CS & IT ENGINEERING

Data Structure

Stack and Quesues

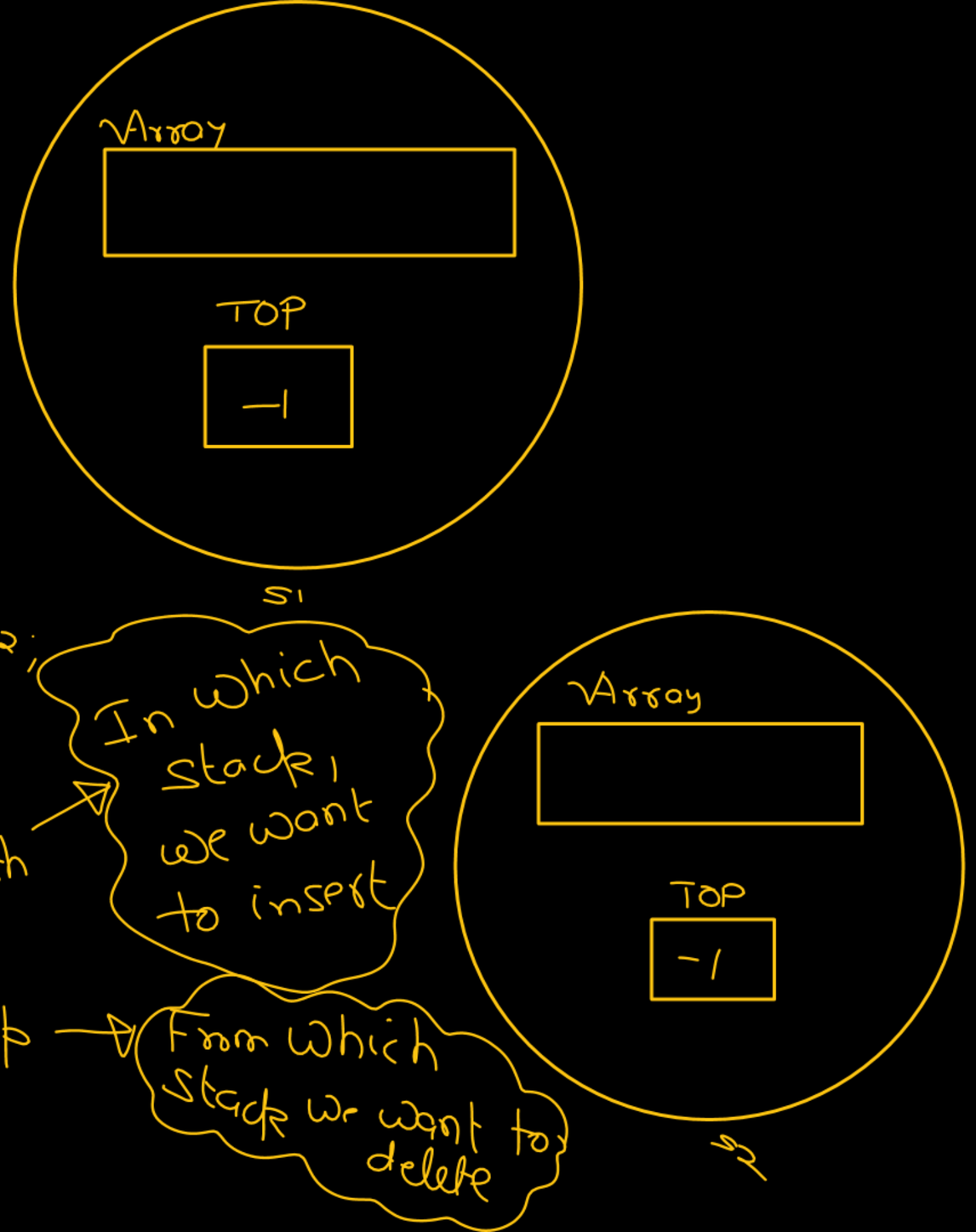
Chapter- 4

Lec- 02

By- Pankaj Sharma sir

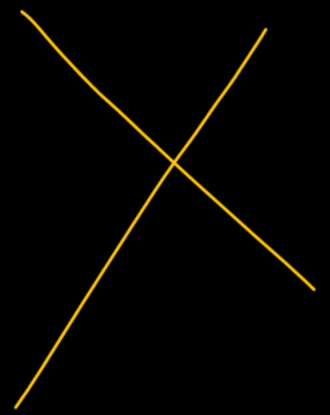
Stack-II

```
struct STACK{
        int Array[10];
        int TOP;
        };
void main(){
        struct STACK S1,S2;
        S1.TOP = -1;
        S2.TOP = -1;
```



```
void main(){
    Push(S1, 10);
```

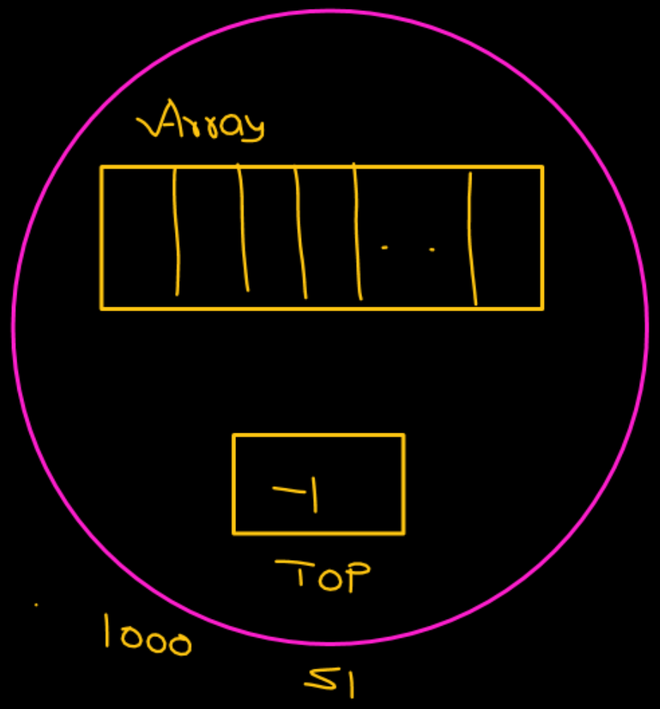
Call by value



```
void Push(struct STACK t, int x)
{
}
```

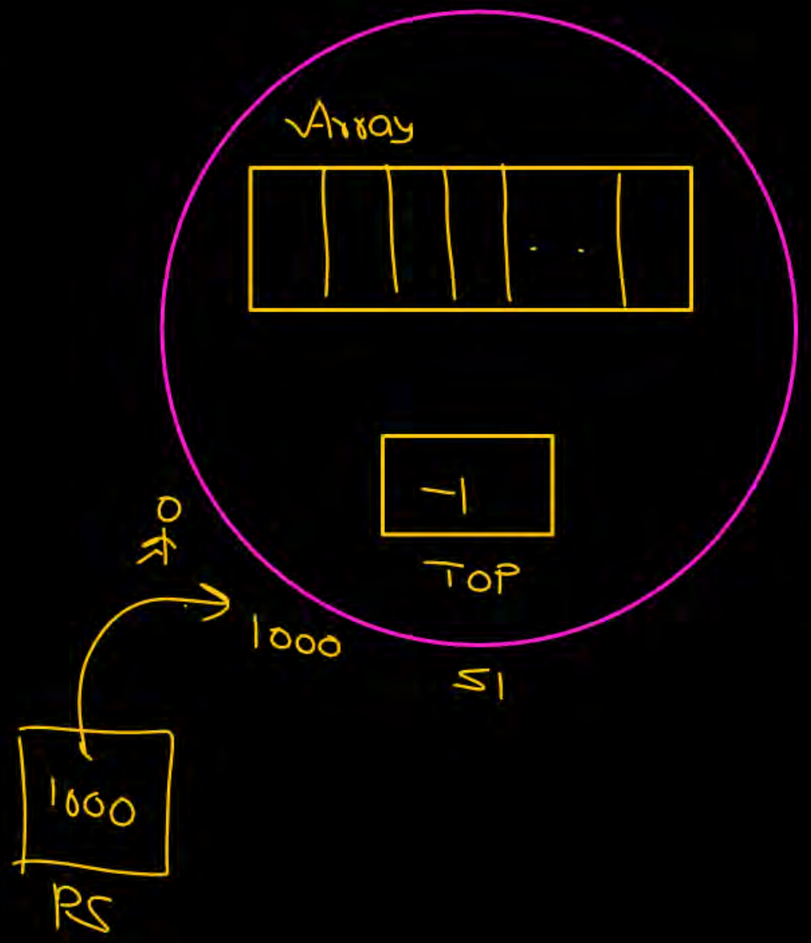
```
void main(){
    struct STACK S1, S2;
    S1.TOP = -1;
    S2.TOP = -1;

    Push(&S1, 10);
```

```
void Push(struct STACK *PS, int x)
{
```




```
}
```

```
void main(){
    struct STACK S1, S2;
    S1.TOP = -1;
    S2.TOP = -1;
    Push(&S1, 10);
```




```
void Push(struct STACK *PS, int x)
{
    if(PS->TOP == SIZE-1)
        return;
    PS->TOP = PS->TOP + 1;
    Array[TOP] = x; X
    PS->Array[PS->TOP] = x; ✓
}
```

Doubt → ①

PS → member1

PS → member2

PS → TOP

(*PS).member1

(*PS).member2

(*PS).TOP

Stack Permutation

$n=3$

1. 1, 2, 3
2. 1, 3, 2
3. 2, 1, 3
4. 2, 3, 1
5. 3, 1, 2
6. 3, 2, 1

} 3! permutation

(i) Insertion order is fix

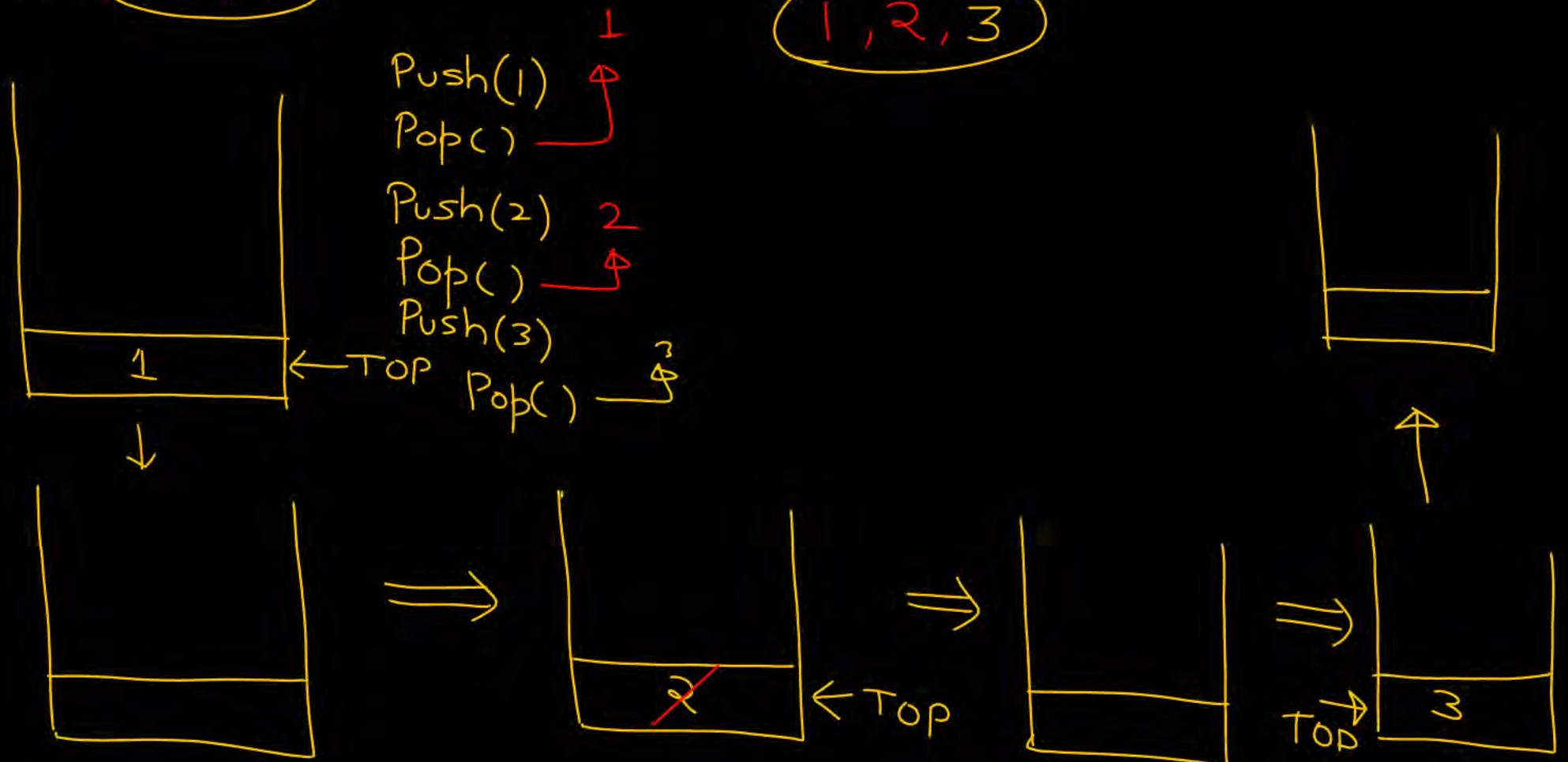
ex 1, 2, 3 →

but you can pop any time

insertion order : 1, 2, 3 →

1) Pop order (1, 2, 3) ✓

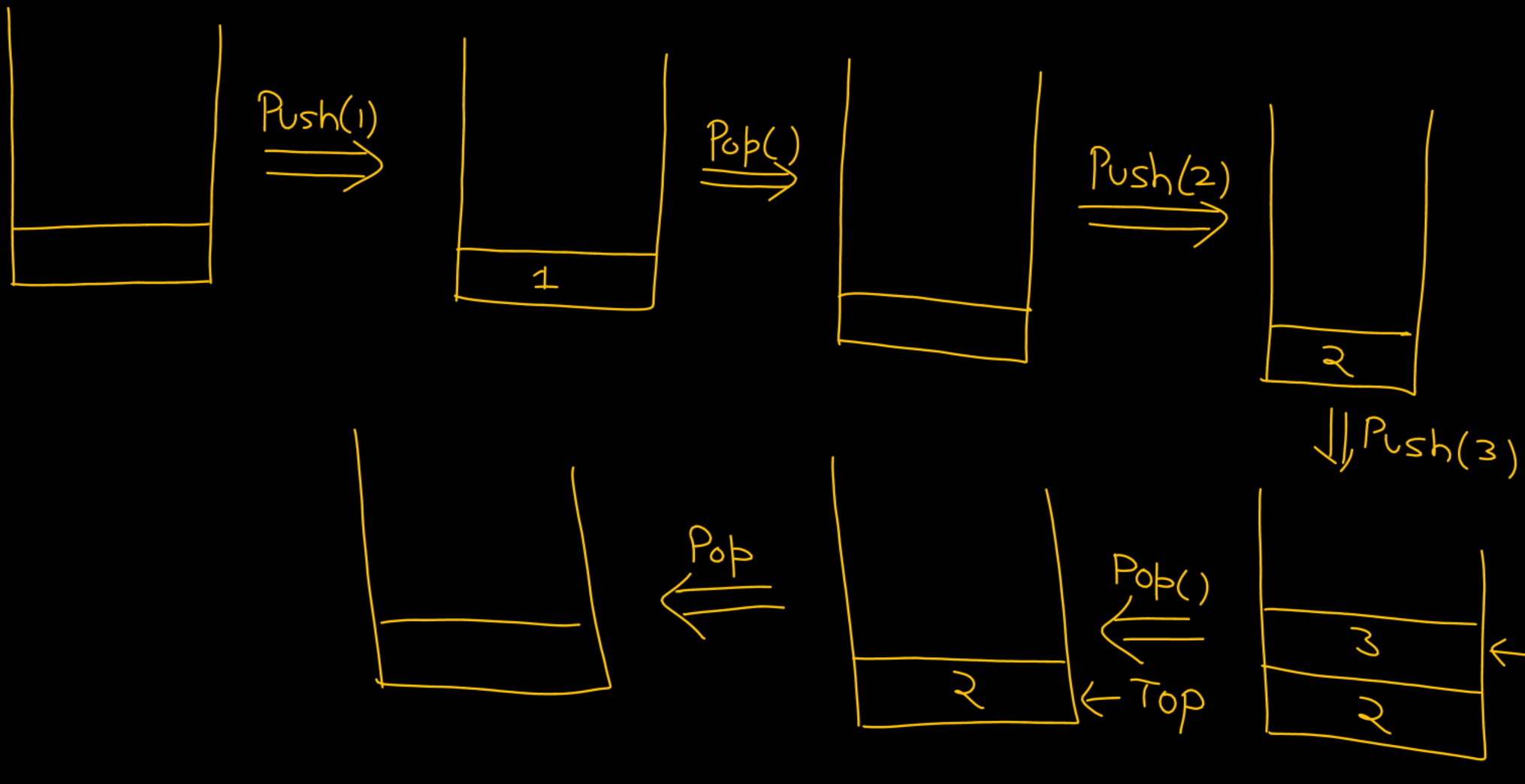
(1, 2, 3)



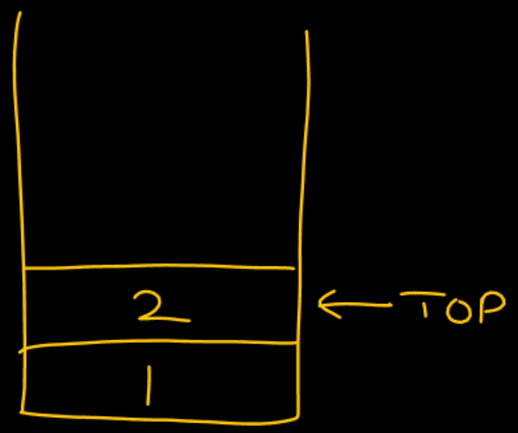
2) 1,3,2 (Pop-seq)

1,3,2

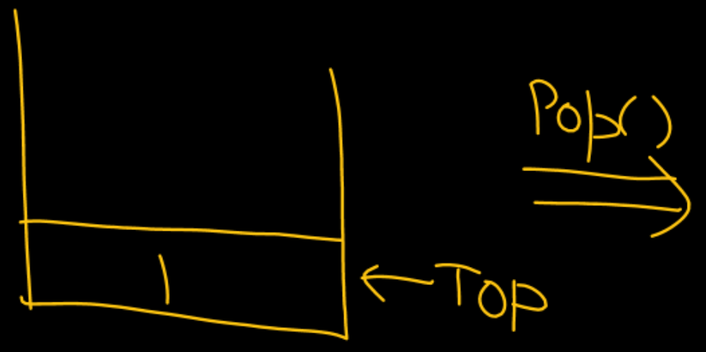
a)



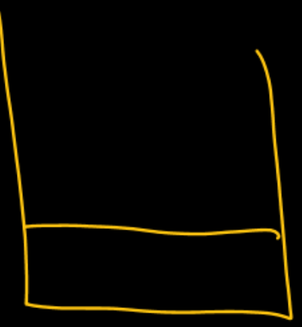
3) 2, 1, 3 ⇒ a) Push(1)
b) Push(2)



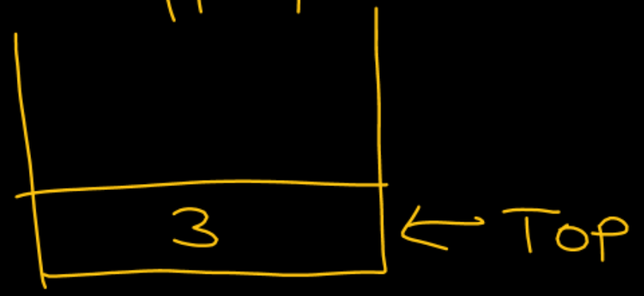
Pop()



Pop()



Push(3)



Pop

2, 1, 3

4) 2, 3, 1

Push(1)
Push(2)
Pop()
Push(3)
Pop()
Pop()

2 3 1

5) 3, 1, 2

Push(1)
Push(2)
Push(3)
Pop()

3, 1, 2 is not a valid stack permutation.

b) 3,2,1

out of 6 possible permutation $\Longrightarrow$ 5 are valid stack permutation

Total valid stack permutation

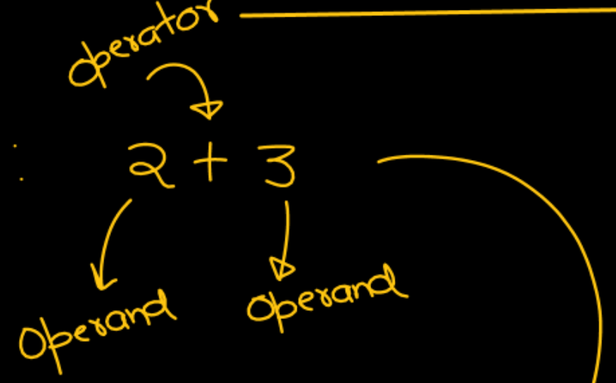
$n = 3$

1,2,3

$\frac{2nC_n}{n+1}$ Catalyn number

${}^{6}C_{4} = 5$

Infix, prefix, postfix



Infix : 2 + 3

Postfix : 2 3 +

Operator comes after operands

Prefix : + 2 3

Operator comes before operands

BODMAS ✗

Why Postfix ?

Eyes

Theory

i/p : 3 + 4 x 6 / 3 ^ 2

left to right scan

Eval :

infix Eval :

→ time consuming

3 + 4 x 6 / 9

Multiple scan needed

Infix Expression Eval.

Applications of stack

1) infix ⟶ postfix
2) Eval. postfix expression

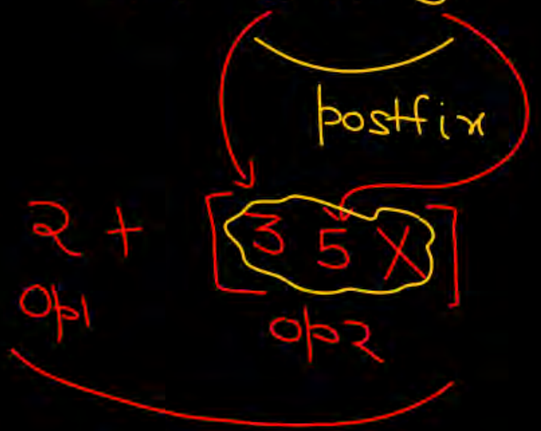
infix to postfix conversion

Priority
Associativity

- without using stack
- using stack

Ex 1

i/p: 2 + 3 × 5



Postfix : 2 3 5 X +

Theory

i/p (infix) : 3 + 5 x 6 / 2 ^ 4

3 + 5 x 6 / [2 4 ^]

3 + [5 6 x] / [2 4 ^]

op1 op2

3 + [5 6 x 2 4 ^ /]

op1 op2

postfix : 3 5 6 x 2 4 ^ / +

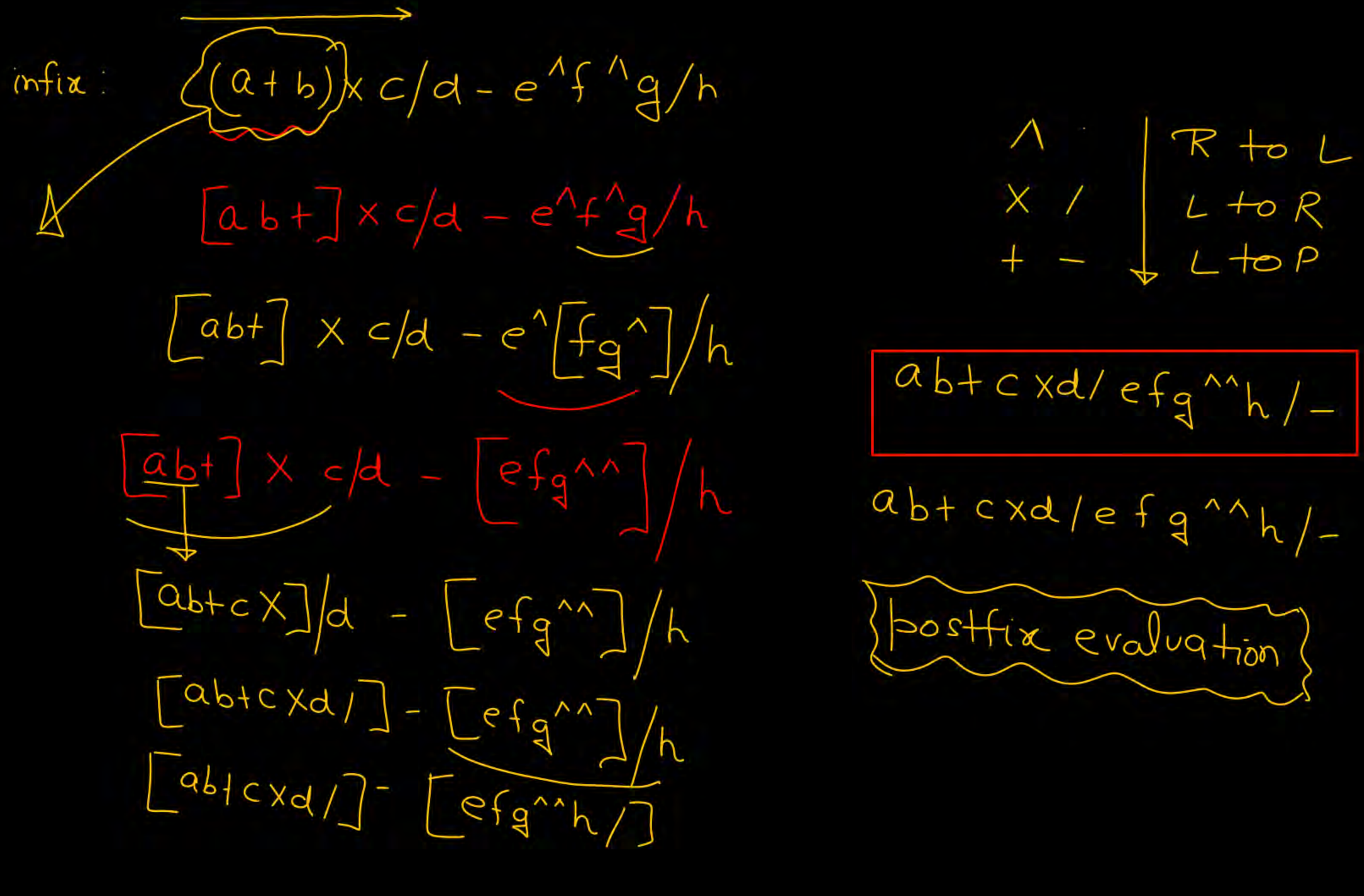
↑ RtoL

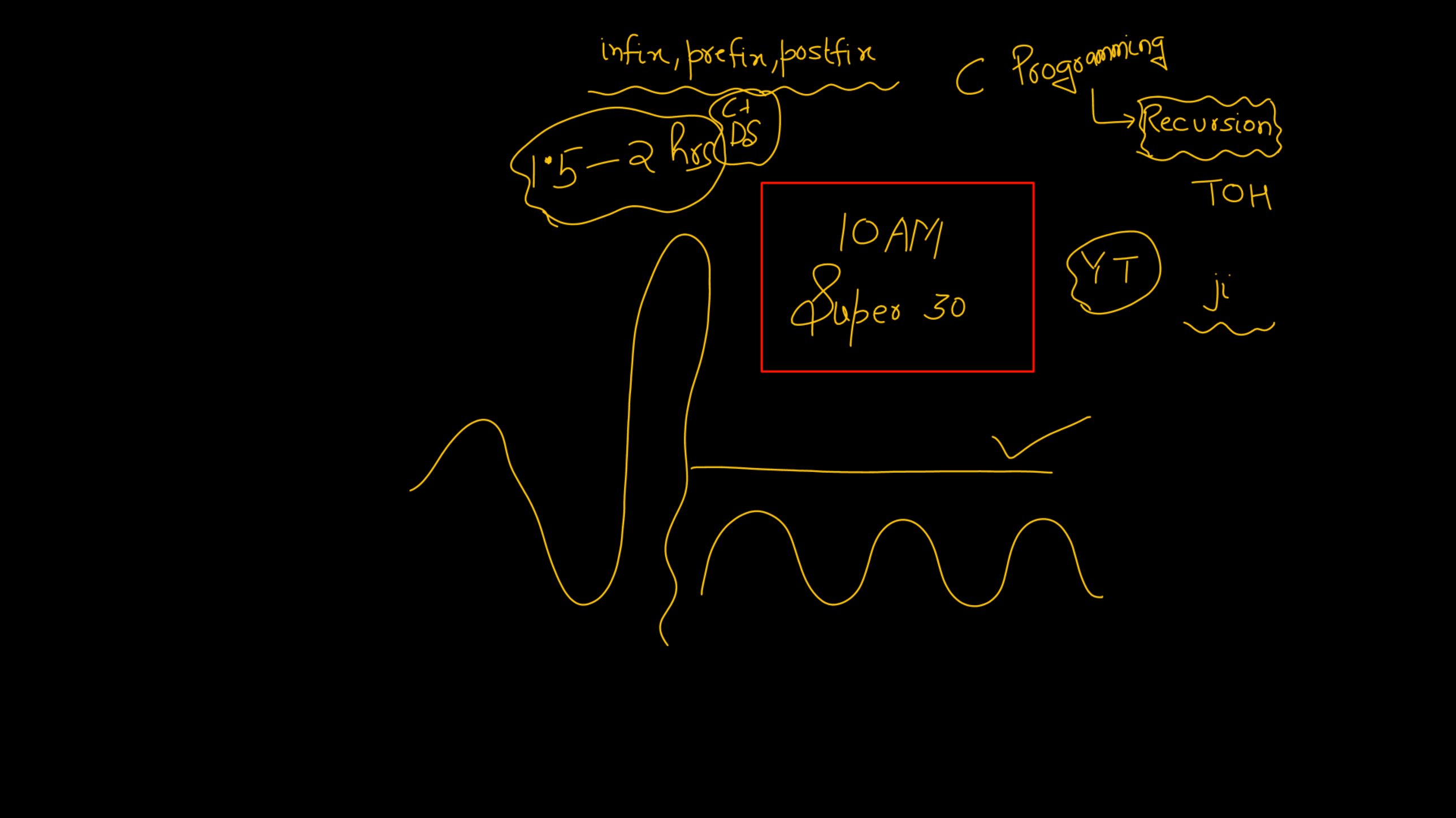
x , / LtoR

+ , - LtoR

10 AM

Super 30


infix : (a+b)x c/d - e^f^g/h
[ab+] x c/d - e^f^g/h
[ab+] x c/d - e^[fg^]/h
[ab+] x c/d - [efg^^]/h
[ab+cx]/d - [efg^^]/h
[ab+cxd/] - [efg^^]/h
[ab+cxd/] - [efg^^h/]
^ R to L
x / L to R
+ - L to P
ab+cxd/efg^^h/-
ab+cxd/efg^^h/-
postfix evaluation


infix, prefix, postfix
C Programming
Recursion
TOH
1.5 – 2 hrs
C++
DS
10 AM
Super 30
YT
ji


THANK YOU GW
SOLDIERS !