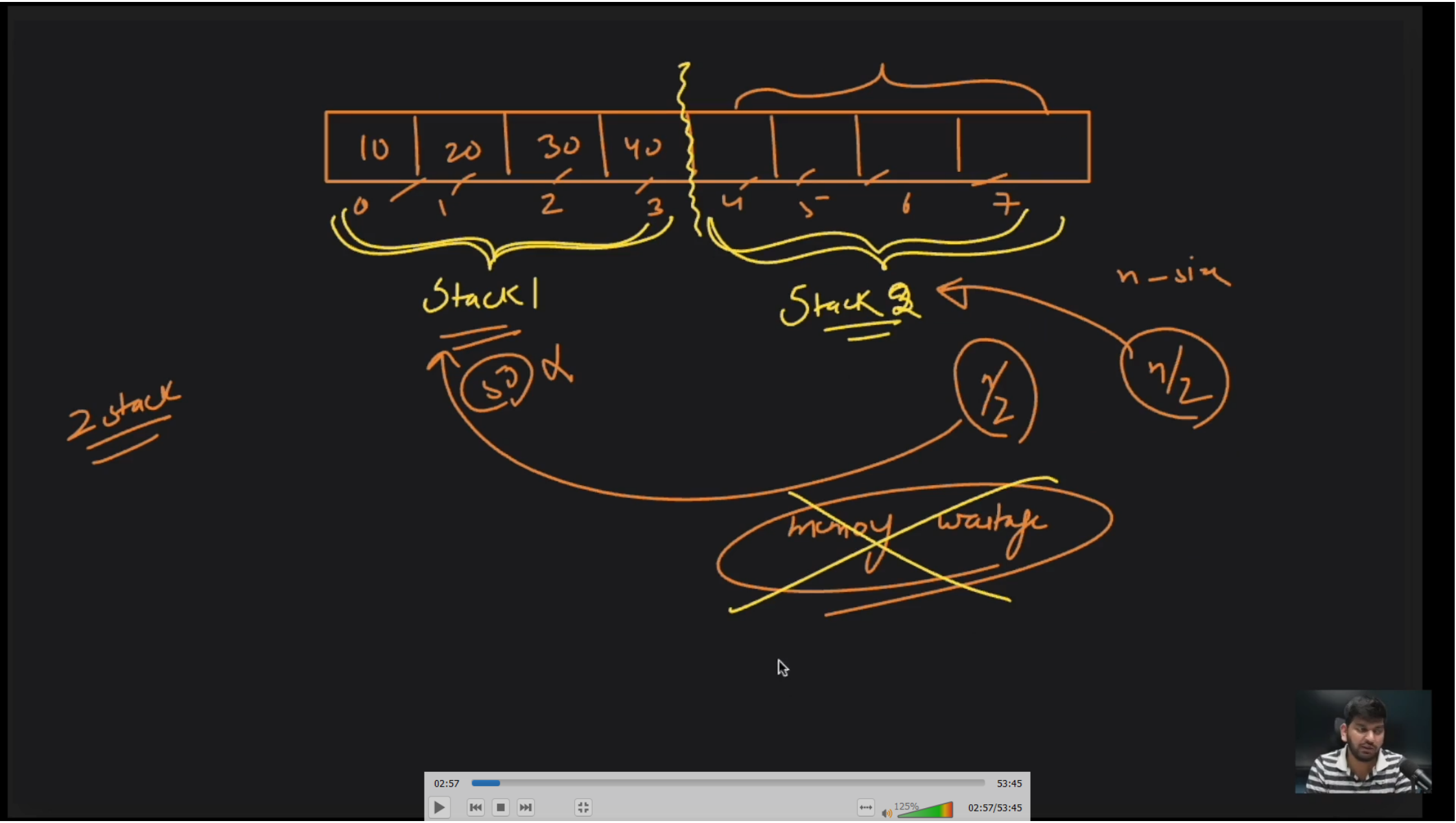
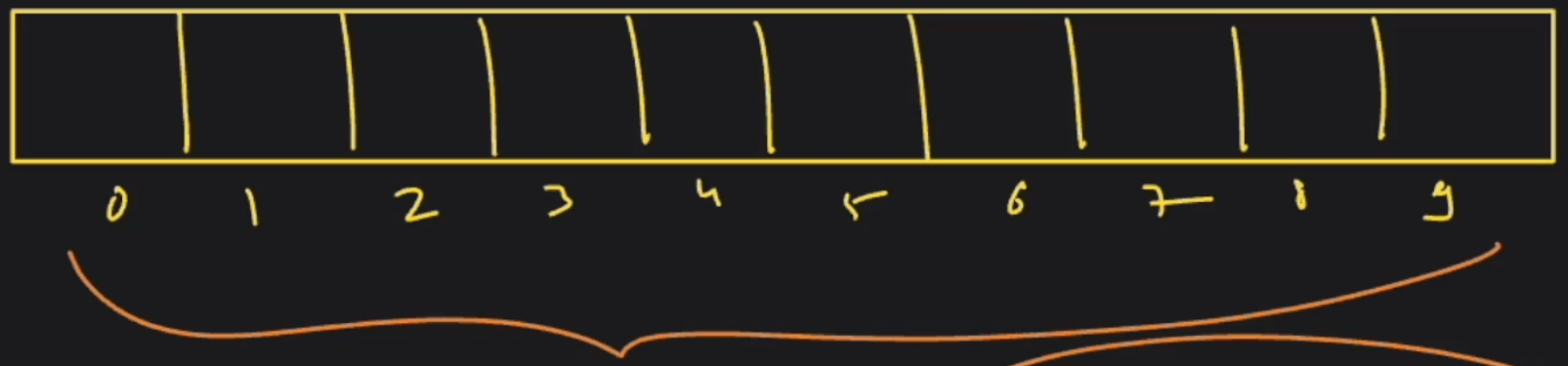

10
20
30
40
0
1
2
3
4
5
6
7
Stack 1
Stack 2
n - size
n/2
n/2
2 stack
memory wastage
02:57
53:45
125%
02:57/53:45

#2



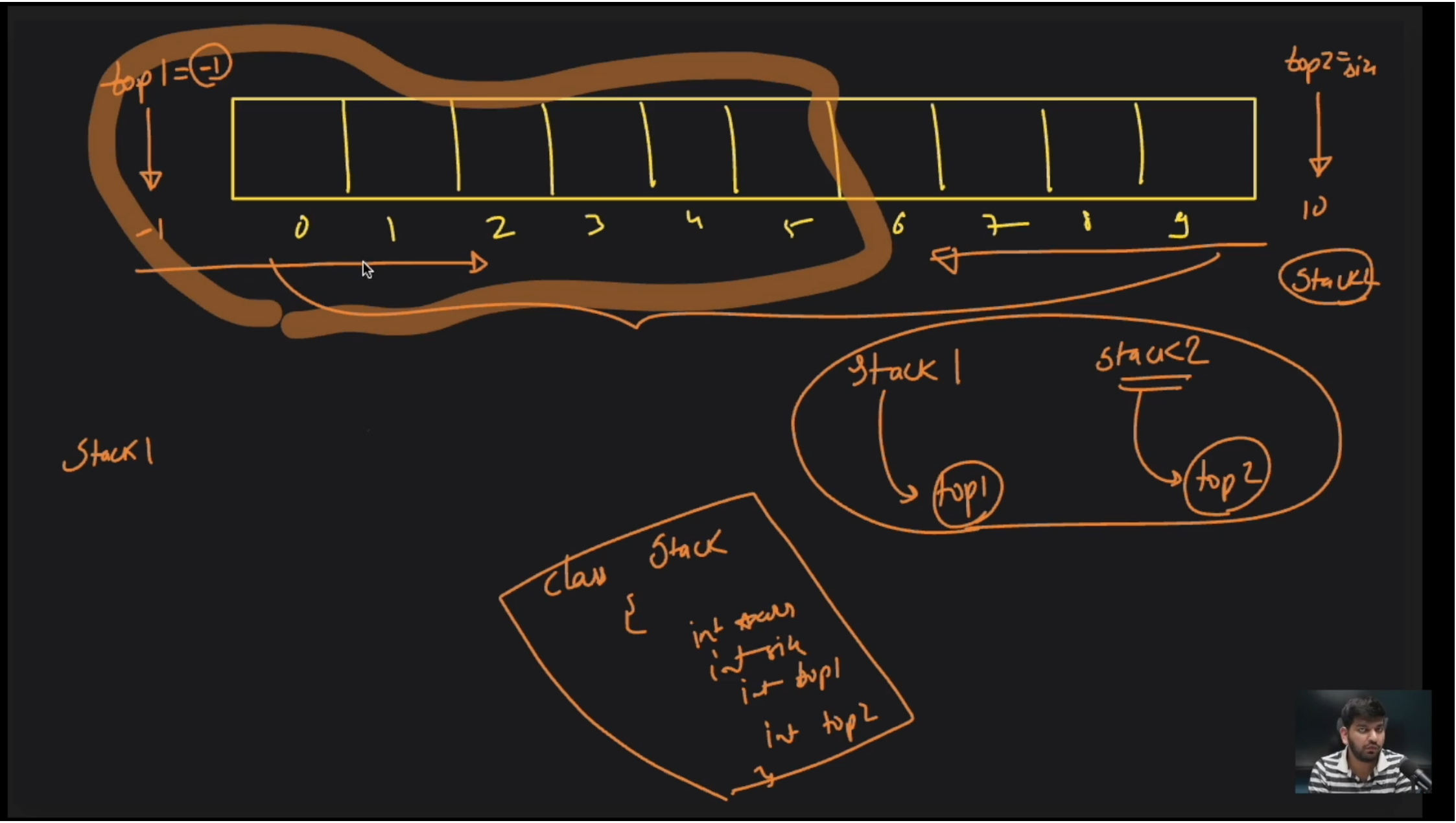
stack1 stack2

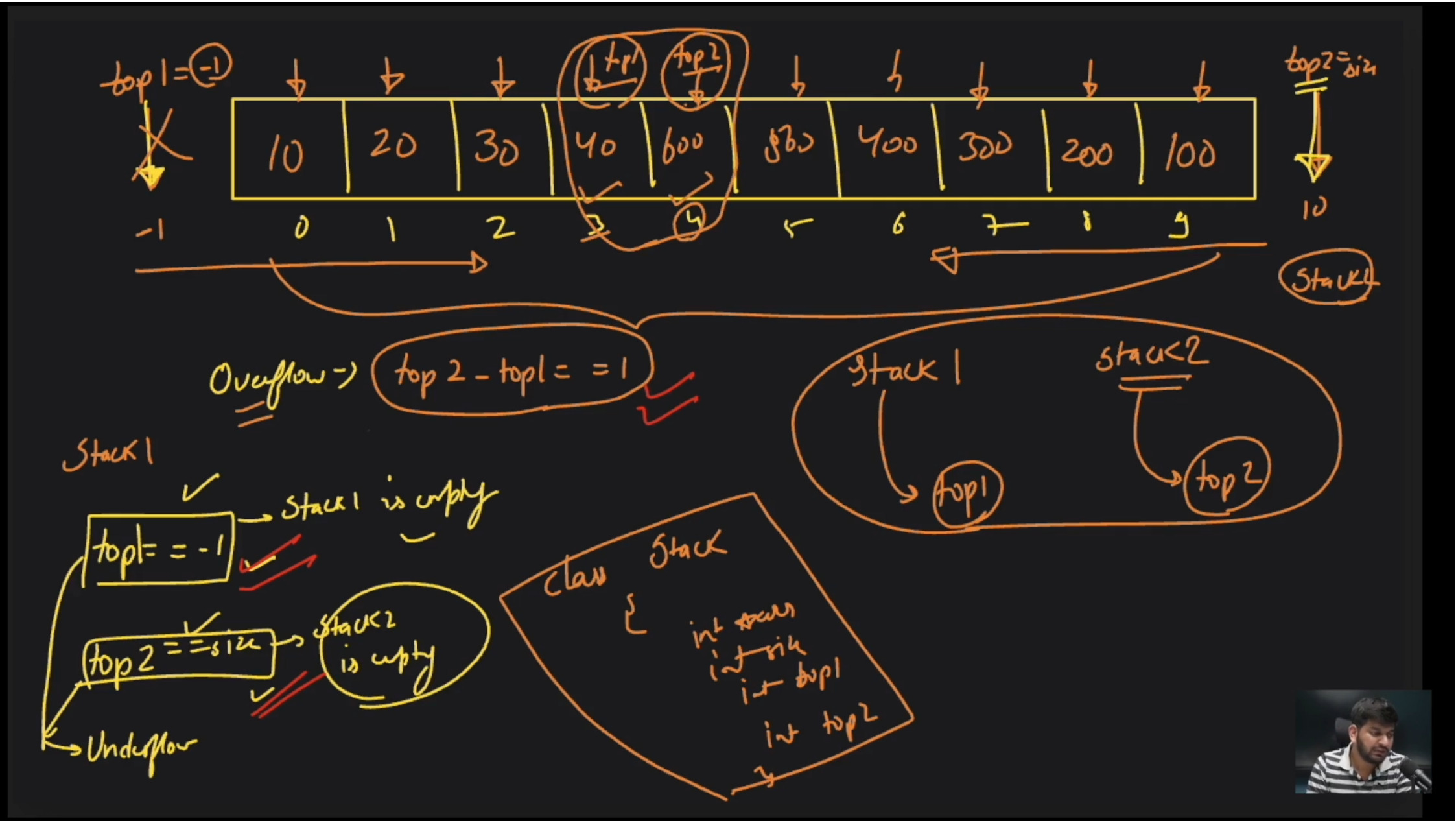
top1 top2

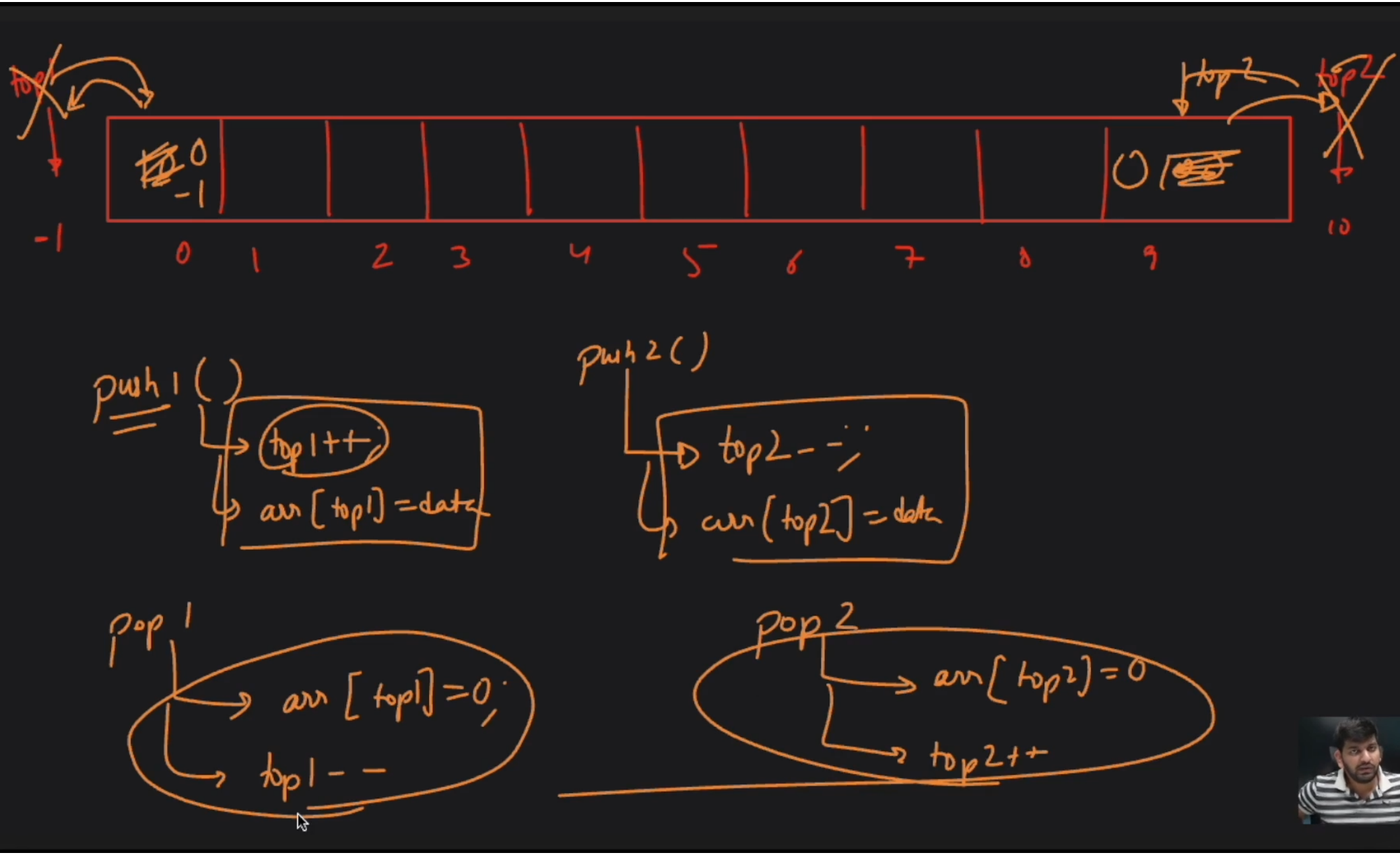
int * arr

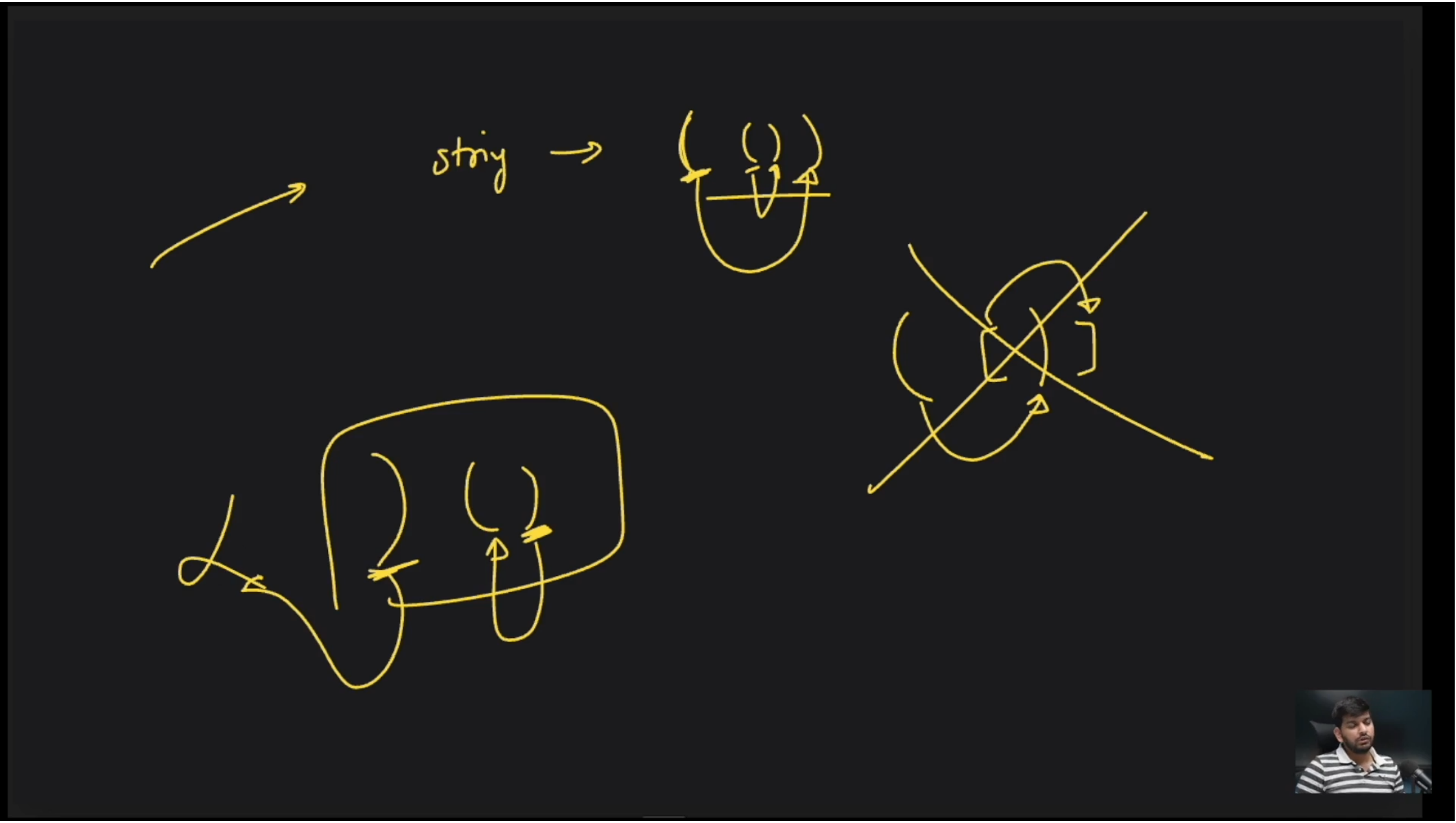
size

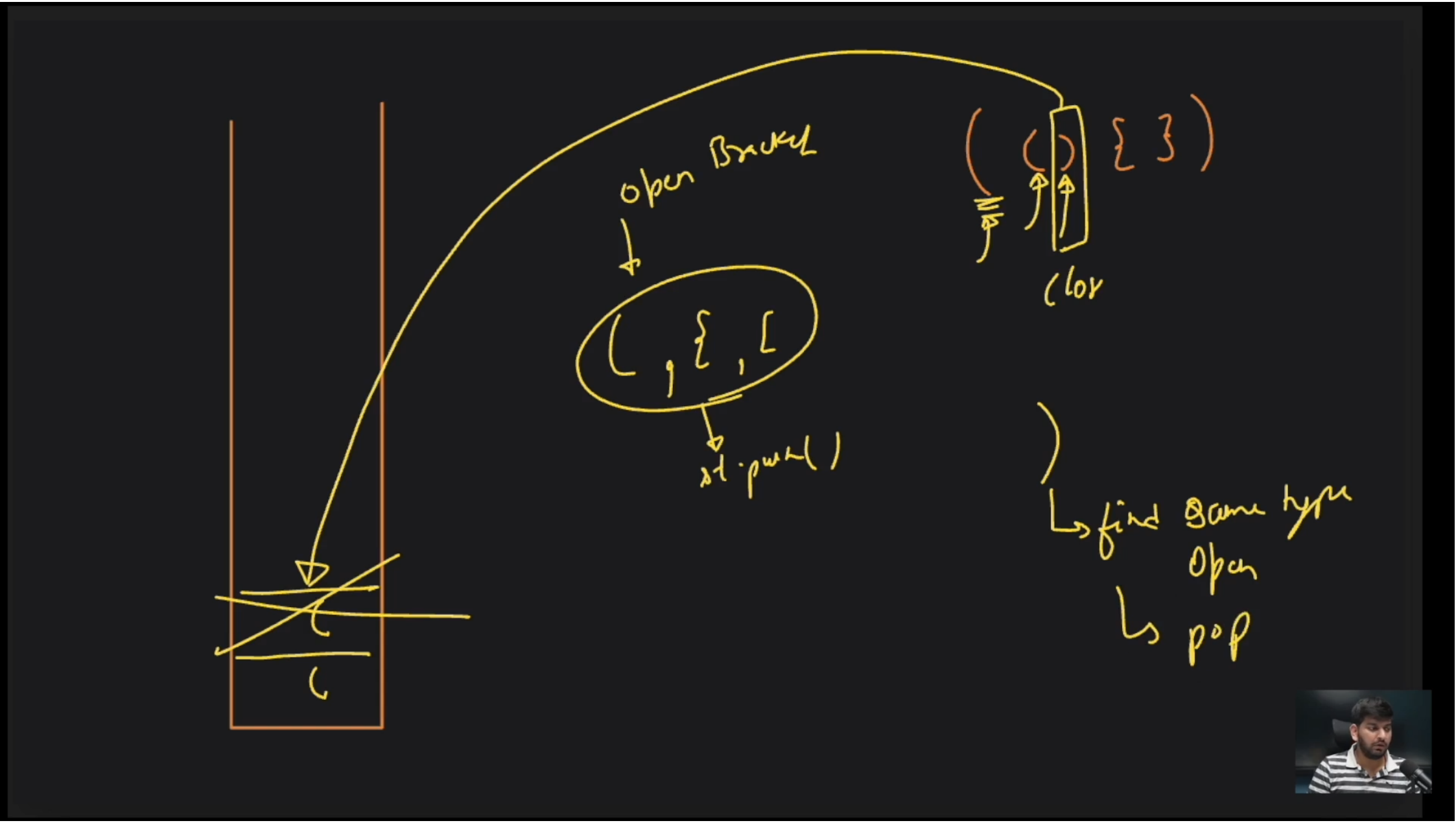
class Stack
{
int *arr
int size
int top1
int top2
}

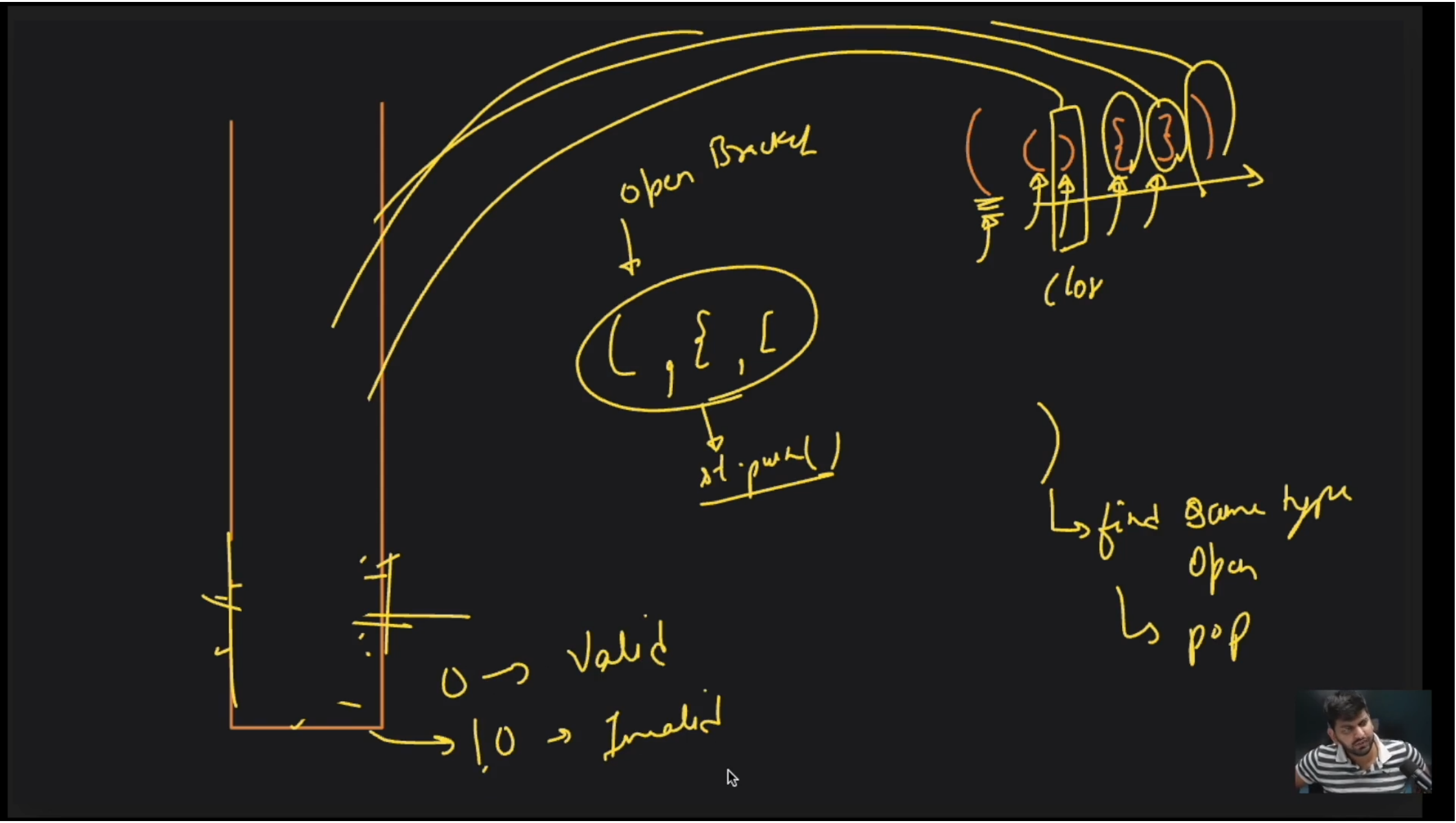

top1 = -1
top2 = size
-1
0 1 2 3 4 5 6 7 8 9
10
Stack1
Stack1
Stack2
top1
top2
Stack1
class Stack
{
int arr
int size
int top1
int top2
}

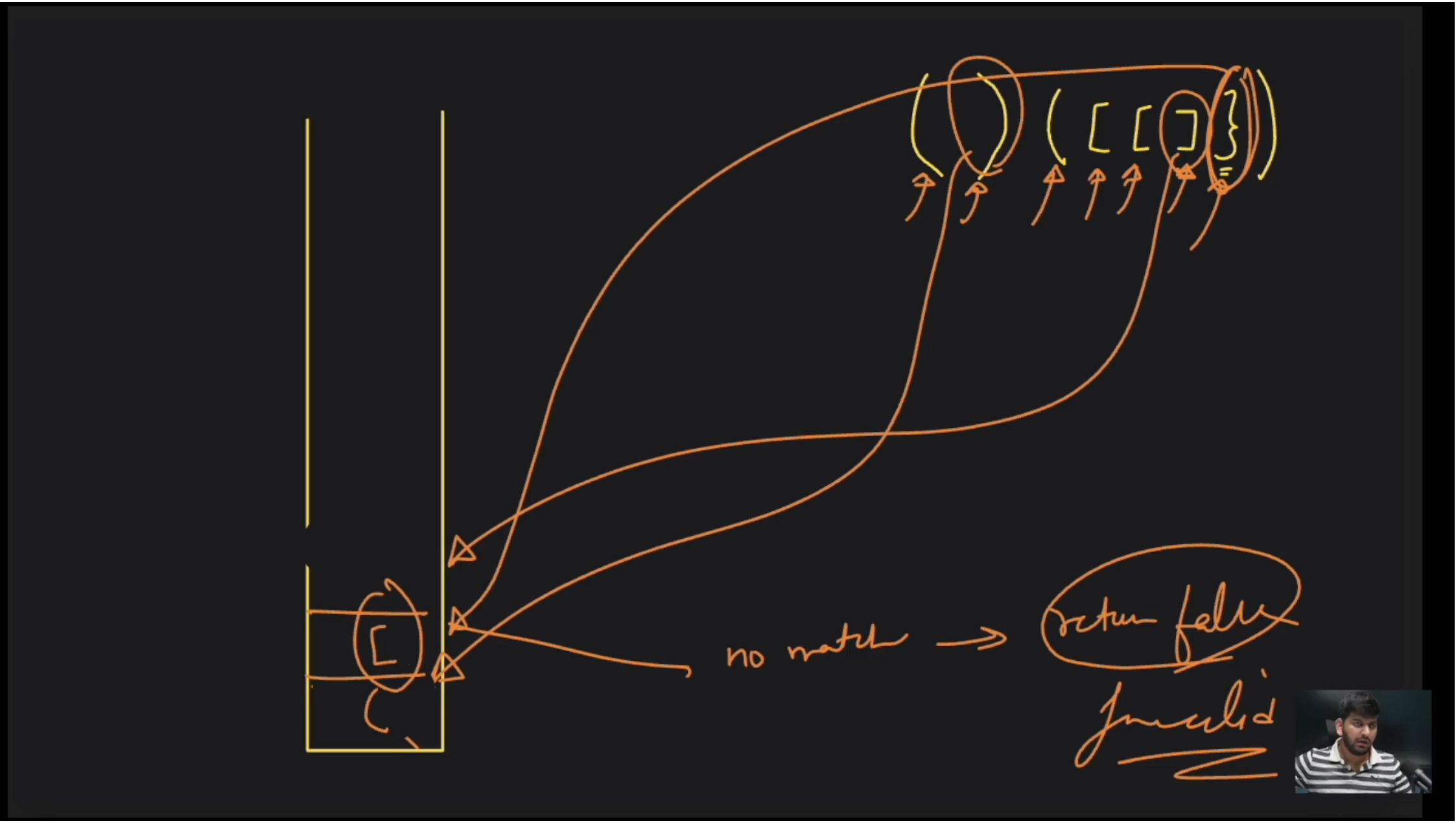

top1 = -1
top1
top2
top2 = size
10
20
30
40
600
800
400
300
200
100
-1
0
1
2
3
4
5
6
7
8
9
10
Stack1
Overflow → top2 - top1 = = 1
Stack1
Stack2
top1
top2
Stack1
top1 = = -1
Stack1 is empty
top2 = = size
Stack2 is empty
Underflow
class Stack
{
int stack
int size
int top1
int top2
}


top1
top2
top2
0
-1
0
-1
0 1 2 3 4 5 6 7 8 9
10
push1()
top1++
arr[top1]=data
push2()
top2--
arr[top2]=data
pop1
arr[top1]=0
top1--
pop2
arr[top2]=0
top2++


string


open Bracket
(, { , [
st.push()
(( ) { })
clos
)
find same type
Open
pop


open Bracket
(, { , [
st.push()
Close
)
↳ find same type
Open
↳ pop
0 → Valid
!0 → Invalid


no match
return false
invalid

Remove Redundant Brackets

(a + (b * c)) → False

((a) + (b * c)) → True

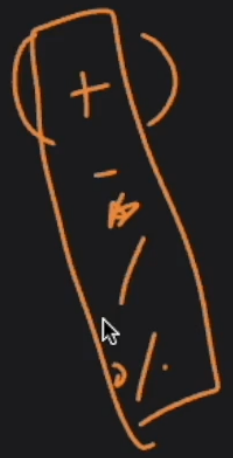
(a + b) → False

(a + (b)) → True

((a + b)) → True

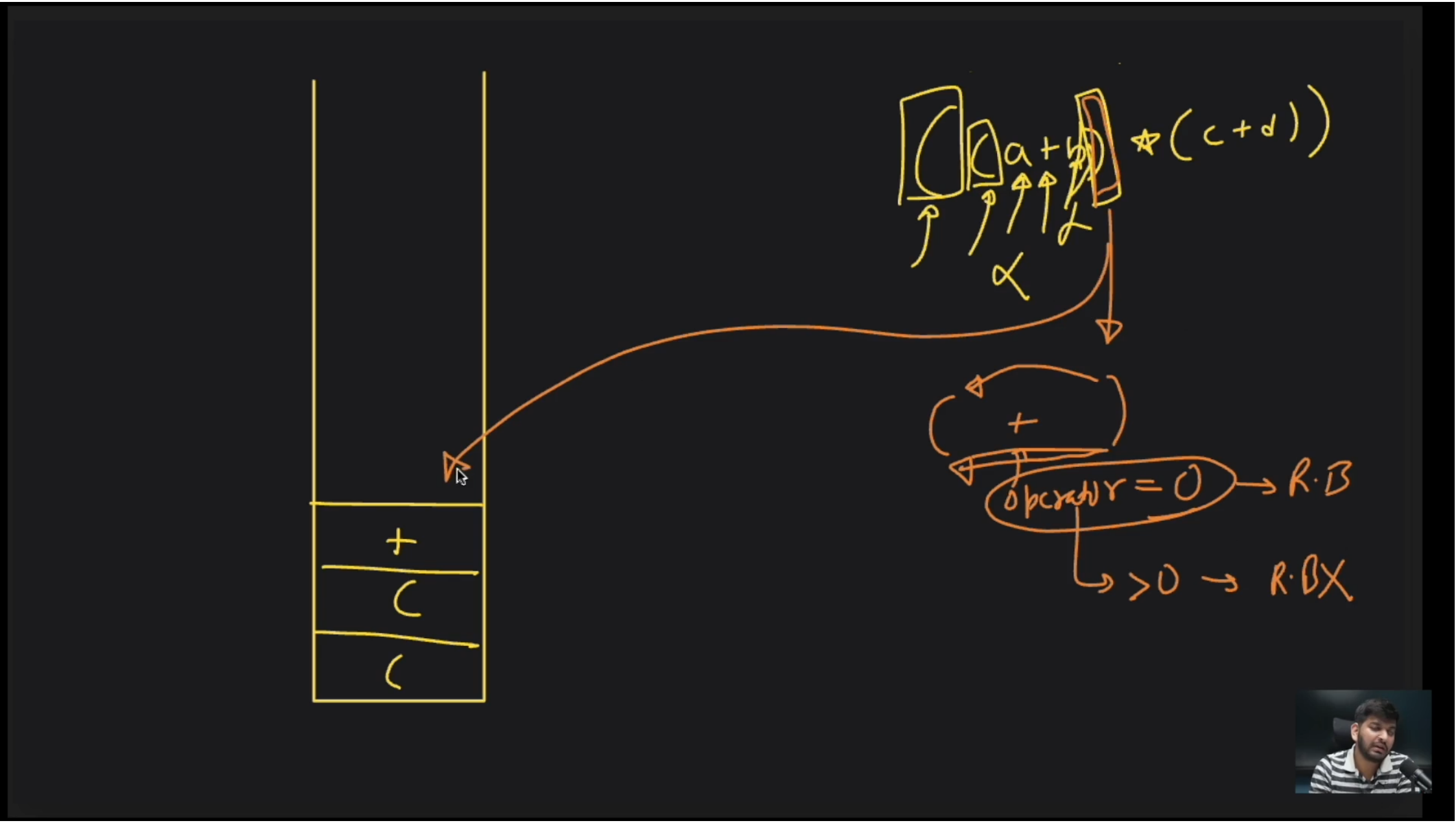
((a) + (b)) → True

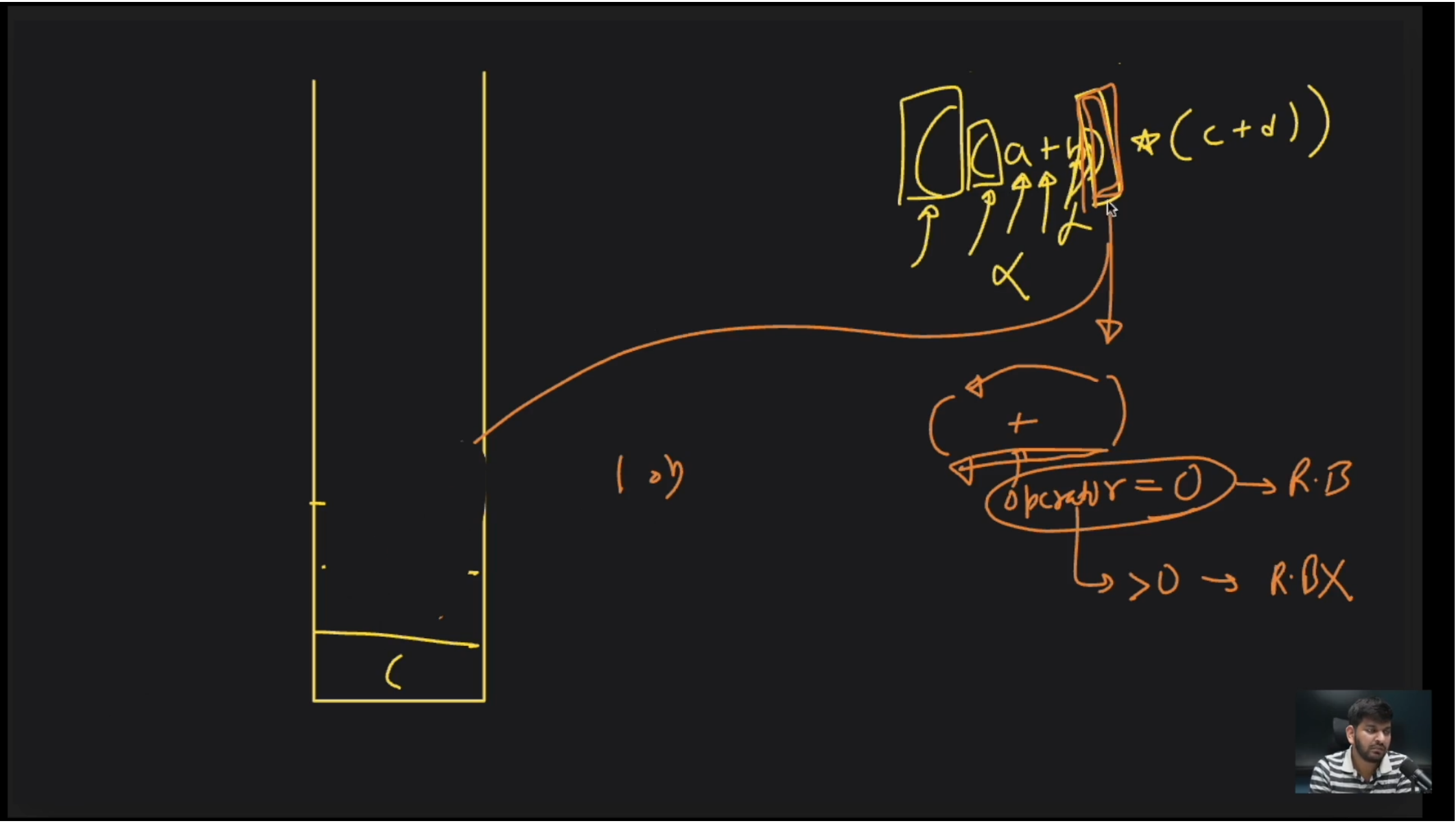
(a + b * c) → False

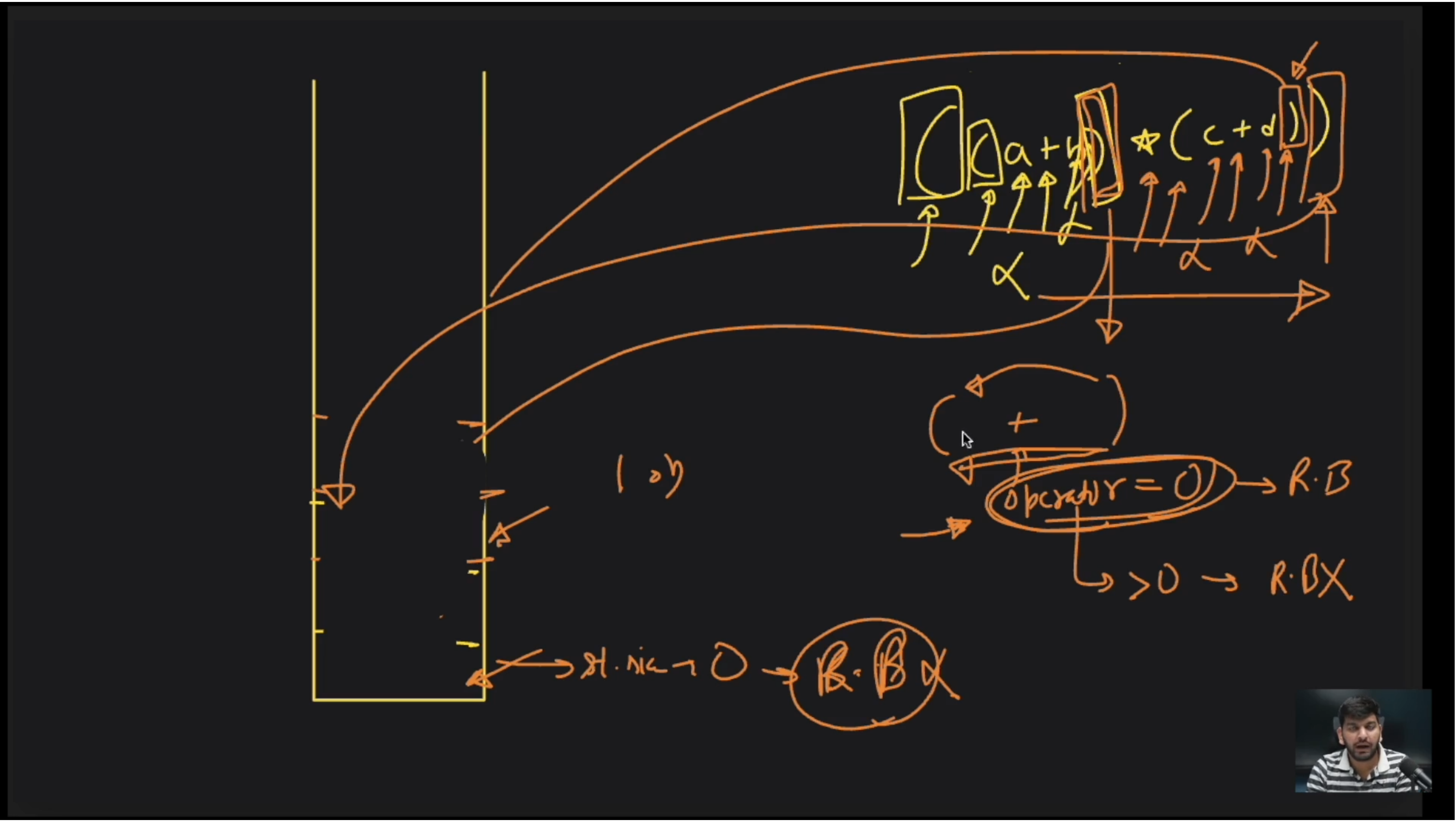



(c+d)!


( )
+
-
1


+
C
C
a+b
*(c+d)
+
Opcode Y = 0
R·B
>0 → R·BX


a+b
*(c+d)
+
opcode
r = 0
R·B
>0
R·BX


((a+b)*(c+d))
+
operator = 0 → R.B
> 0 → R.BX