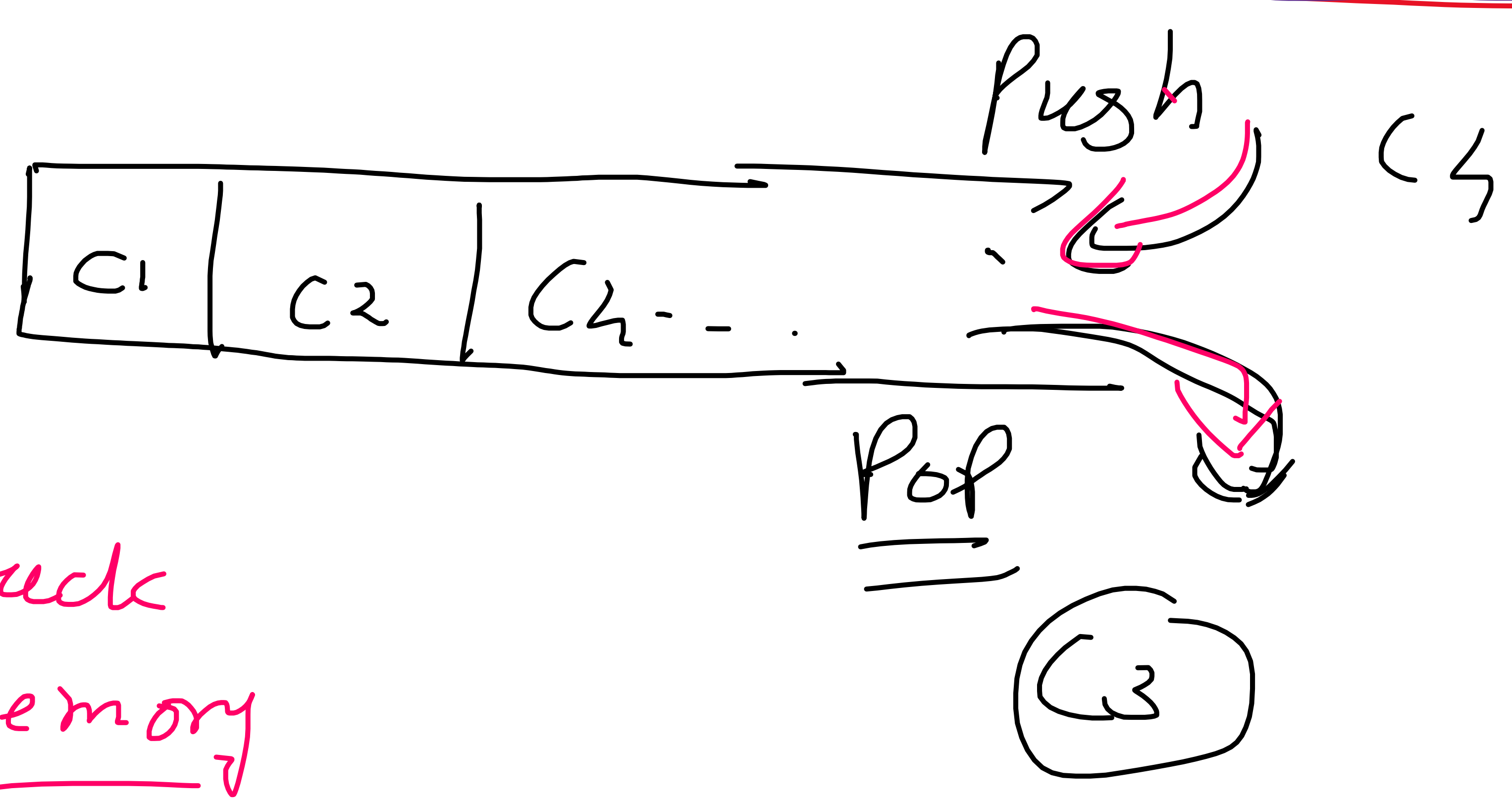
```
function c(c)
{
  c.l("c");
}
```



Variables
← fun



Call Stack.

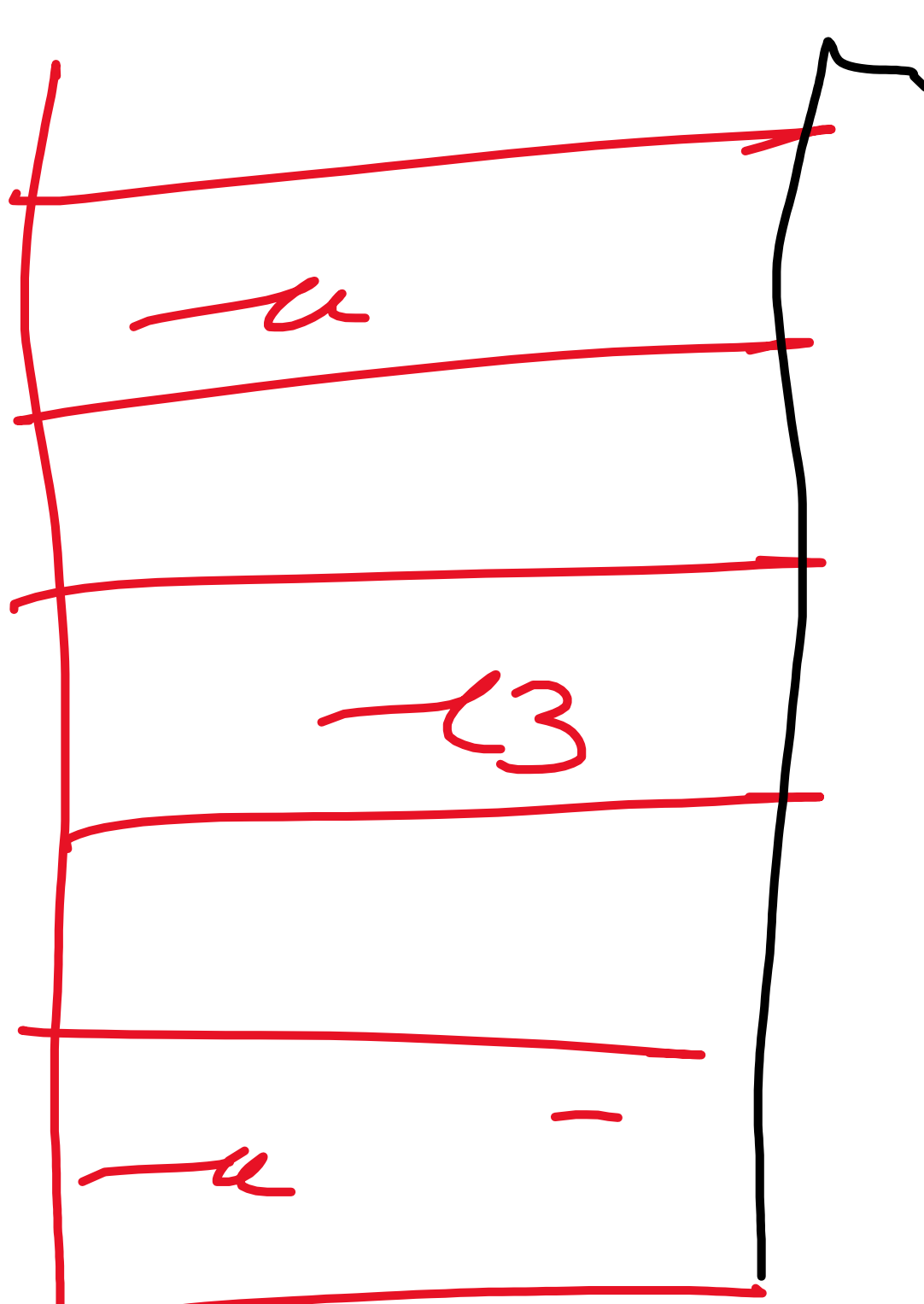


Stack
memory

Stack → variables & fn

Heap → objects

factorial (5)

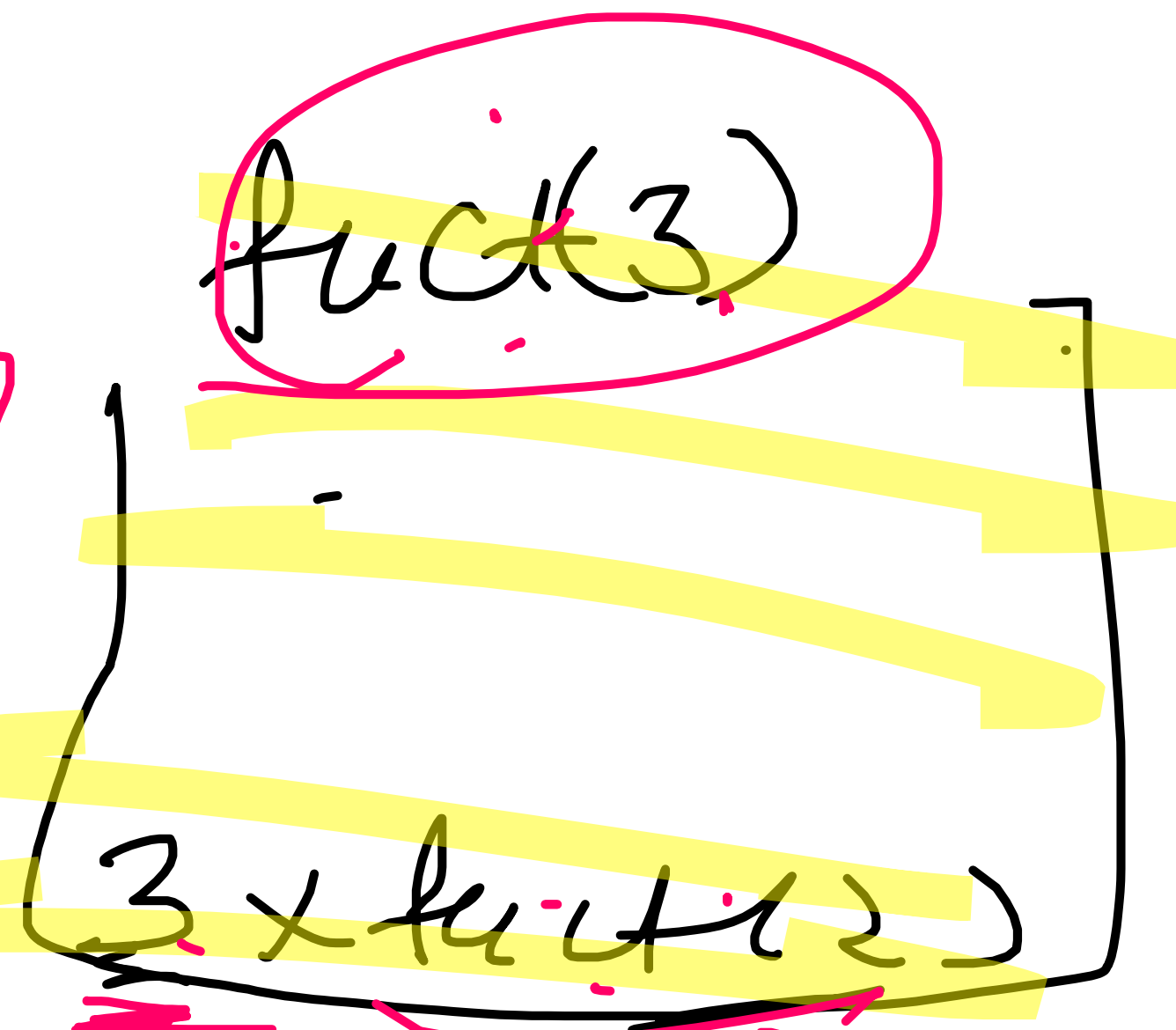
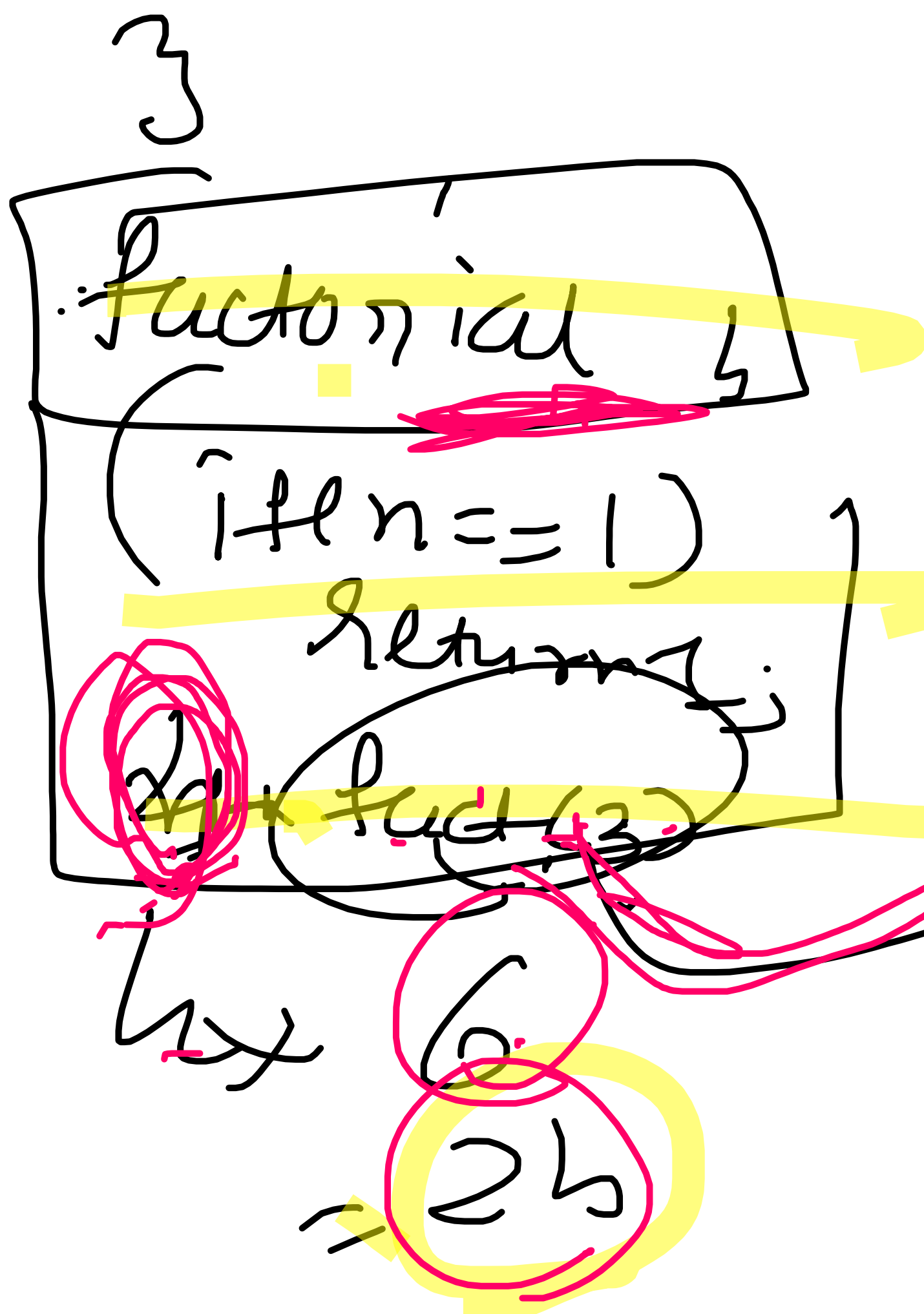
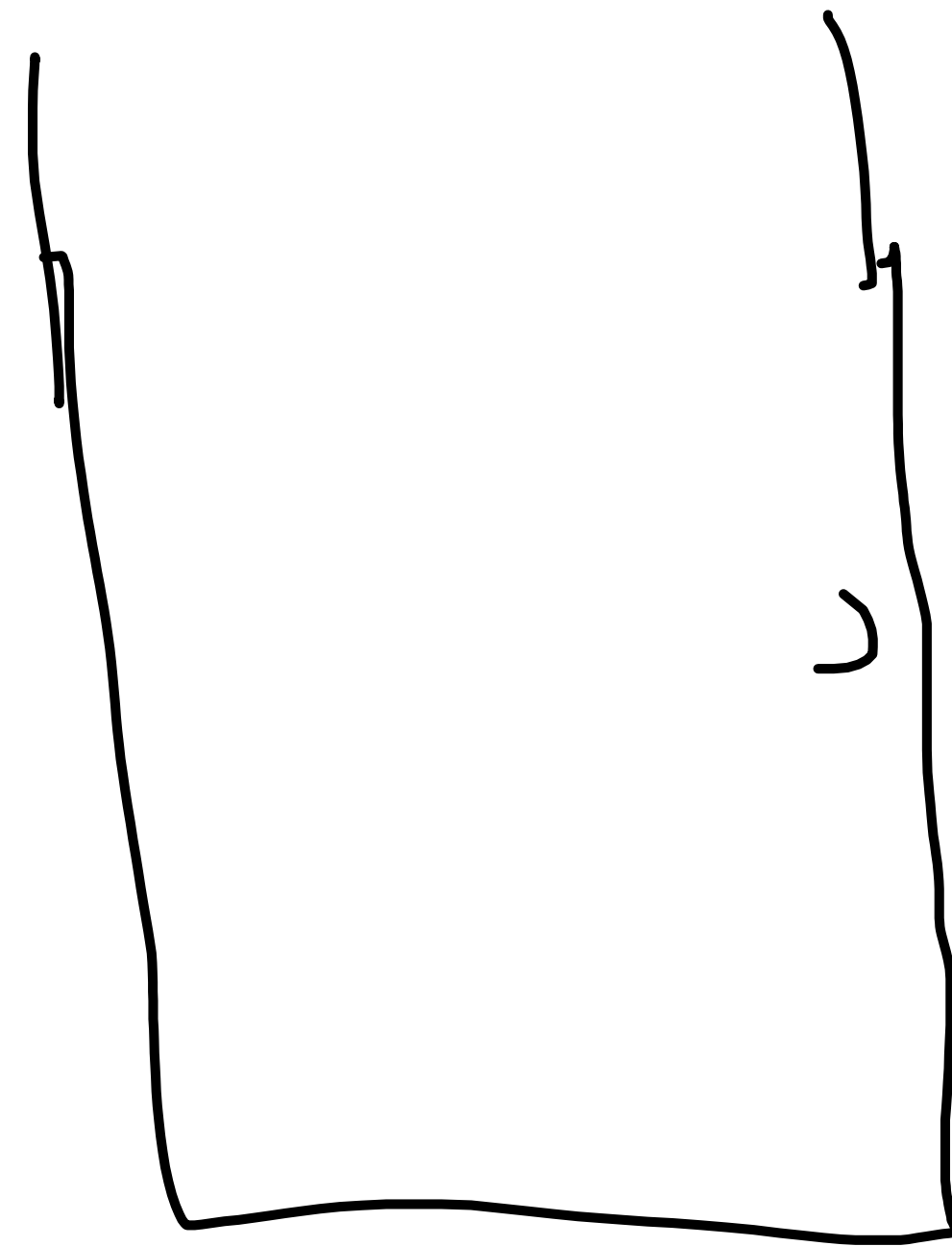


function one() {
 two()
}

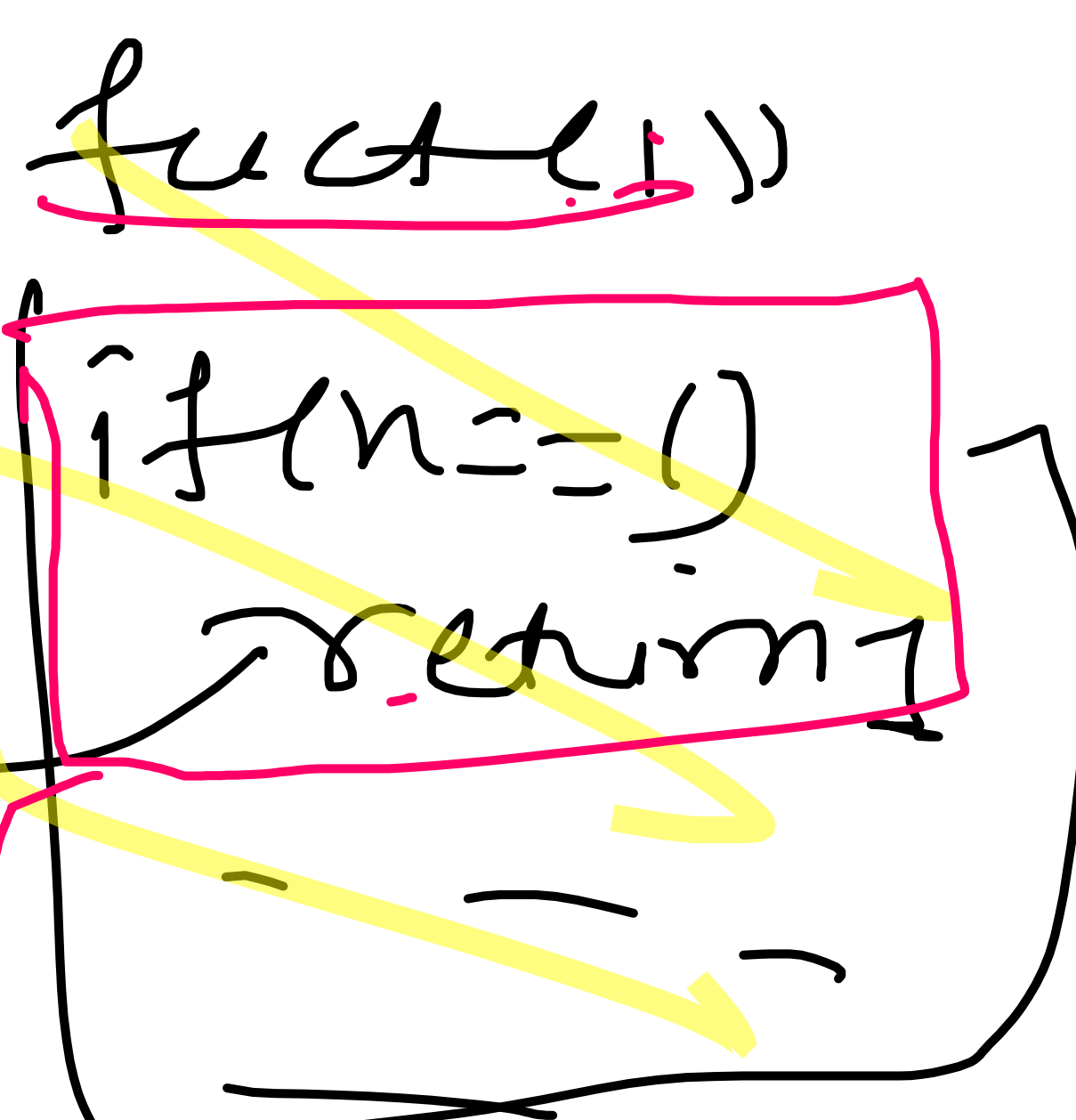
function three() {
 cl("Hi")
}

function two() {
 three()
}

Hi



$3 \times 2 = 6$



no function

Self work

fact()

1

recursion

- ① recursive call
- ② self work
- ③ base case

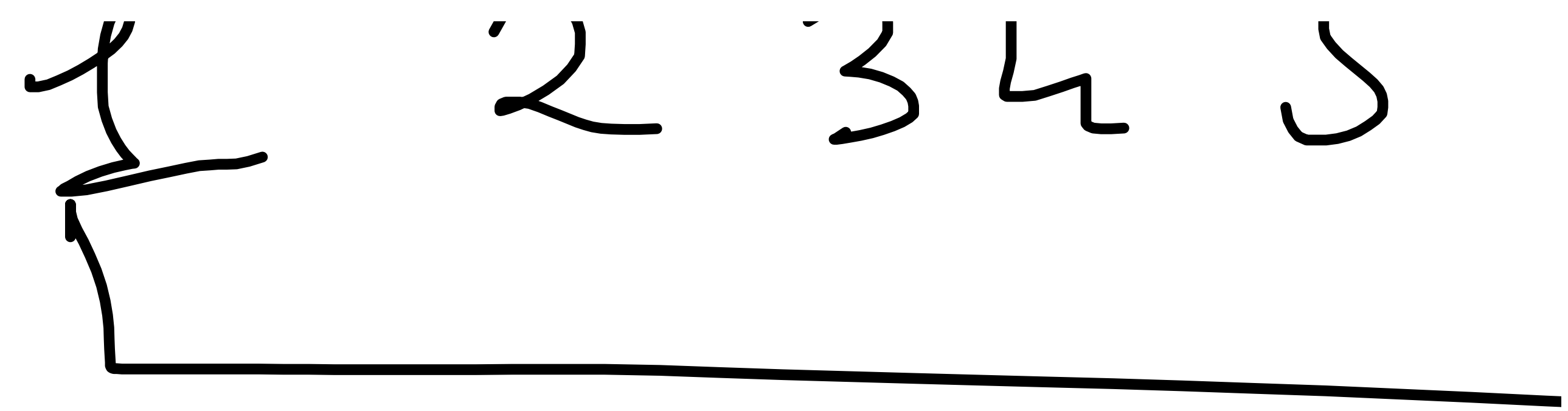
Q

n=5

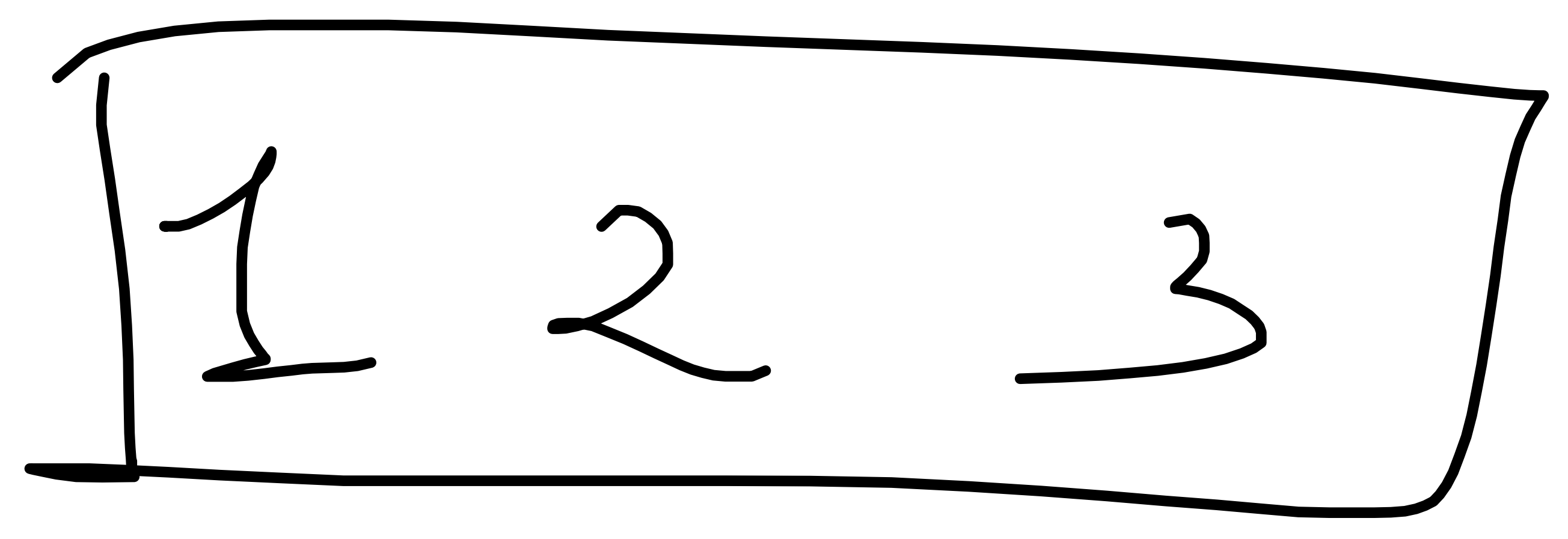
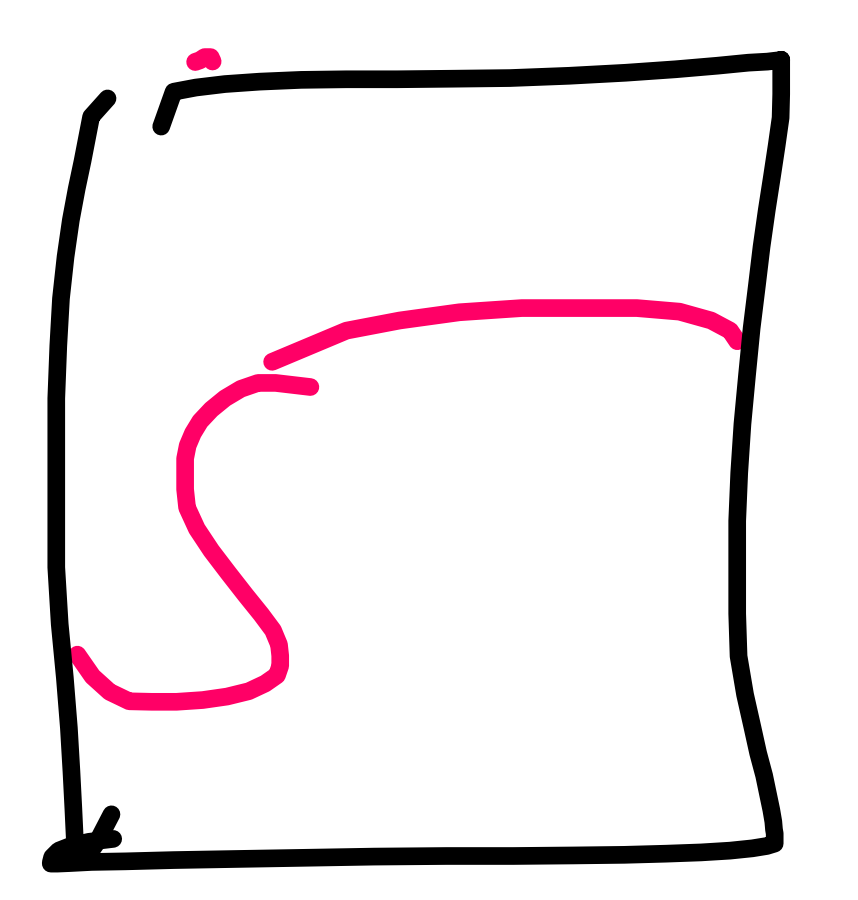
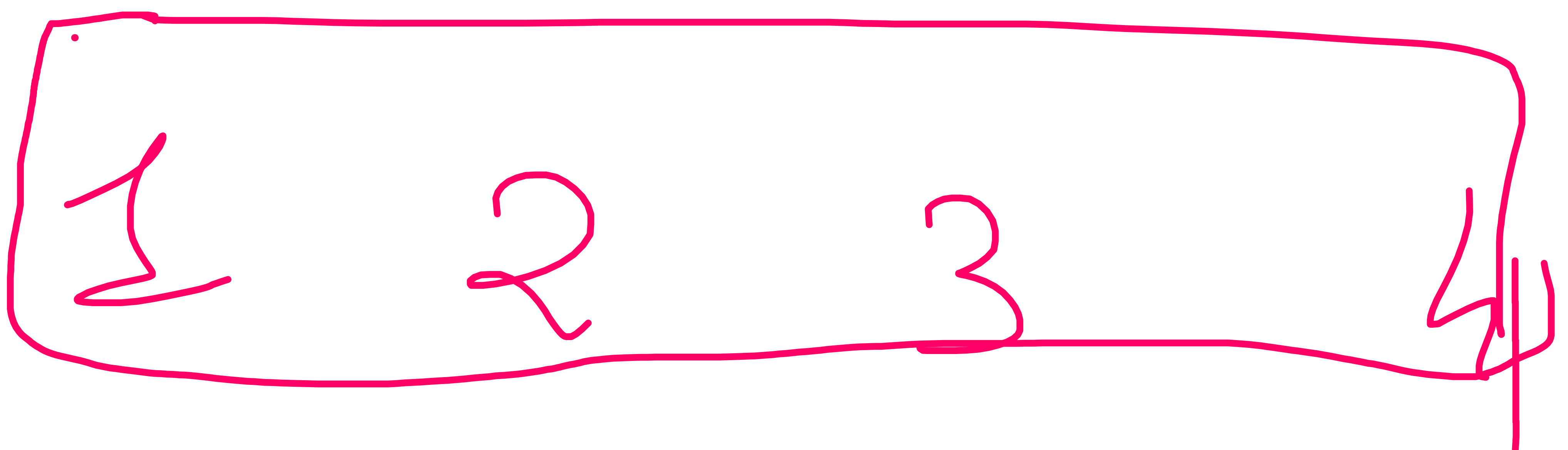
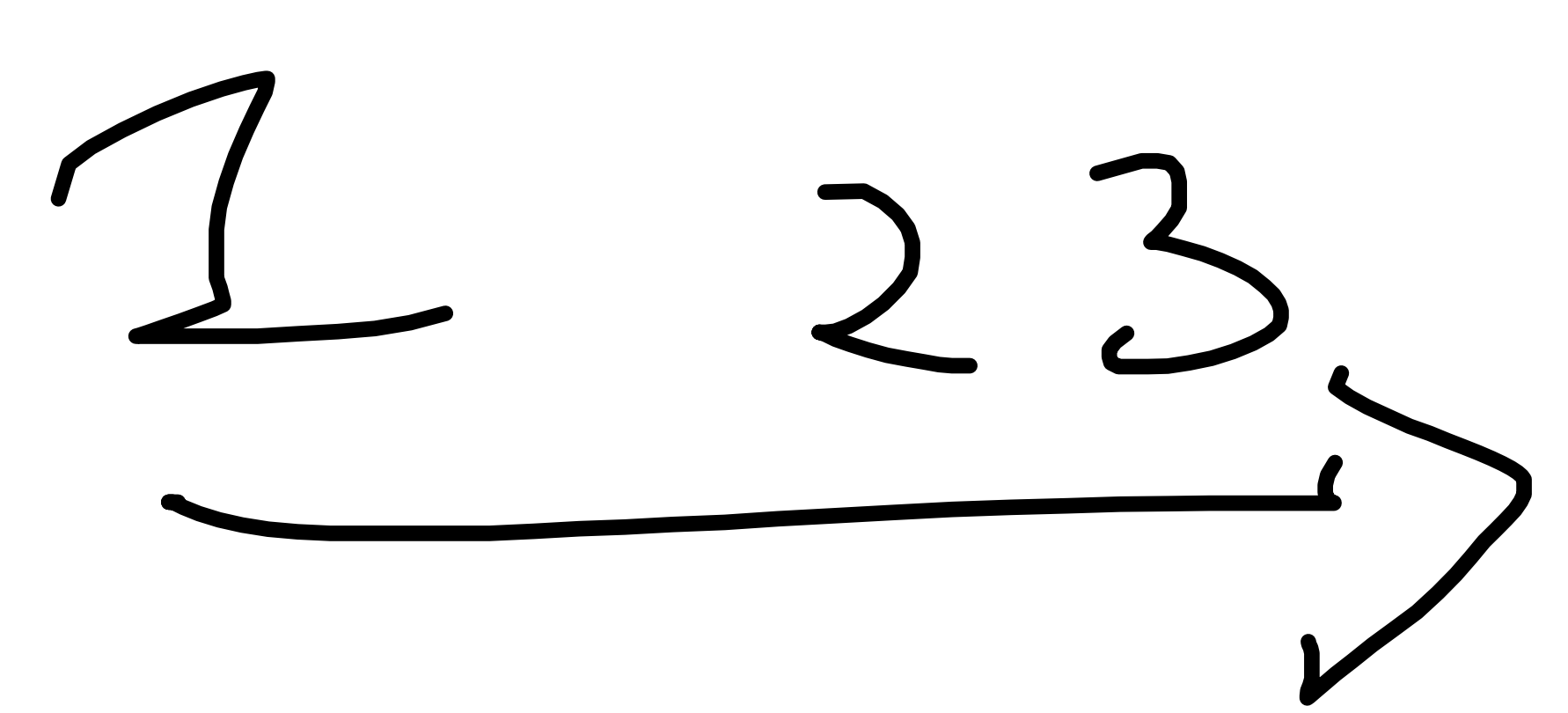
n=7

1 2 3 4

~

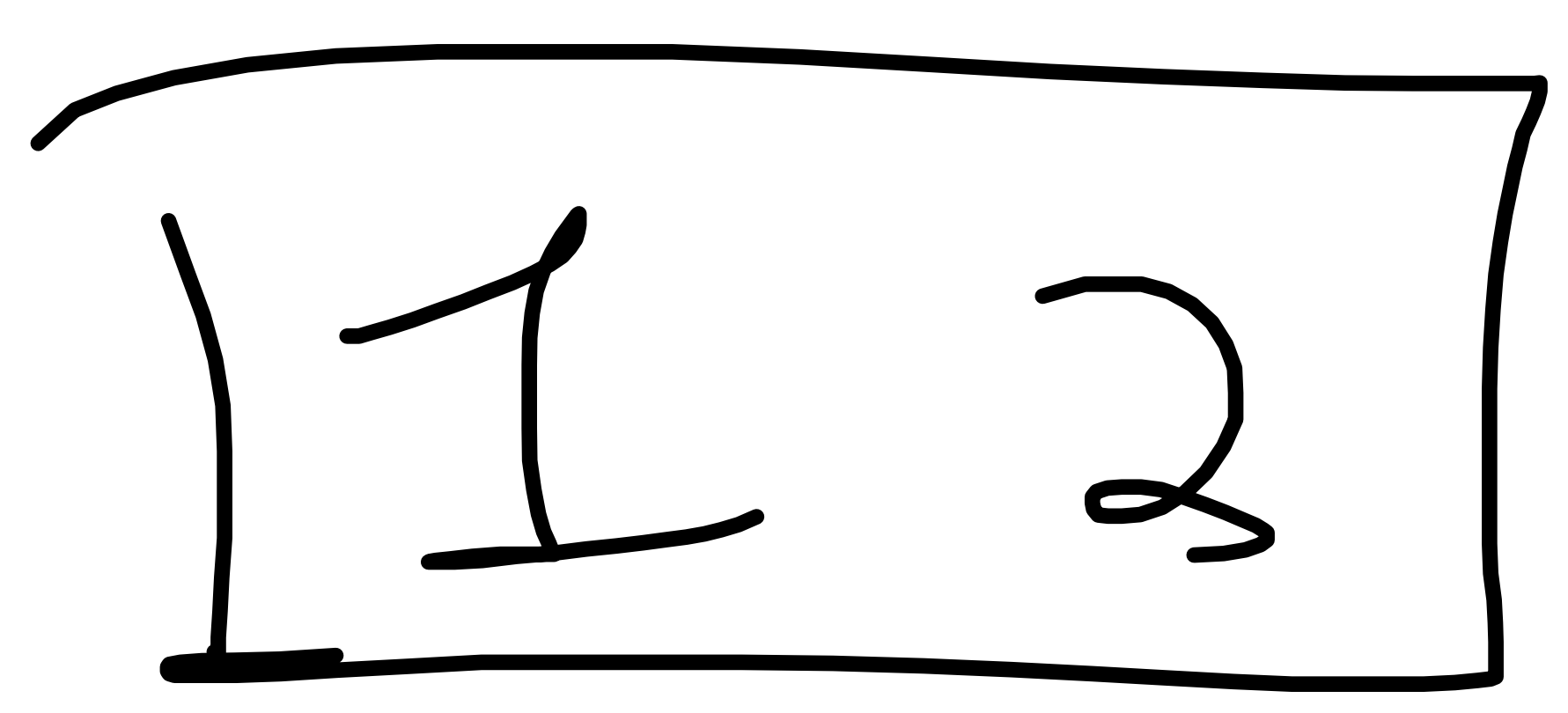


n = 3

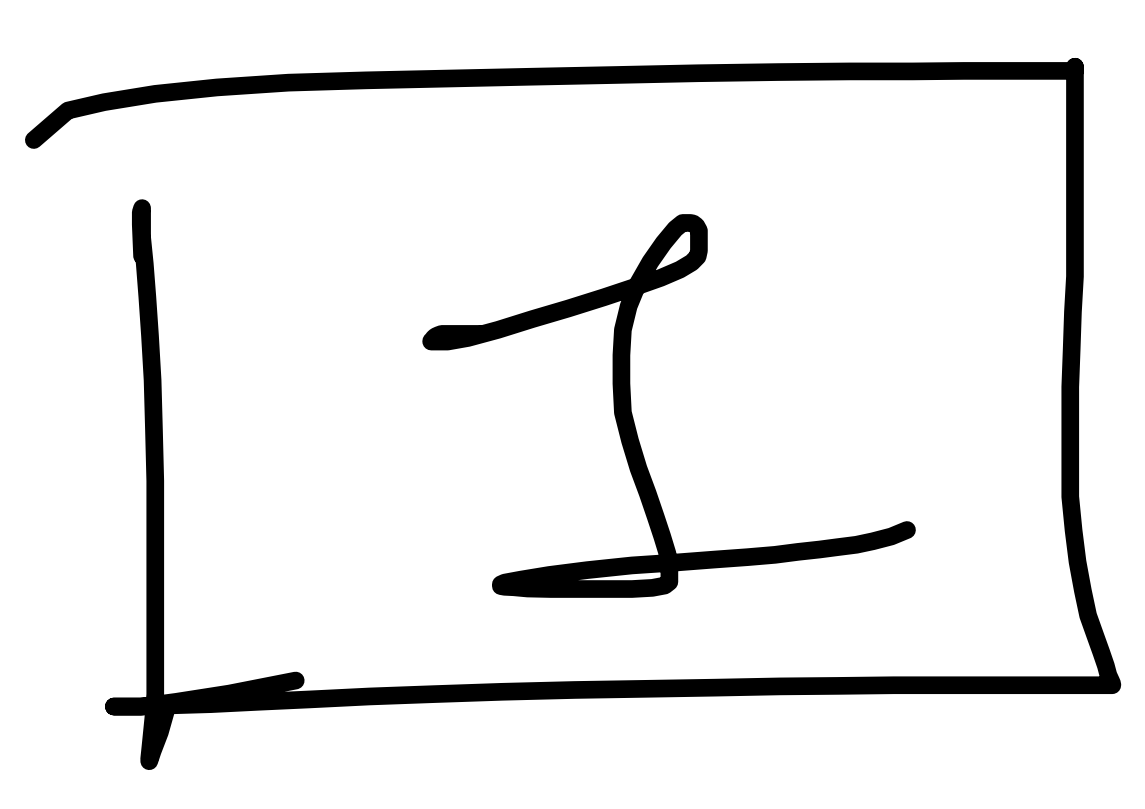


4

print Cr
L



3



2

1

Print(n)

~

~

~

~

2

if (n == 1) {
 c.l.c();
 return
}

// base

}

Print(n-1) // rec

console.log(n) // s

}