Doubt Class with Lakshay

Special class

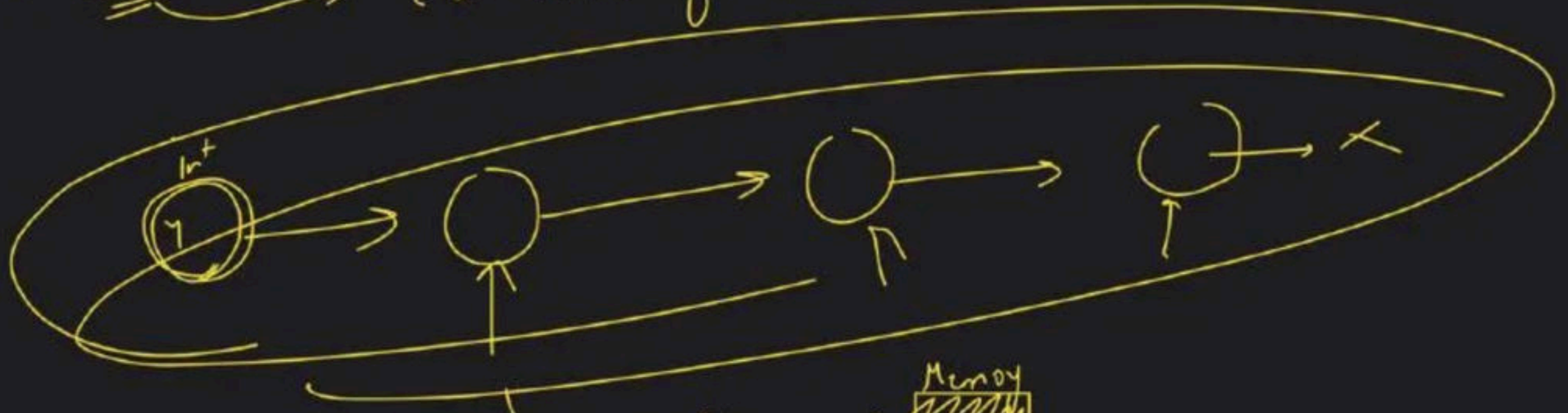
Love Babbar • July 27, 2024

Linked List Class - 1

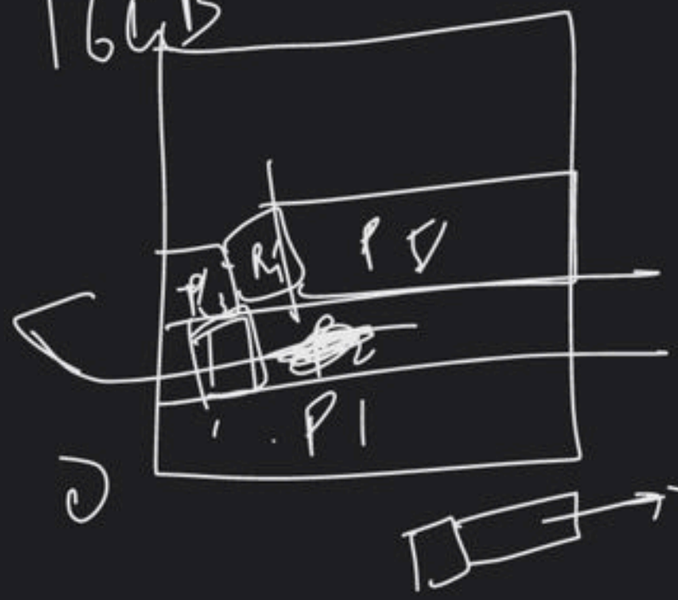
Special class

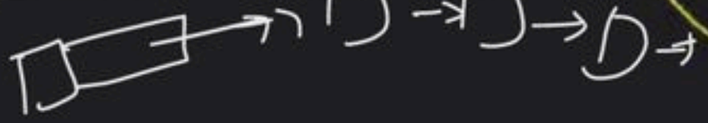
Love Babbar • July 15, 2024

→ Linked List → collection of Nodes

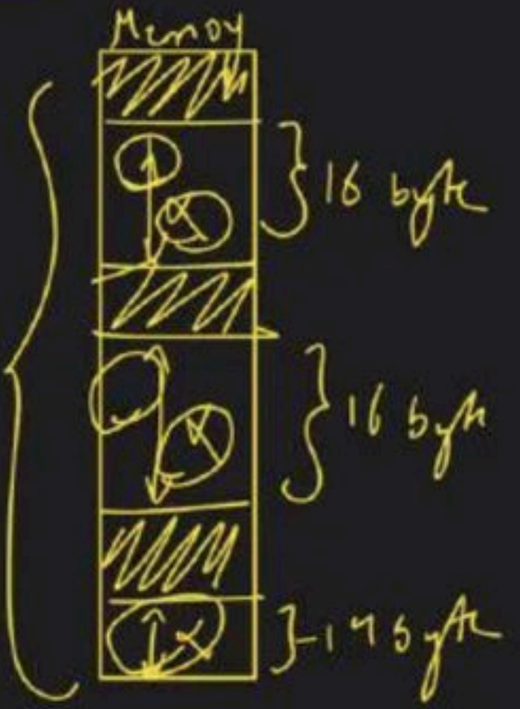


16GB

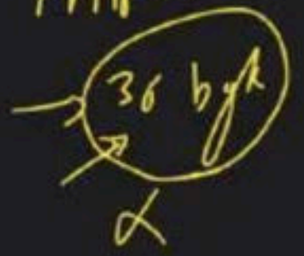


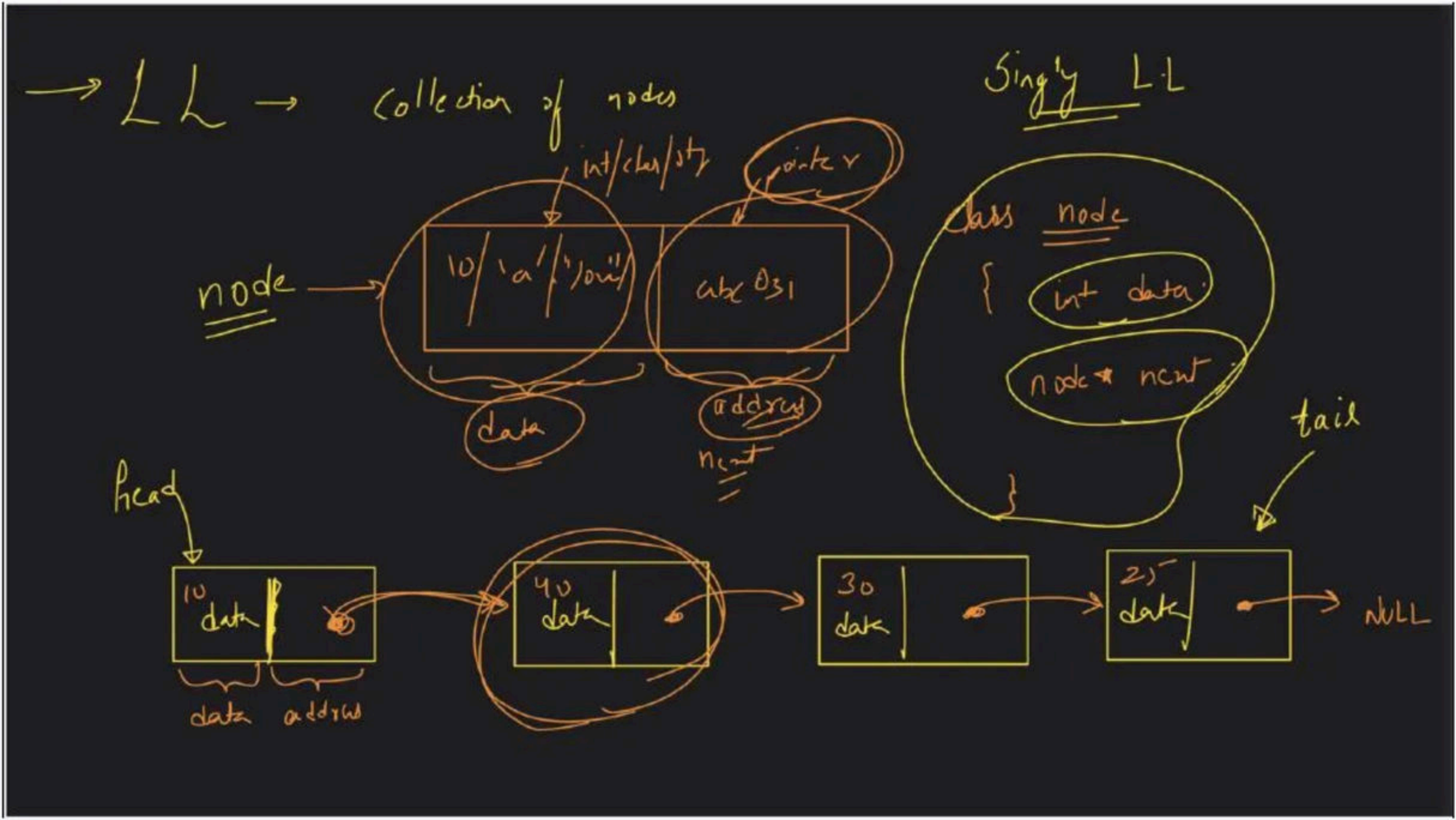


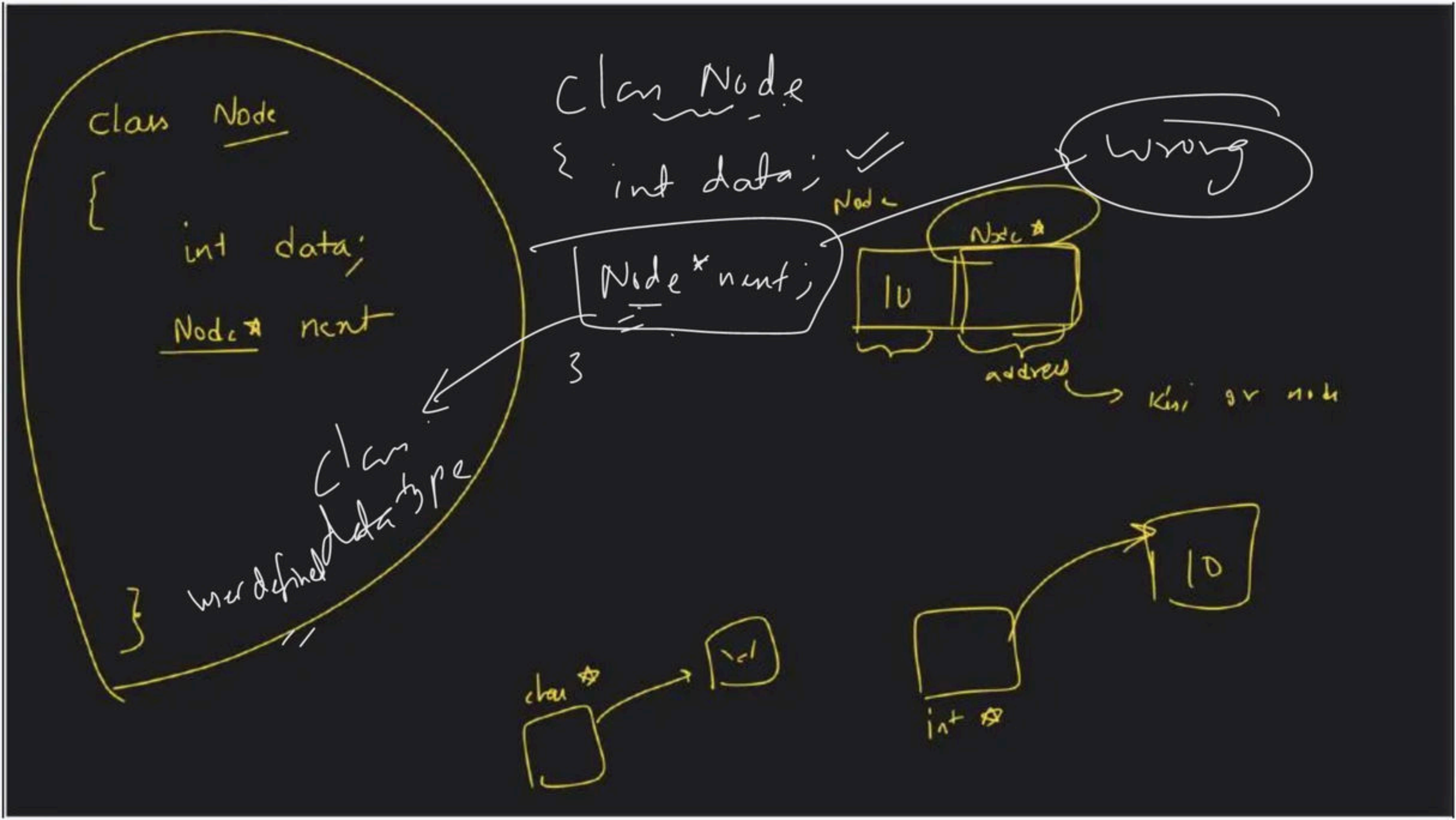
20 byte
array



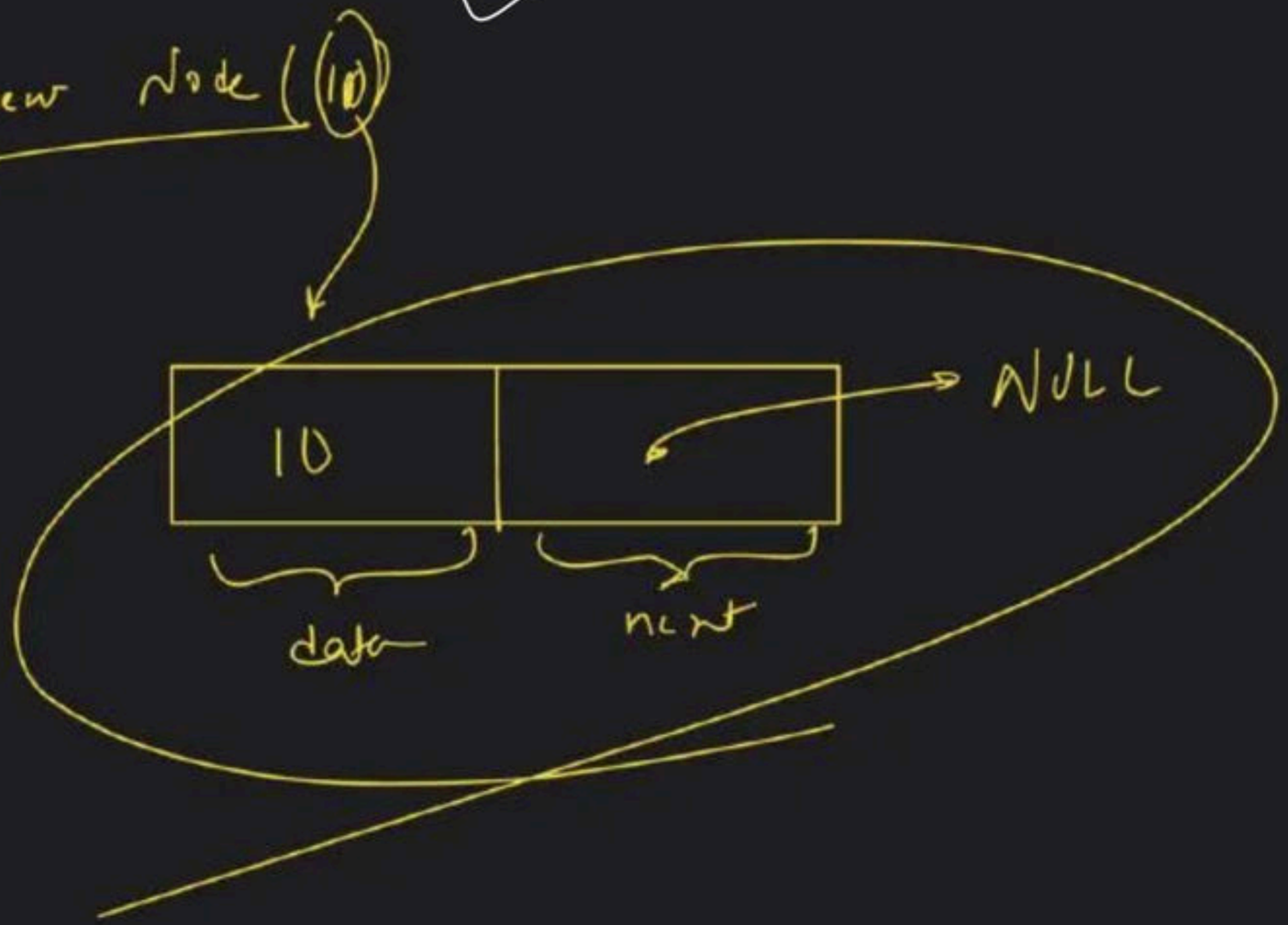
Total free
with (+)

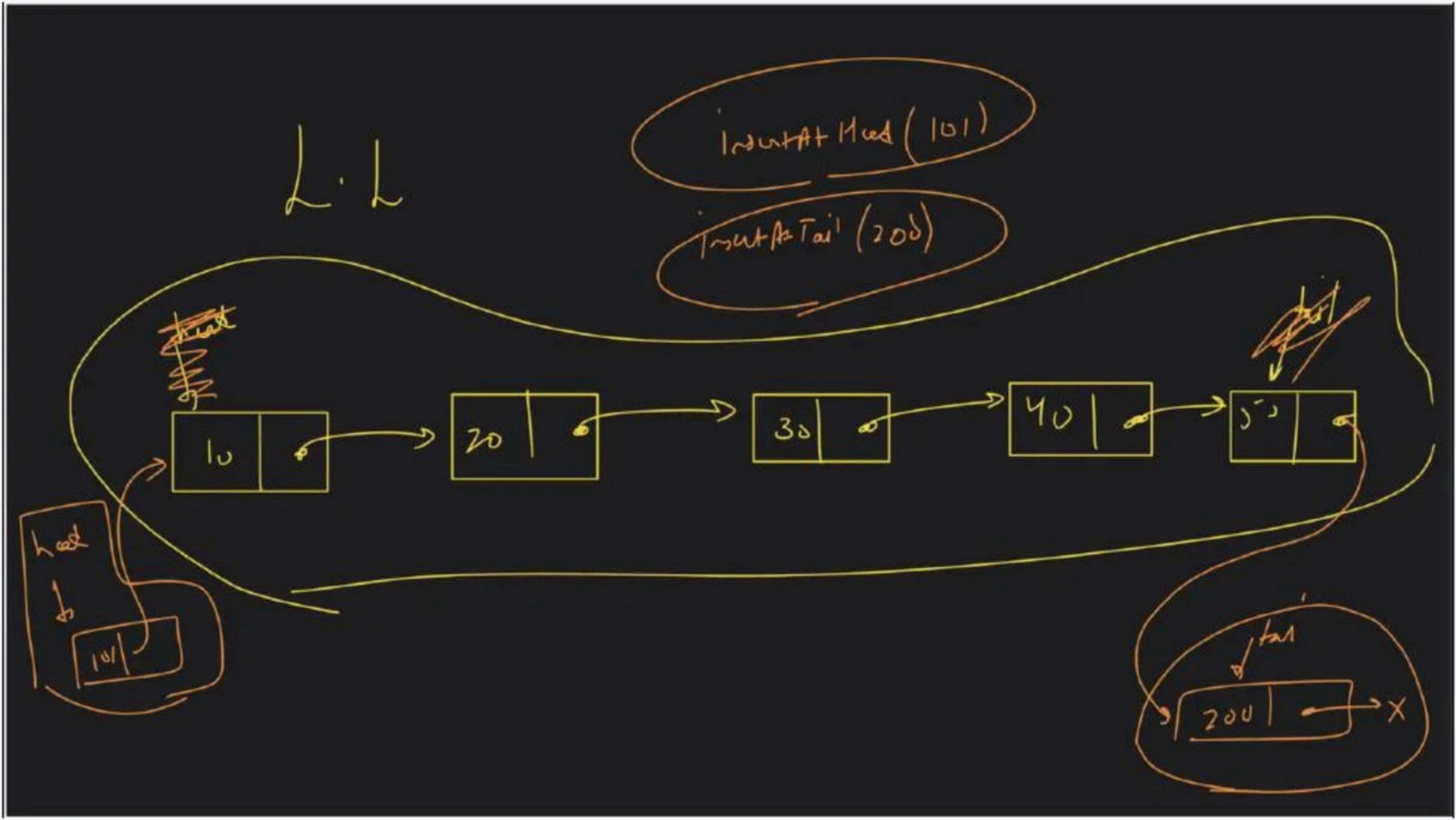



LL → collection of nodes
Singly L.L
int/char/str
pointer
node
10/'a'/"null"
abc 031
data
address
next
class node
{
int data;
node* next;
}
head
tail
10
data
40
data
30
data
20
data
NULL
data address

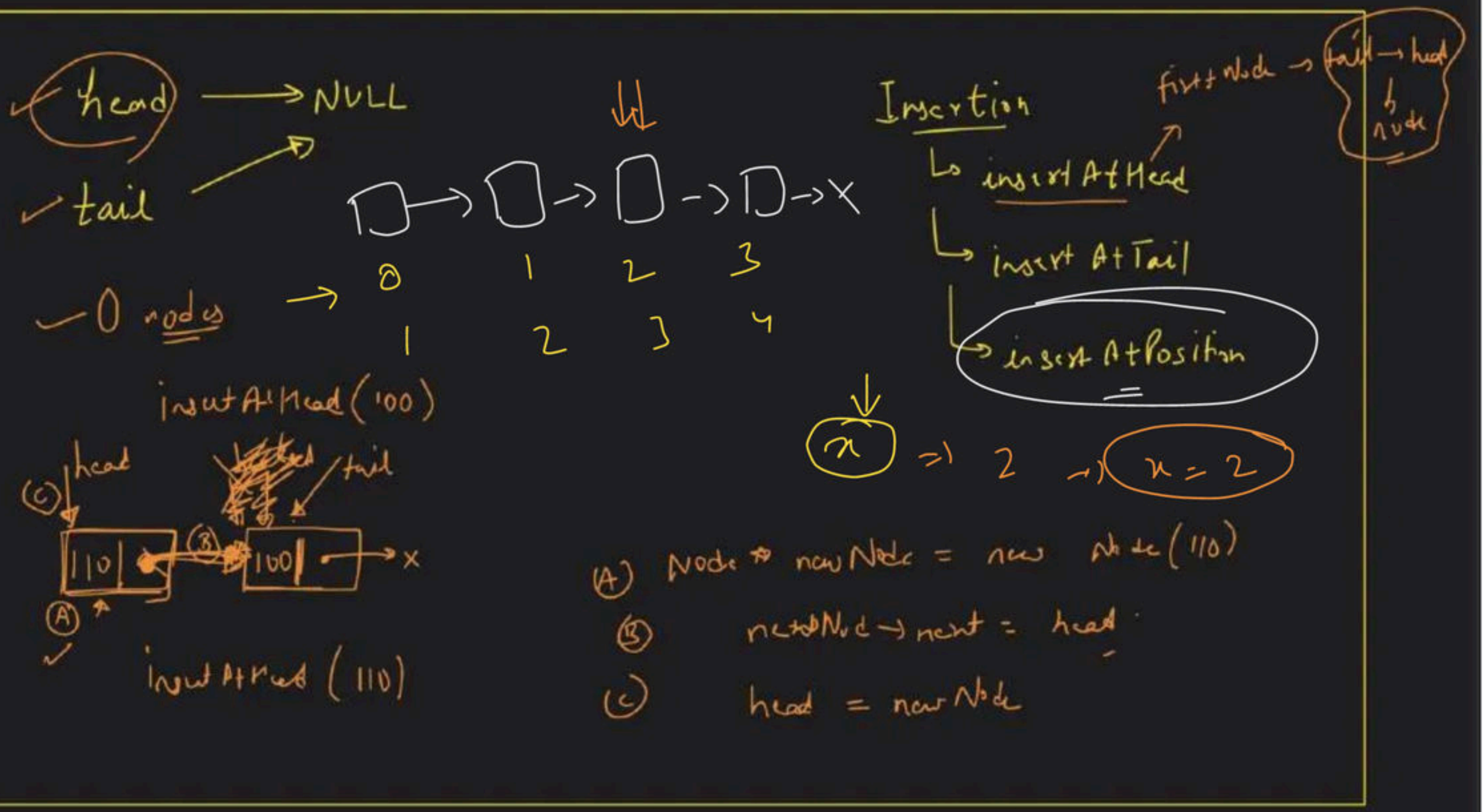

class Node
{
int data;
Node* next
}
user defined data type
class Node
{ int data;
Node* next;
}
Node
Node*
10
address
char*
int*
10

Node * first = new Node(10);

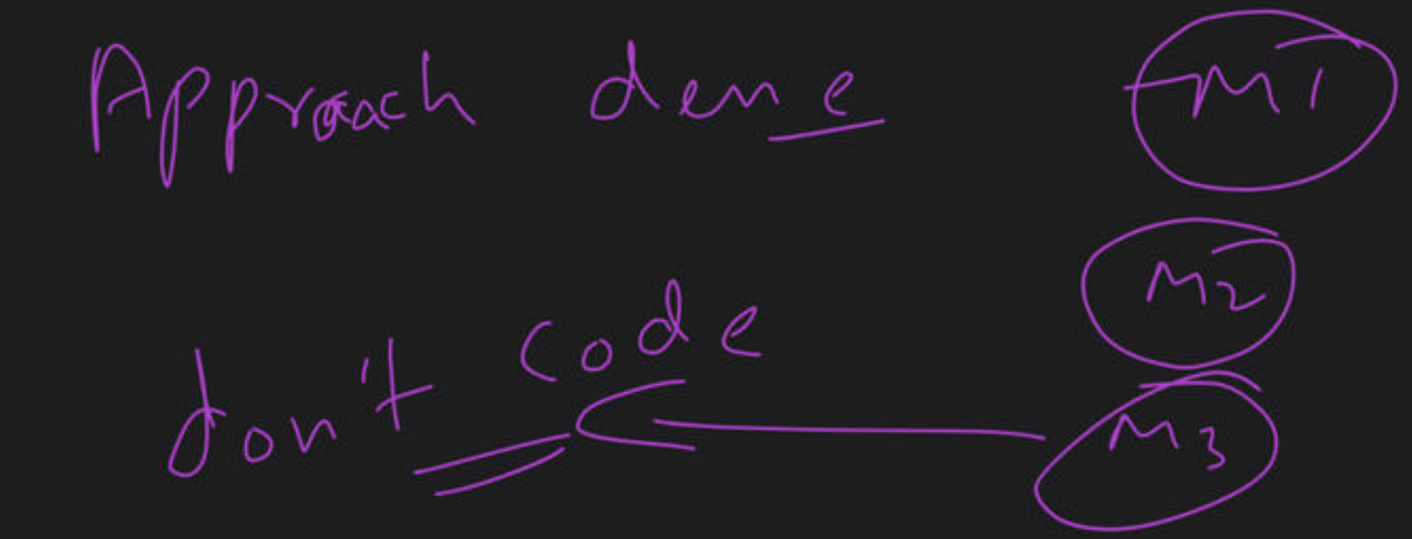



L.L
InsertAtHead(101)
InsertAtTail(200)
head
10
20
30
40
50
tail
head
101
tail
200
X

L.L is Hindi

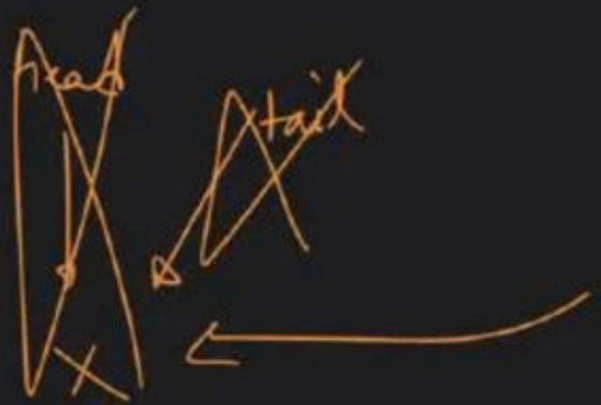

head
NULL
tail
0 nodes
0 1 2 3
1 2 3 4
Insertion
insert At Head
insert At Tail
insert At Position
first Node → tail → head → node
insert At Head (100)
head
tail
110
100
x
A
B
C
insert At Head (110)
n
=> 2
n = 2
(A) Node * newNode = new Node (110)
(B) newNode → next = head
(C) head = newNode

1. Problem statement
2. Approach dene



don't code

3. Ask code krdu konsi approach

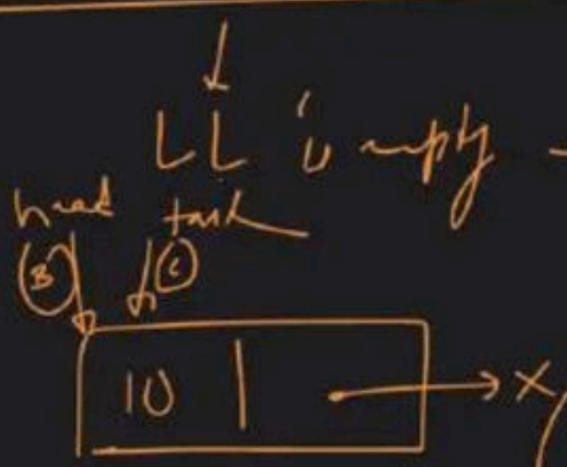


insertAtHead (10)

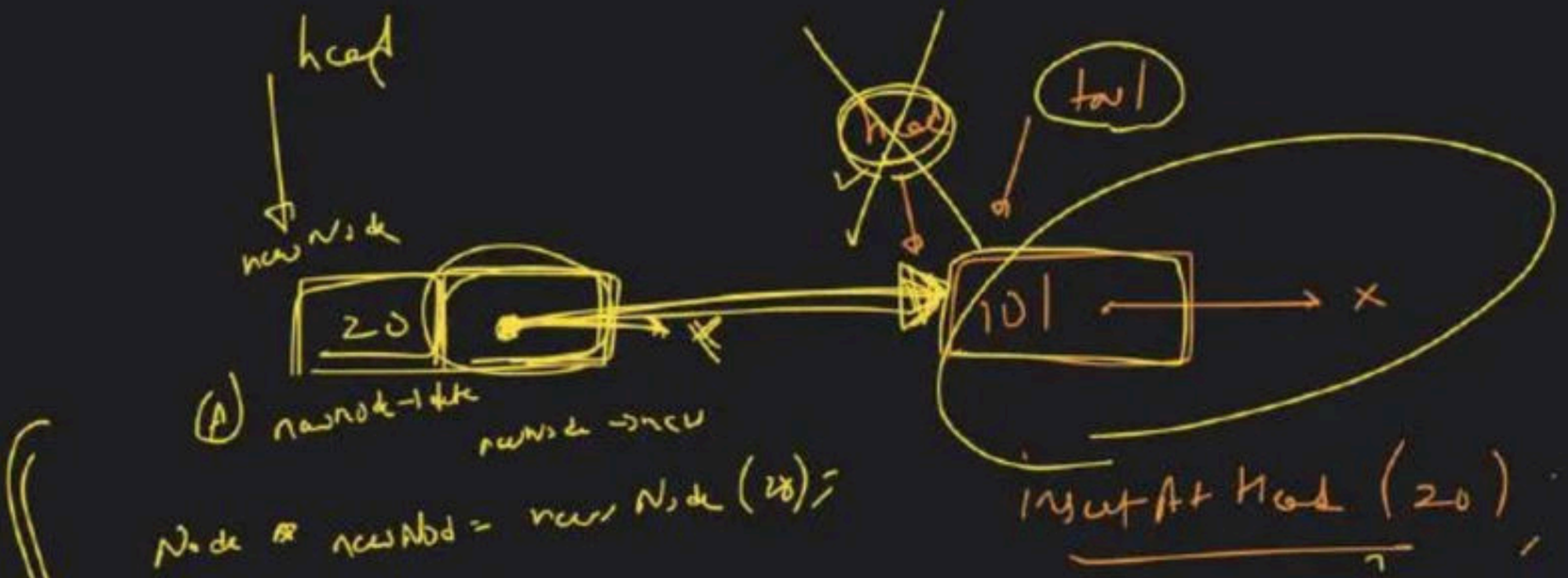
if (head == NULL && tail == NULL)

↓

LL is empty → first Node




```
Node * newNode = new Node(value);
head = newNode;
tail = newNode
```



(A) newNode->data = 20;

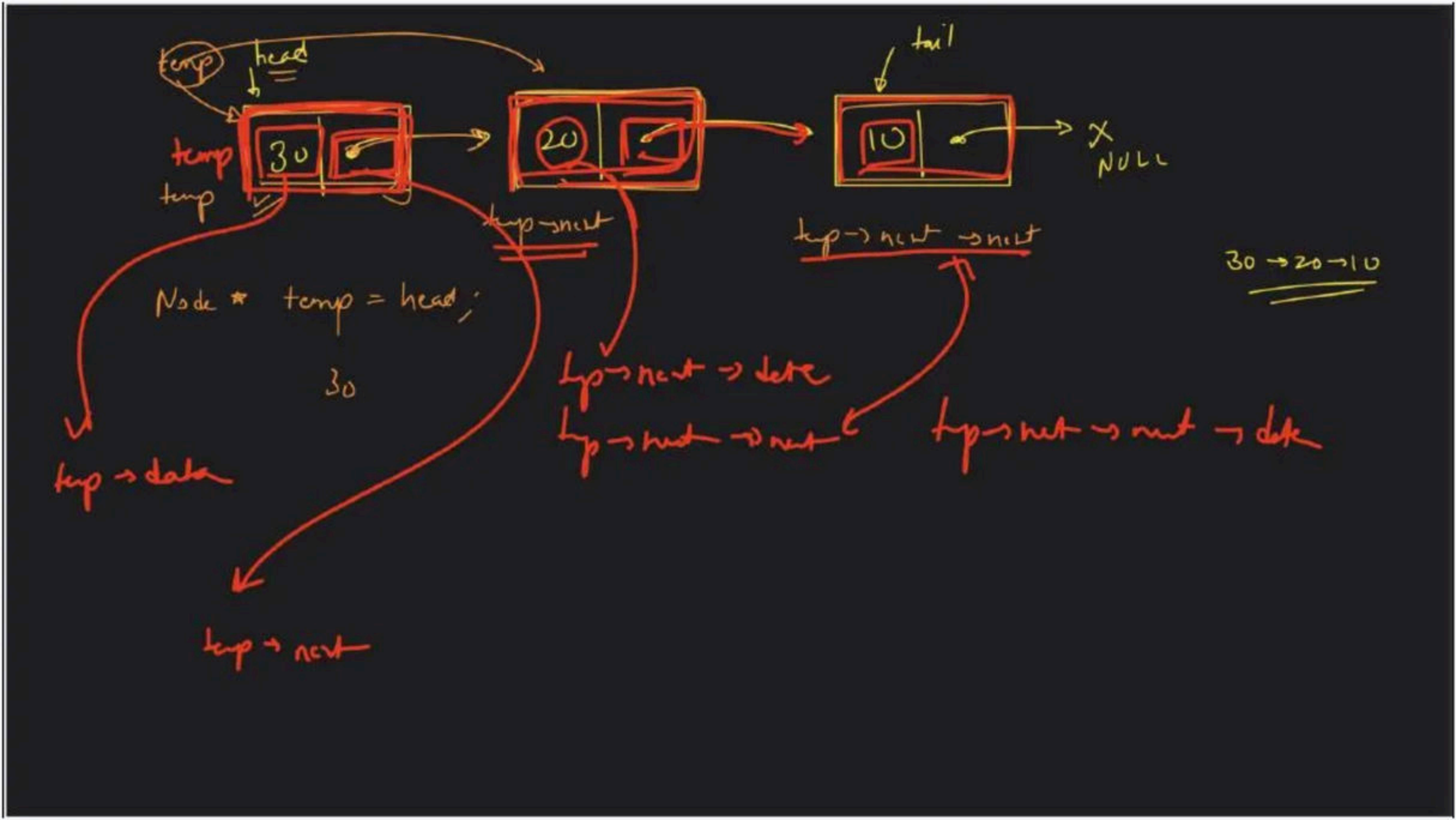
newNode->next = head;

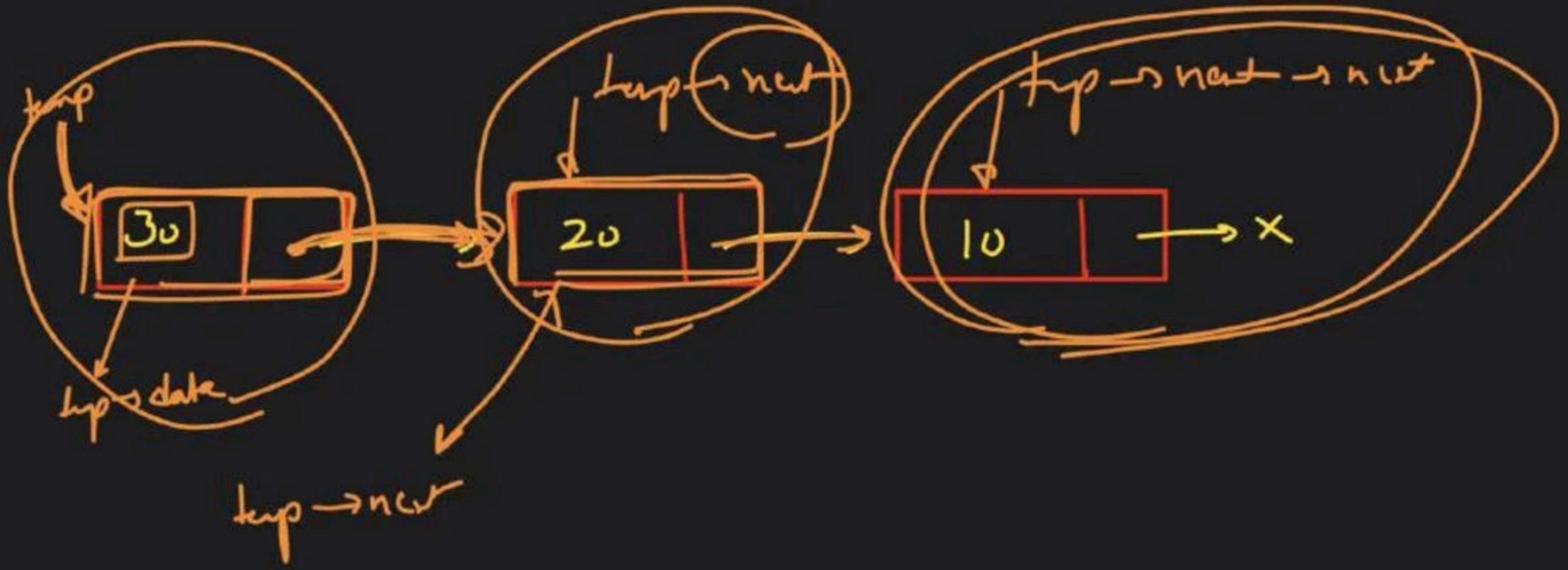
Node * newNod = newNode(20);

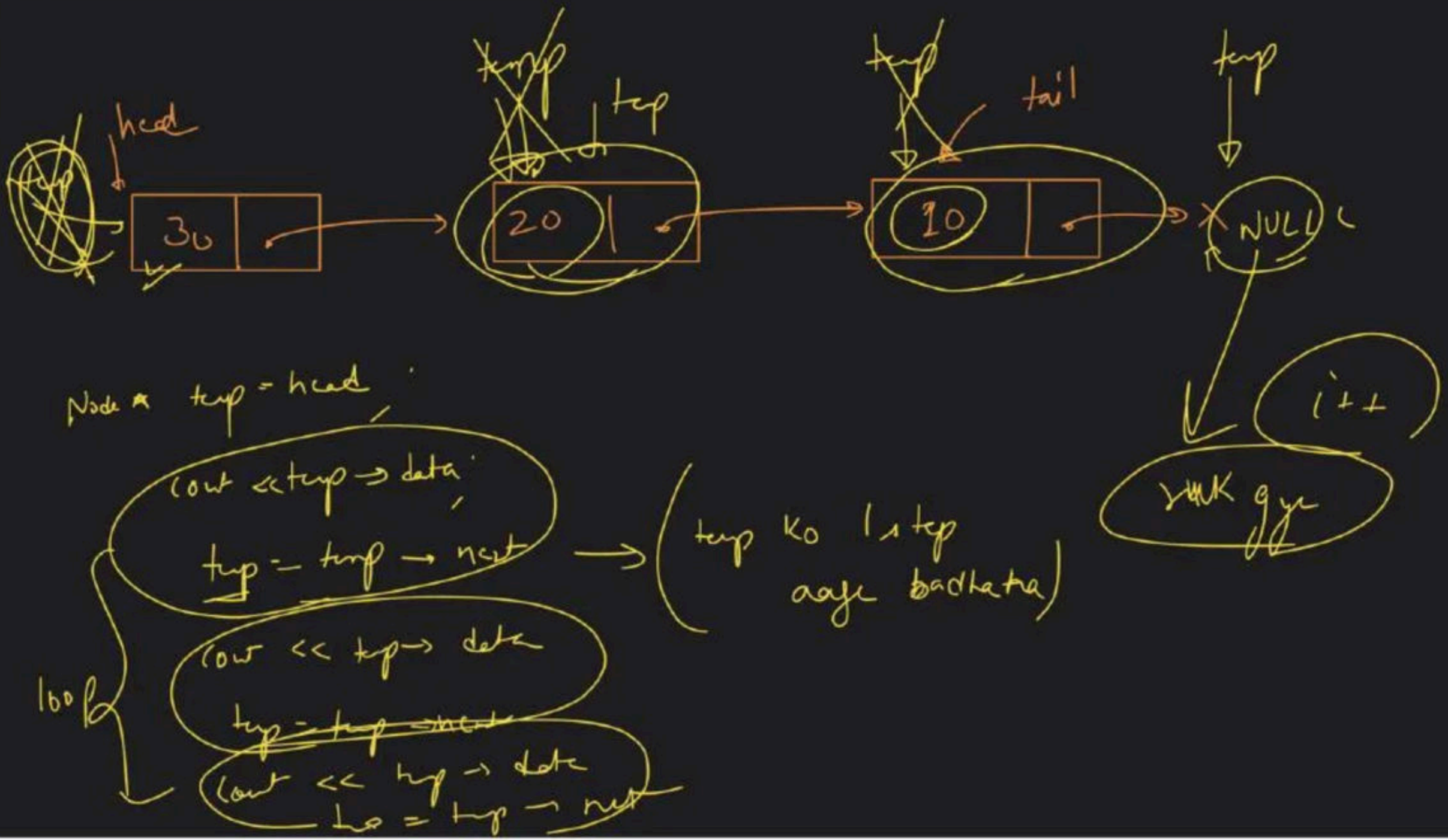
InsertAtHead(20)

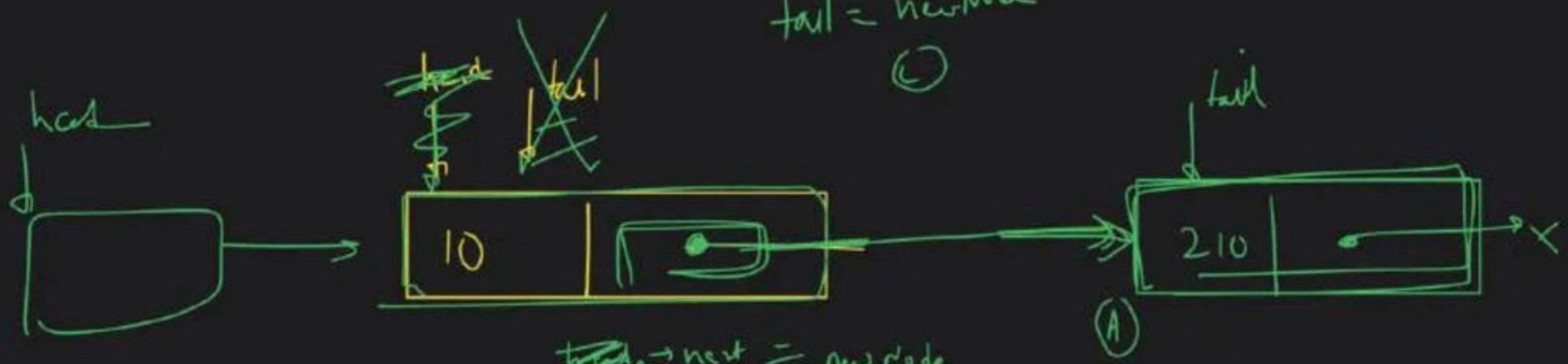
(B) newNode -> next = head

(C) head = newNode


temp
head
tail
30
20
10
X
NULL
temp
temp
temp->next
temp->next->next
30->20->10
Node * temp = head;
30
temp->data
temp->next->delete
temp->next->next
temp->next->next->data
temp->next


temp
30
temp->data
temp->next
20
temp->next
temp->next->next
10
X

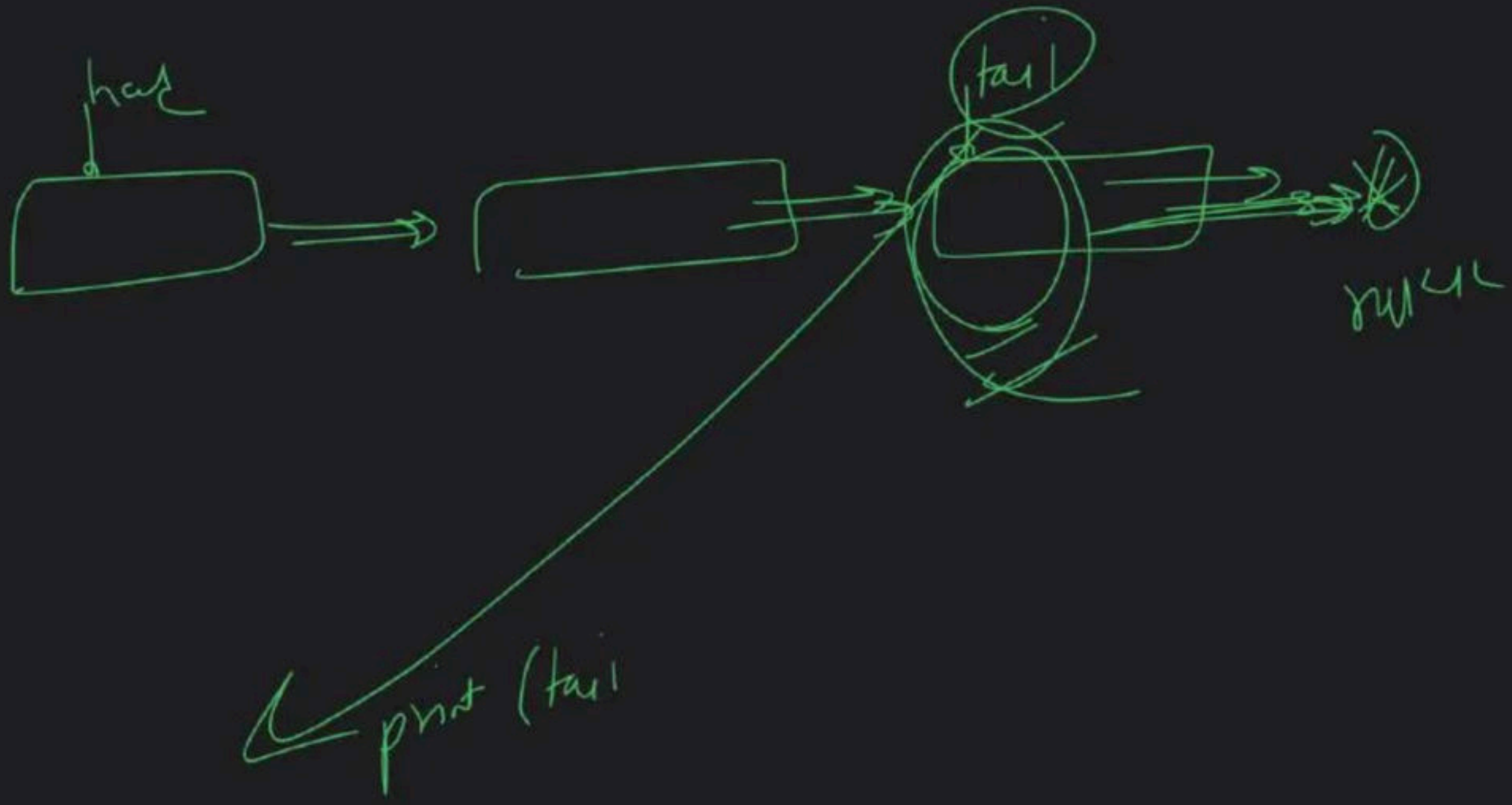

head
temp
temp
tail
temp
30
20
10
NULL
Node * temp = head
cout << temp -> data
temp = temp -> next
temp ko 1 step aage badhata
loop
cout << temp -> data
temp = temp -> next
cout << temp -> data
temp = temp -> next
i++
yahi ruk jao

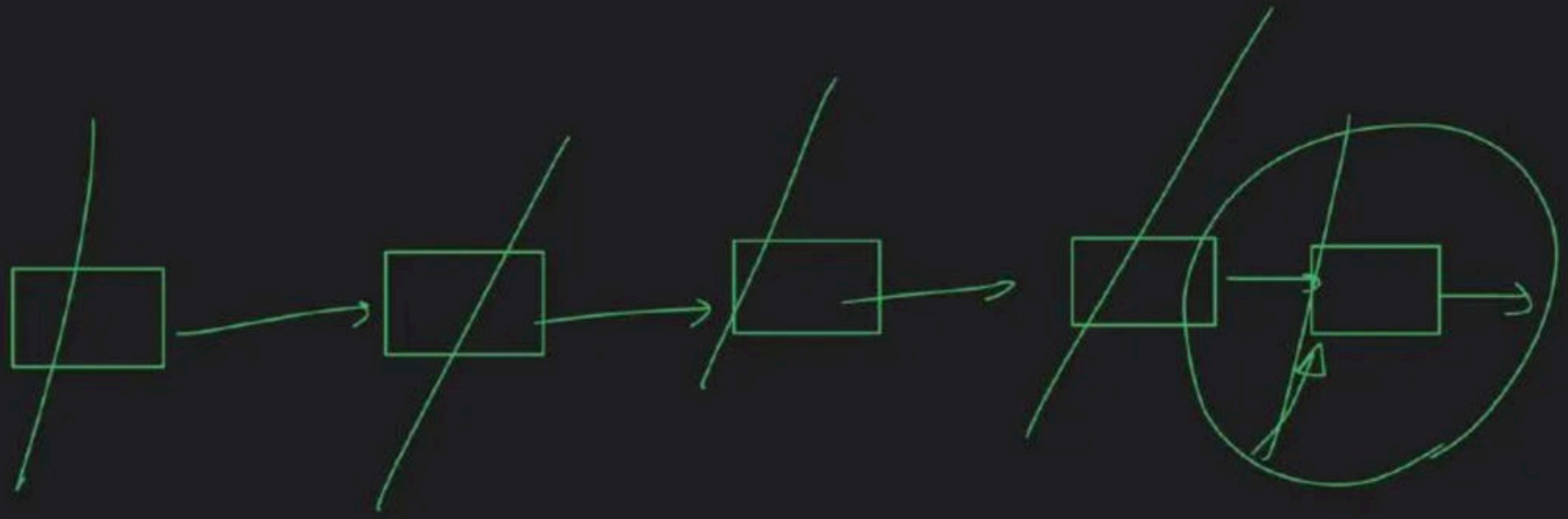


Node * newNode =
new Node(210);

insertAtTail(210);

2 min
Break


head
tail
print (tail



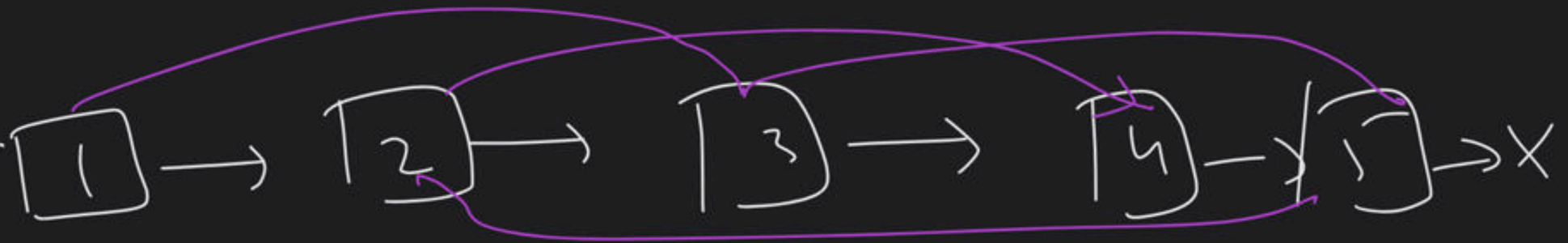
array → O(1)

LL → O(n)

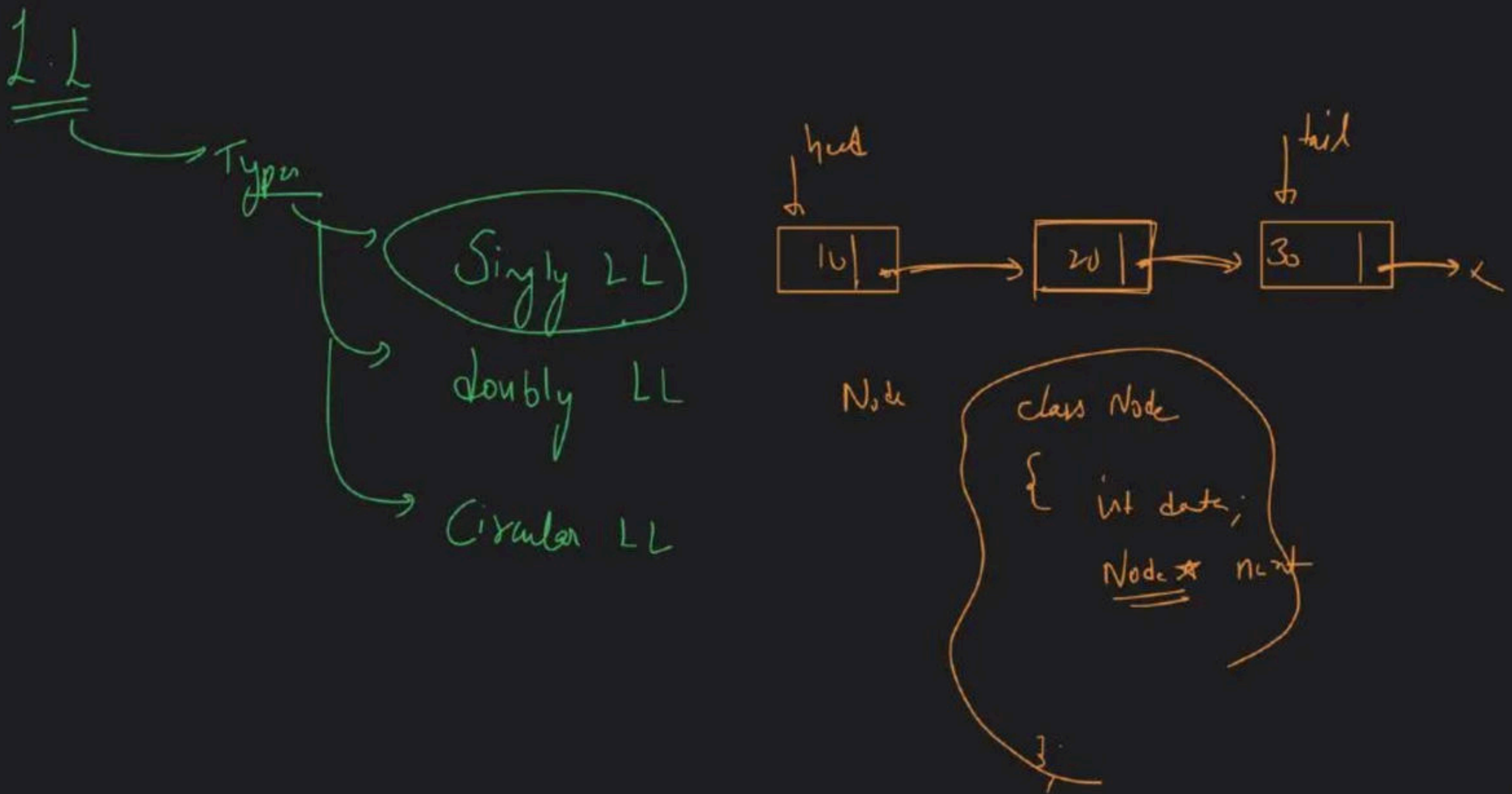
=> All Dry Runs =>

=>

2) alternate node club



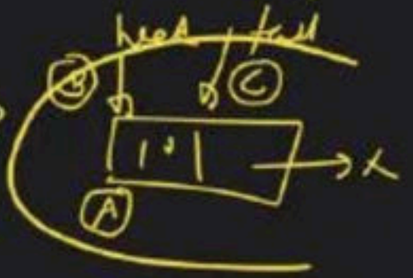
=>

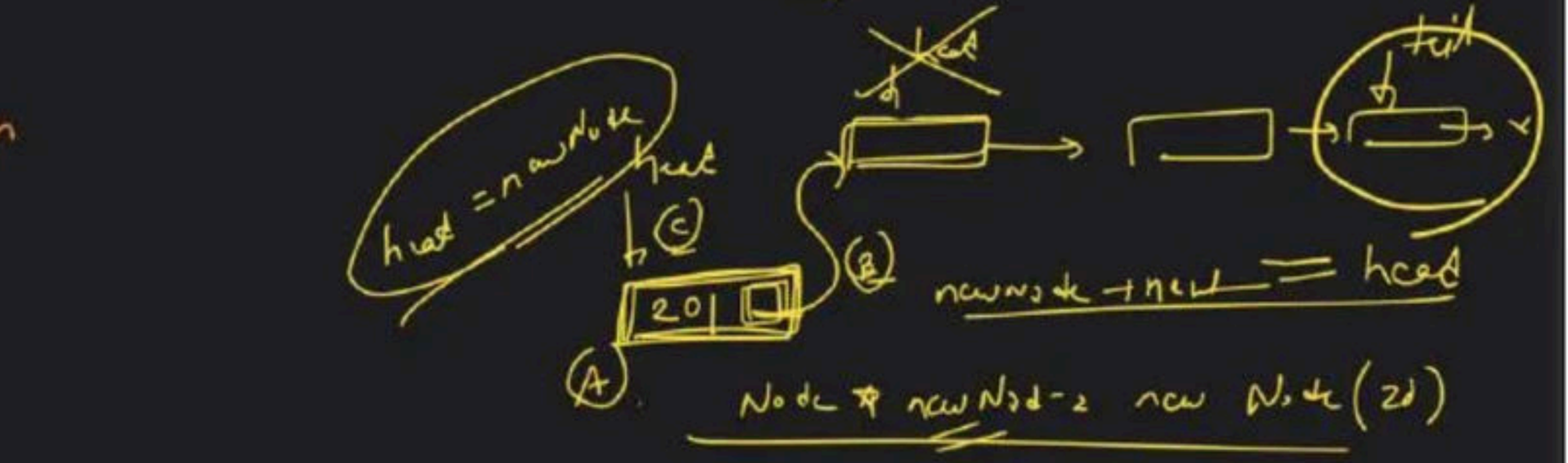

LL
Types
Singly LL
doubly LL
Circular LL
head
tail
10
20
30
Node
class Node
{ int data;
Node* next;
}

2 cases

Insertion

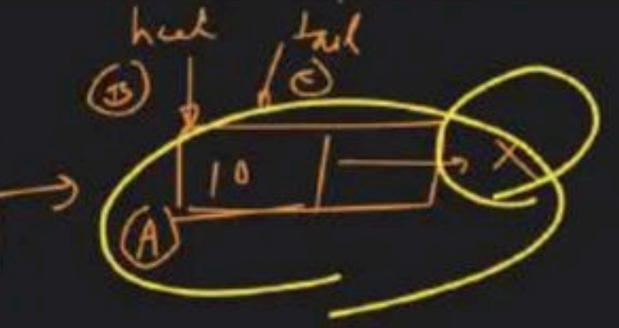
- insertAtHead
  - LL is empty → if (head = NULL, tail = NULL)
  - LL is not empty → else
- insertAtTail
- insertAtPosition



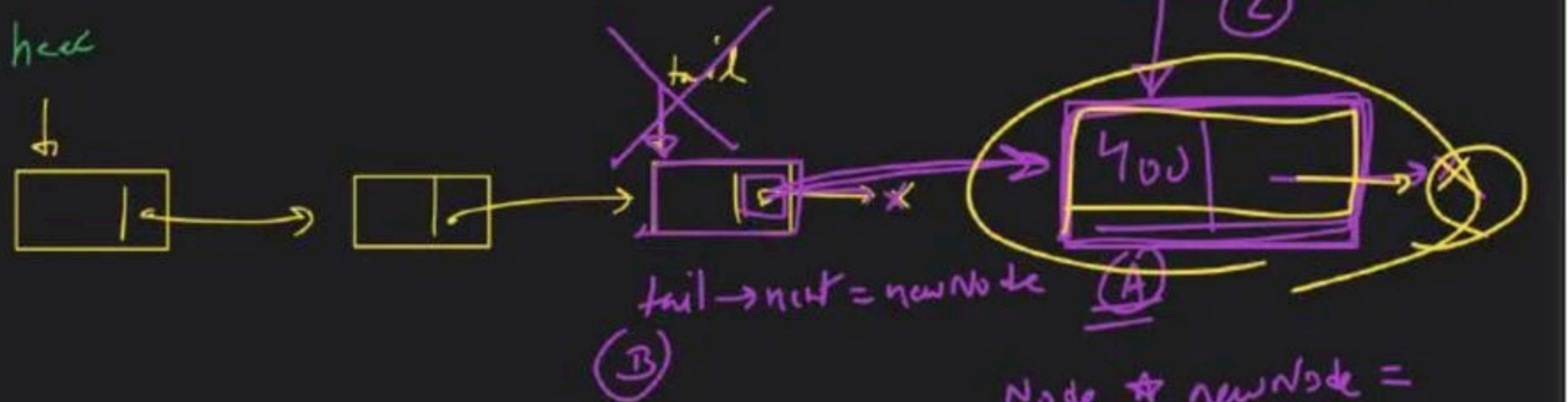


2 Cases

↳ insertAtTail → LL is empty → if (head = NULL, tail = NULL)

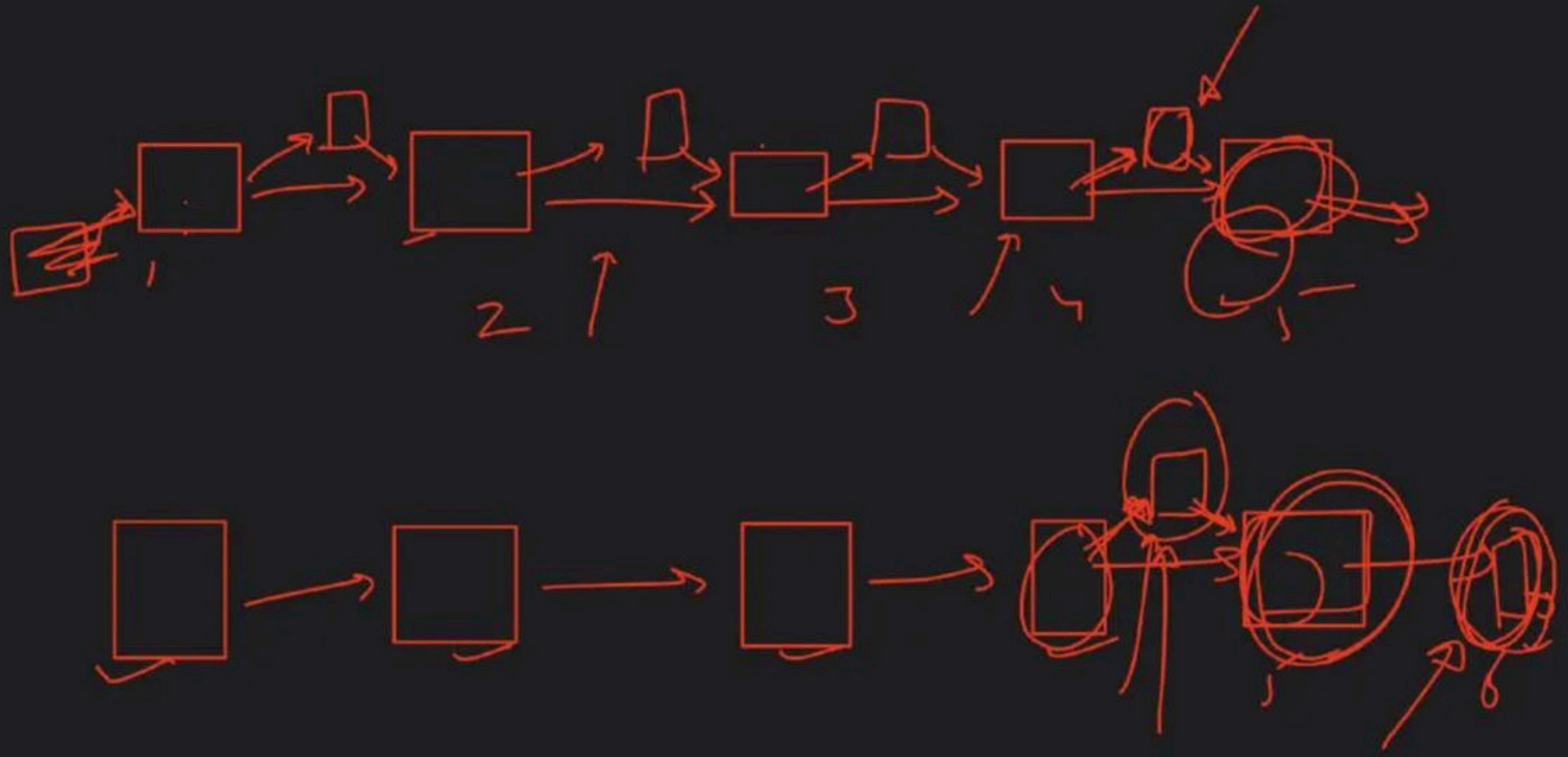


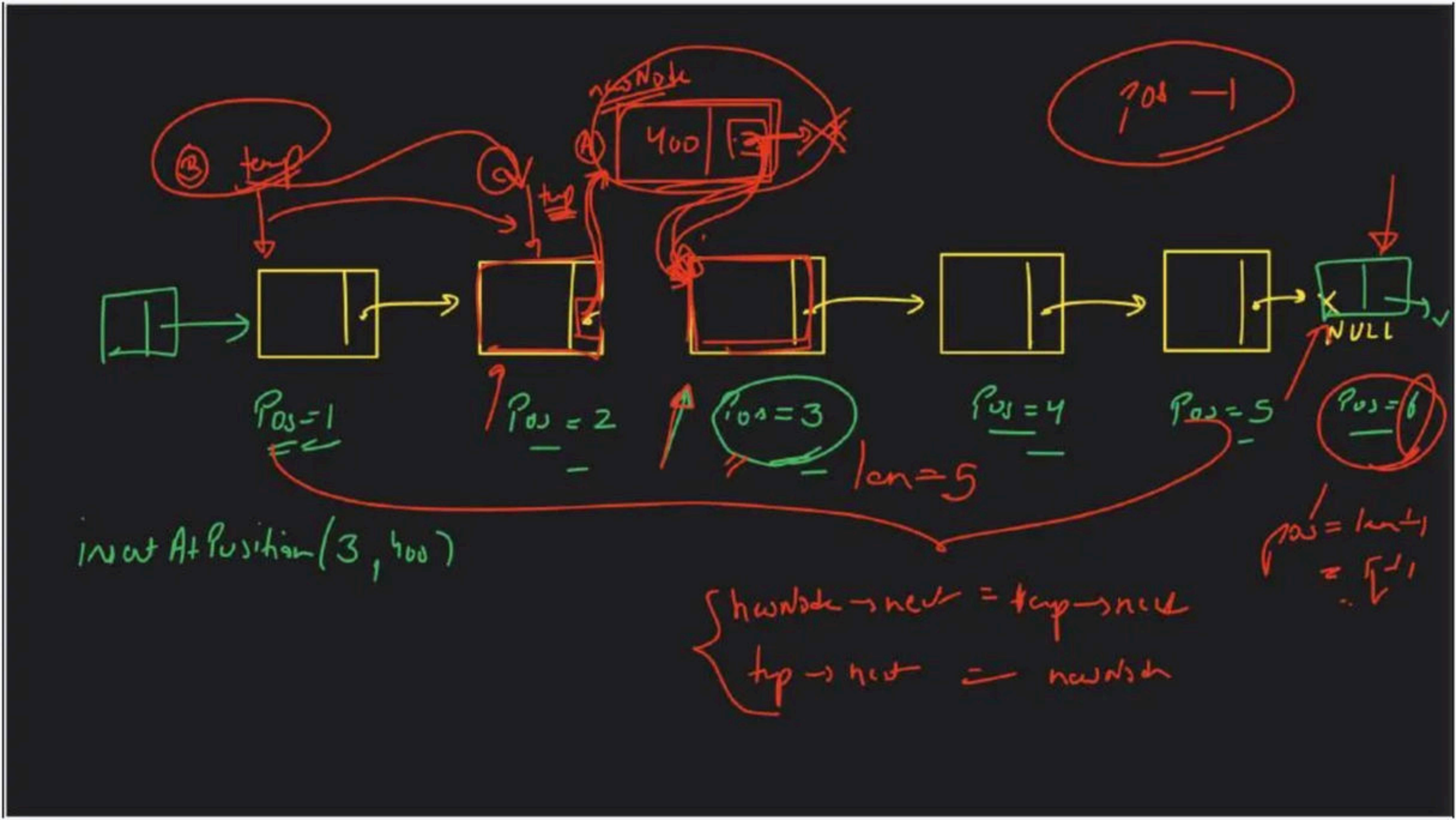
LL is not empty

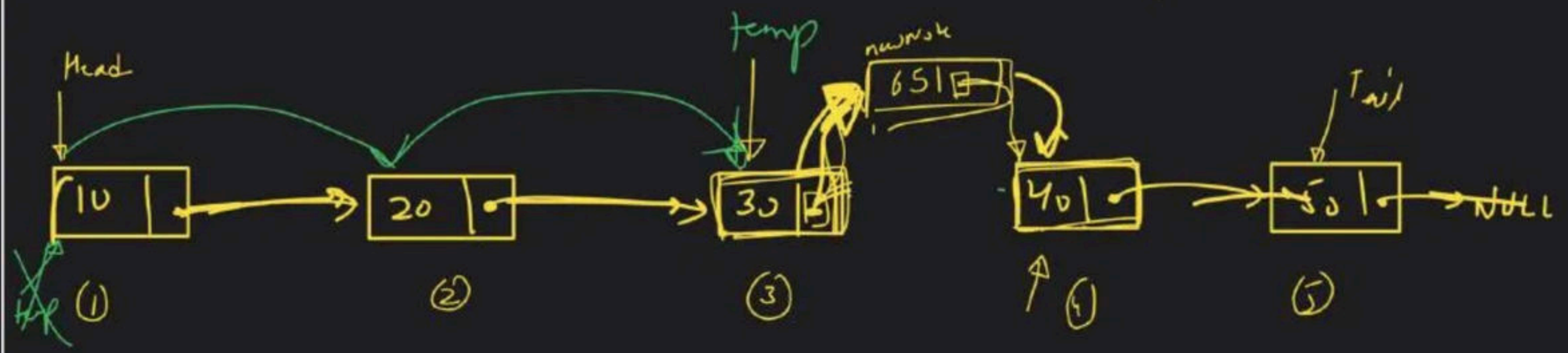


insertAtTail(You)

Node * newNode = new Node(You);


1
2
3

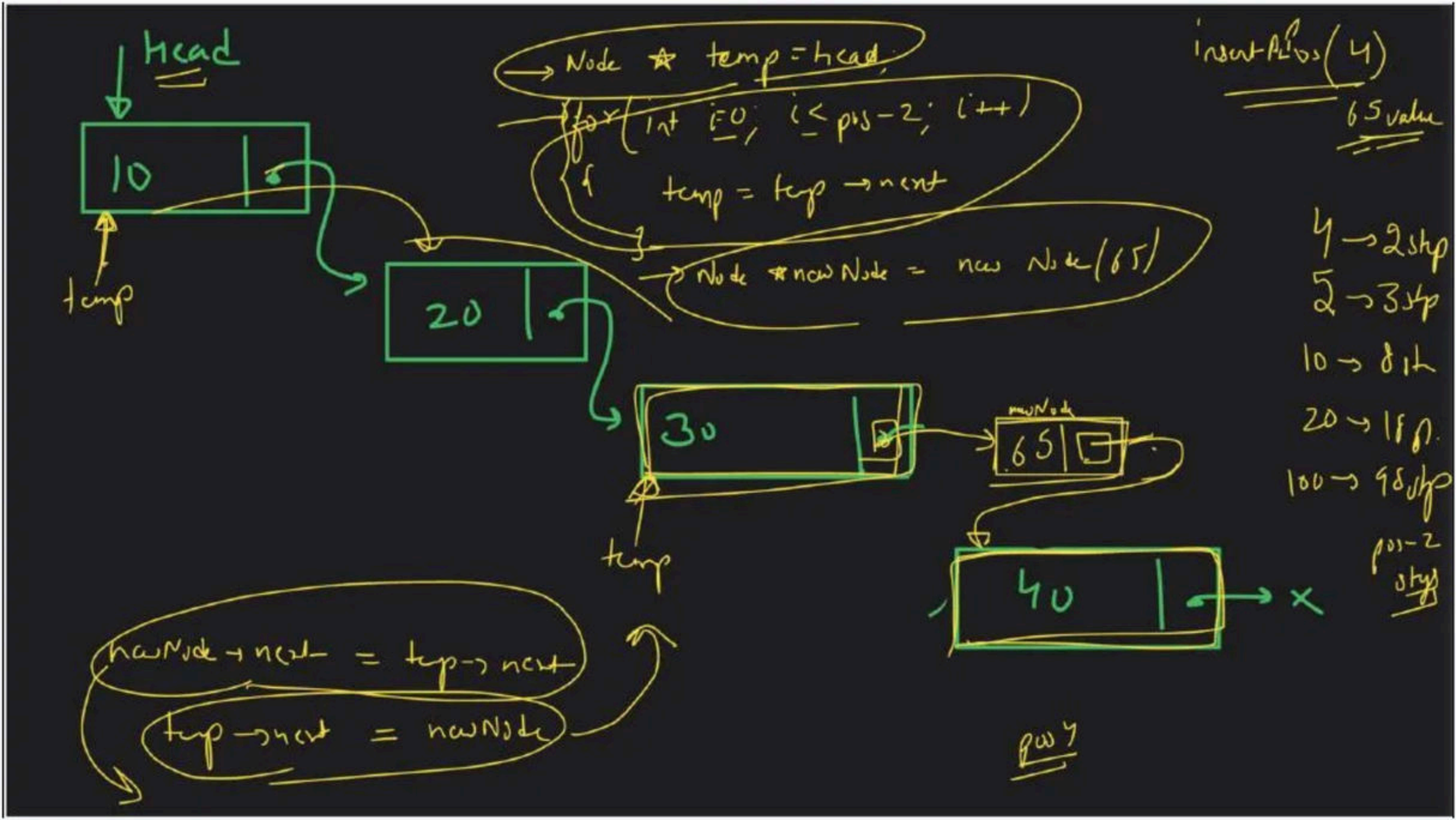

newNode
400
temp
temp
pos = 1
pos
NULL
Pos = 1
Pos = 2
Pos = 3
Pos = 4
Pos = 5
Pos = 6
len = 5
InsertAtPosition(3, 400)
pos = len + 1
= 5 + 1
newNode → next = temp → next
temp → next = newNode

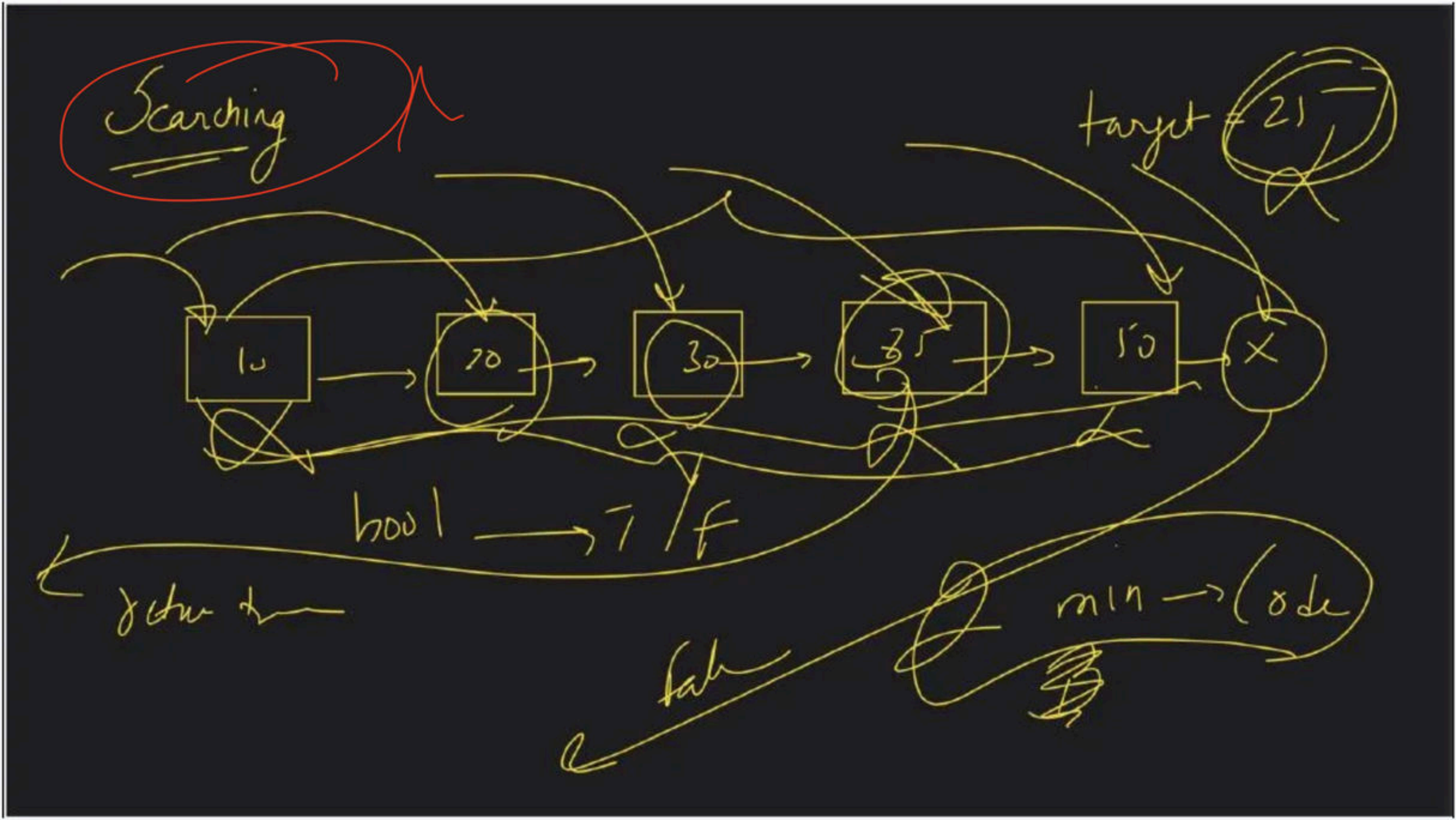


Position - 2

Insert At Position (4, 65, head, tail)

```
Node * temp = head;
for(i=0; i<pos-2; i++)
    temp = temp -> next;
Node * newNode = new Node(15);
newNode -> next = temp -> next;
temp -> next = newNode
```


head
10
20
30
40
65
temp
temp
newNode
Node * temp = head;
for(int i=0; i<=pos-2; i++)
{
temp = temp->next
}
Node * newNode = new Node(65)
newNode->next = temp->next
temp->next = newNode
insertAtPos(4)
65 value
4 → 2 step
5 → 3 step
10 → 8 th
20 → 18 th
100 → 98 step
pos-2 steps
x
Row 4


Searching
target = 21
10
20
30
40
50
X
bool → T / F
return
false
min → node

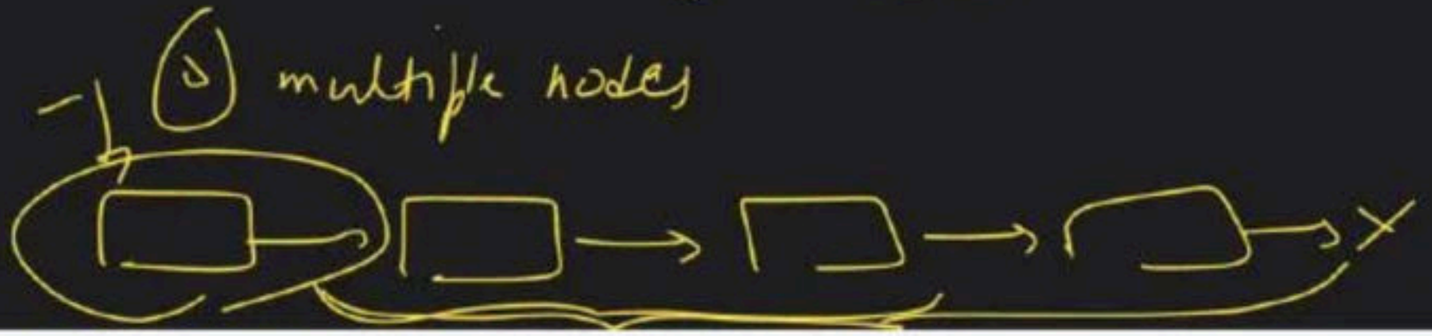
Deletion → cases

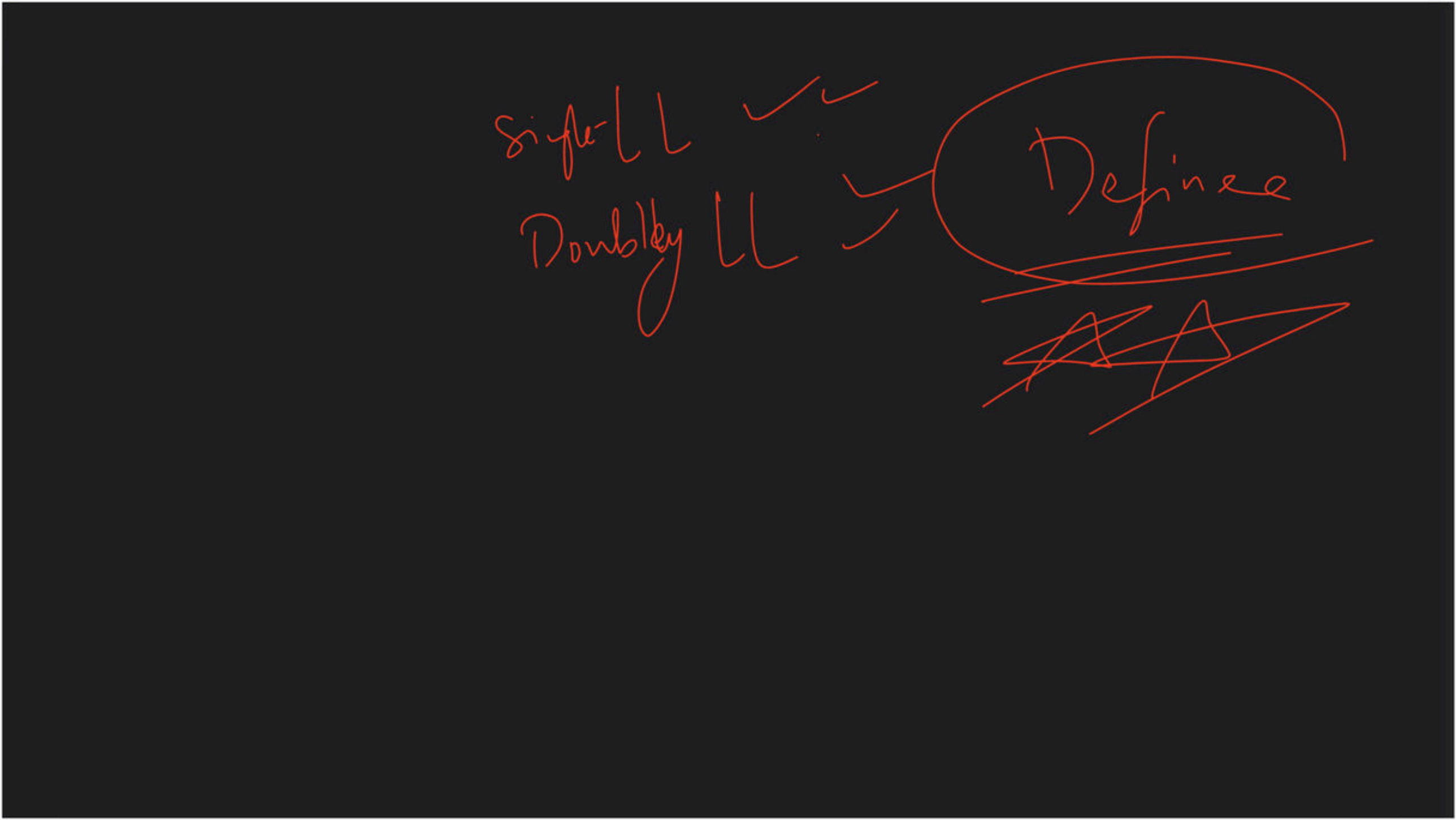
① LL is empty

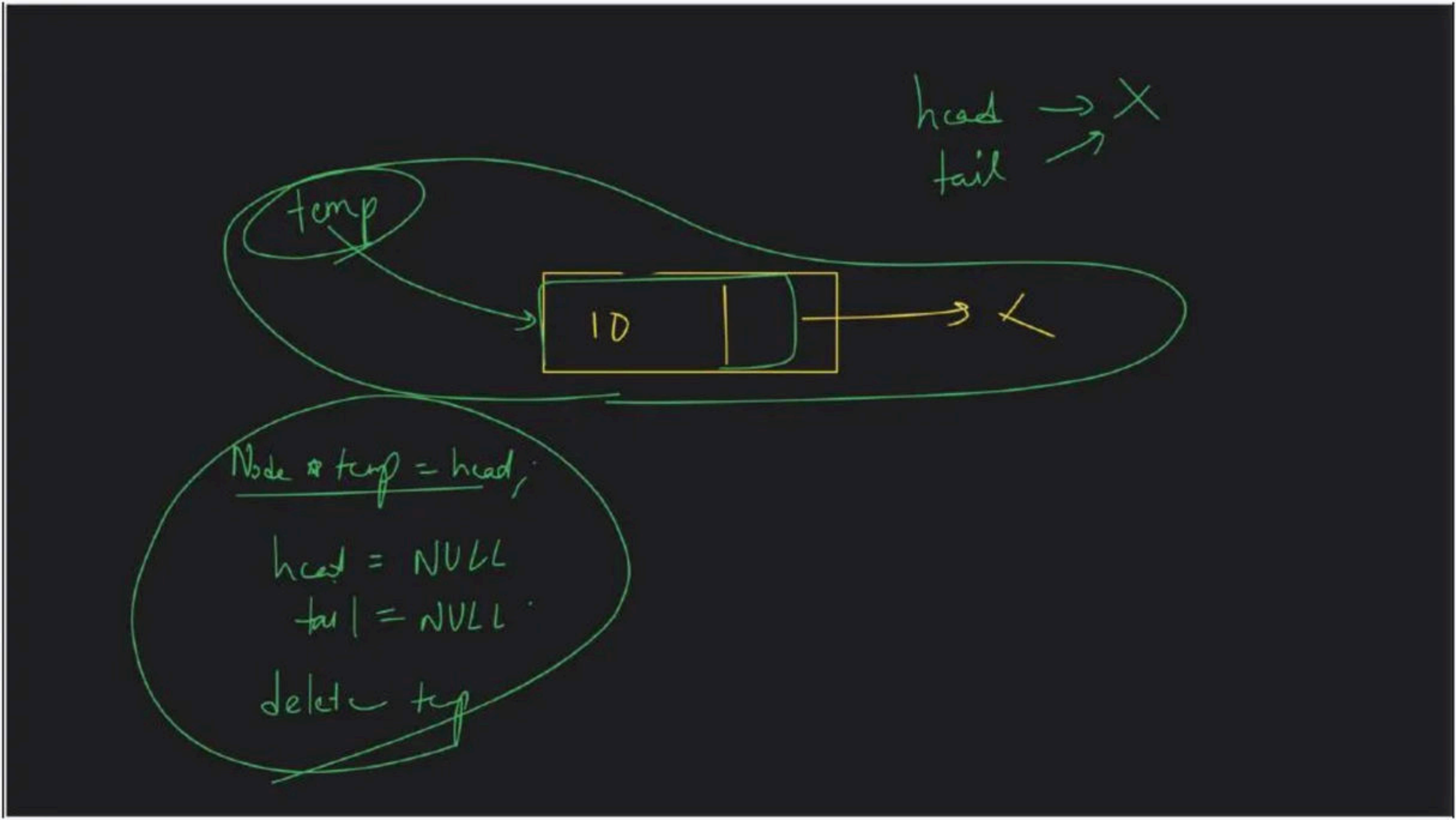
→ cannot delete

② Single node

③ multiple nodes



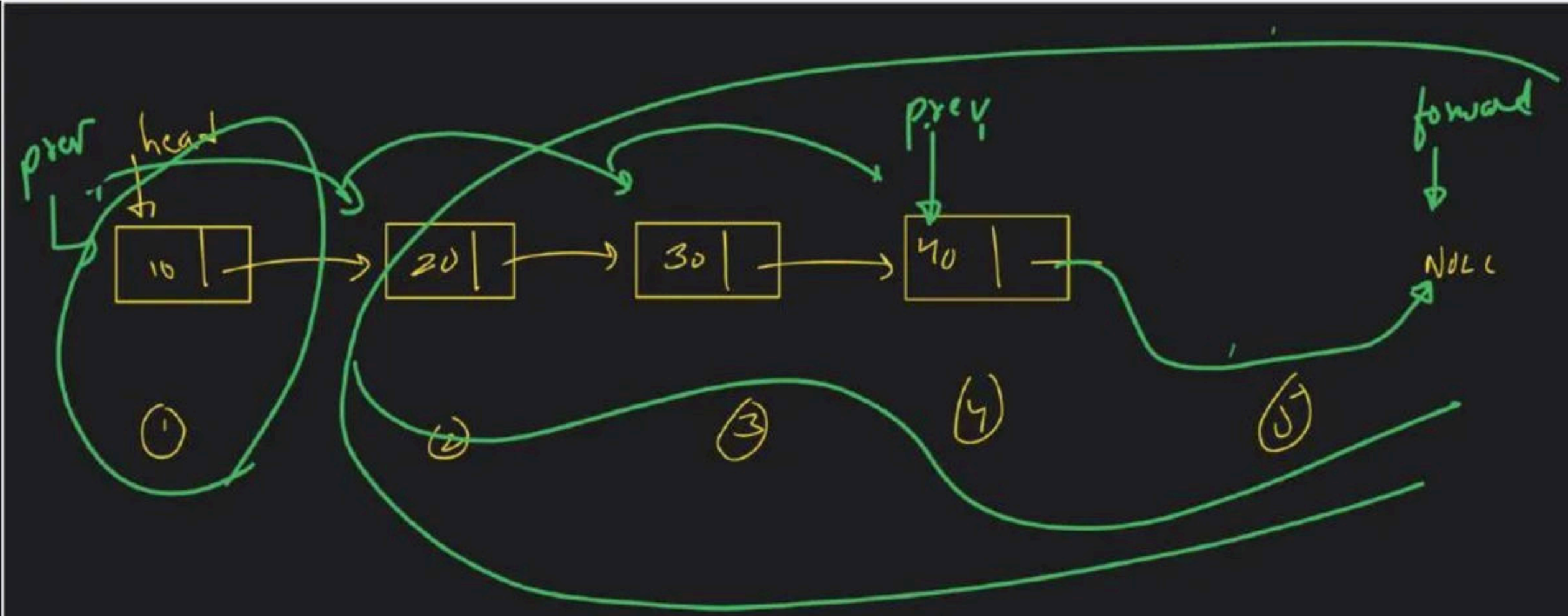

Dolby
Defined


head → X
tail →
temp
10
X
Node * temp = head;
head = NULL
tail = NULL
delete temp

deletePos(3)

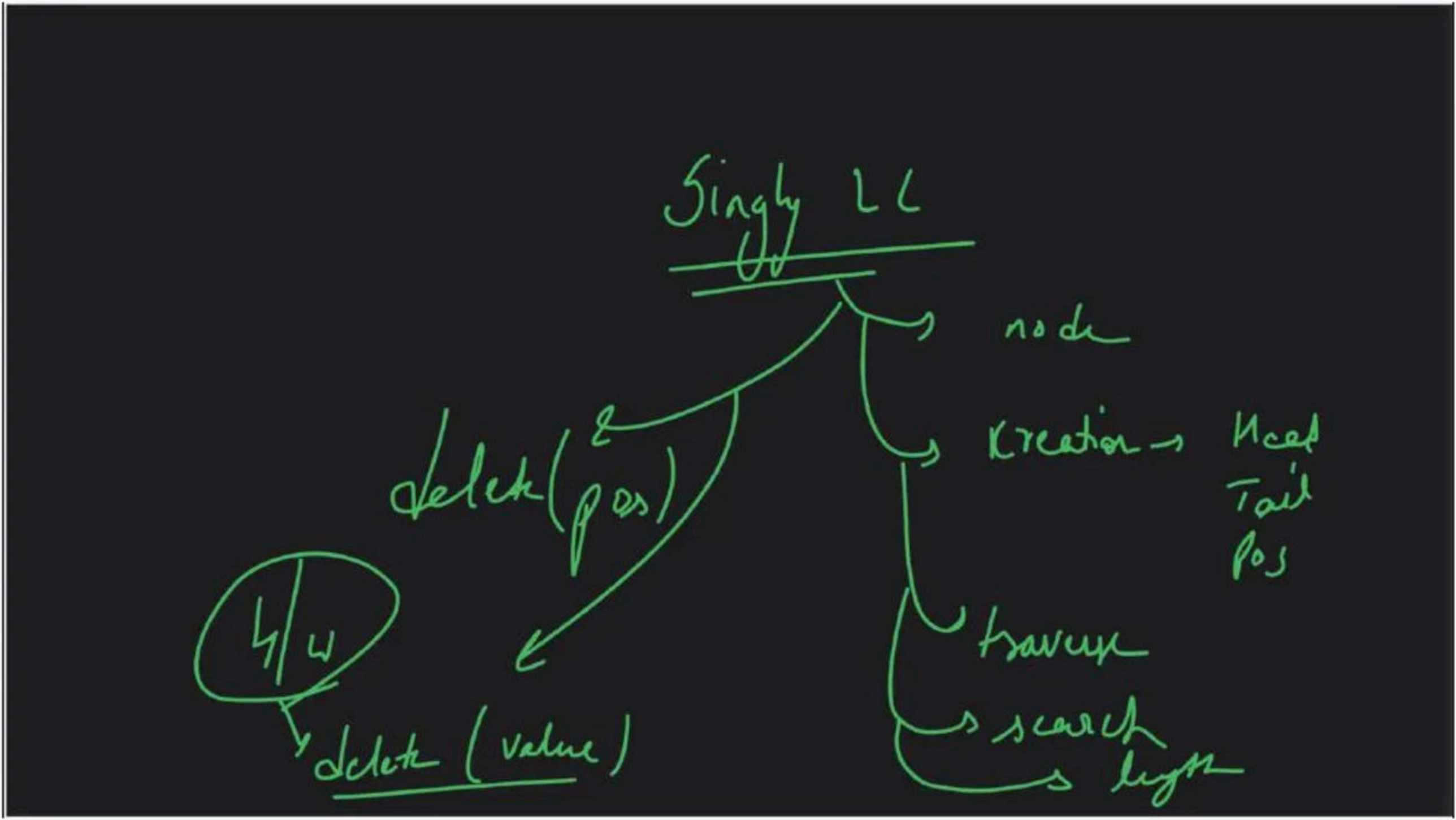
```
Node * prev = head
for (i=0; i<pos-2; i++)
{ prev = prev->next
}
Node * curr = prev->next
Node * forward = curr->next;
```

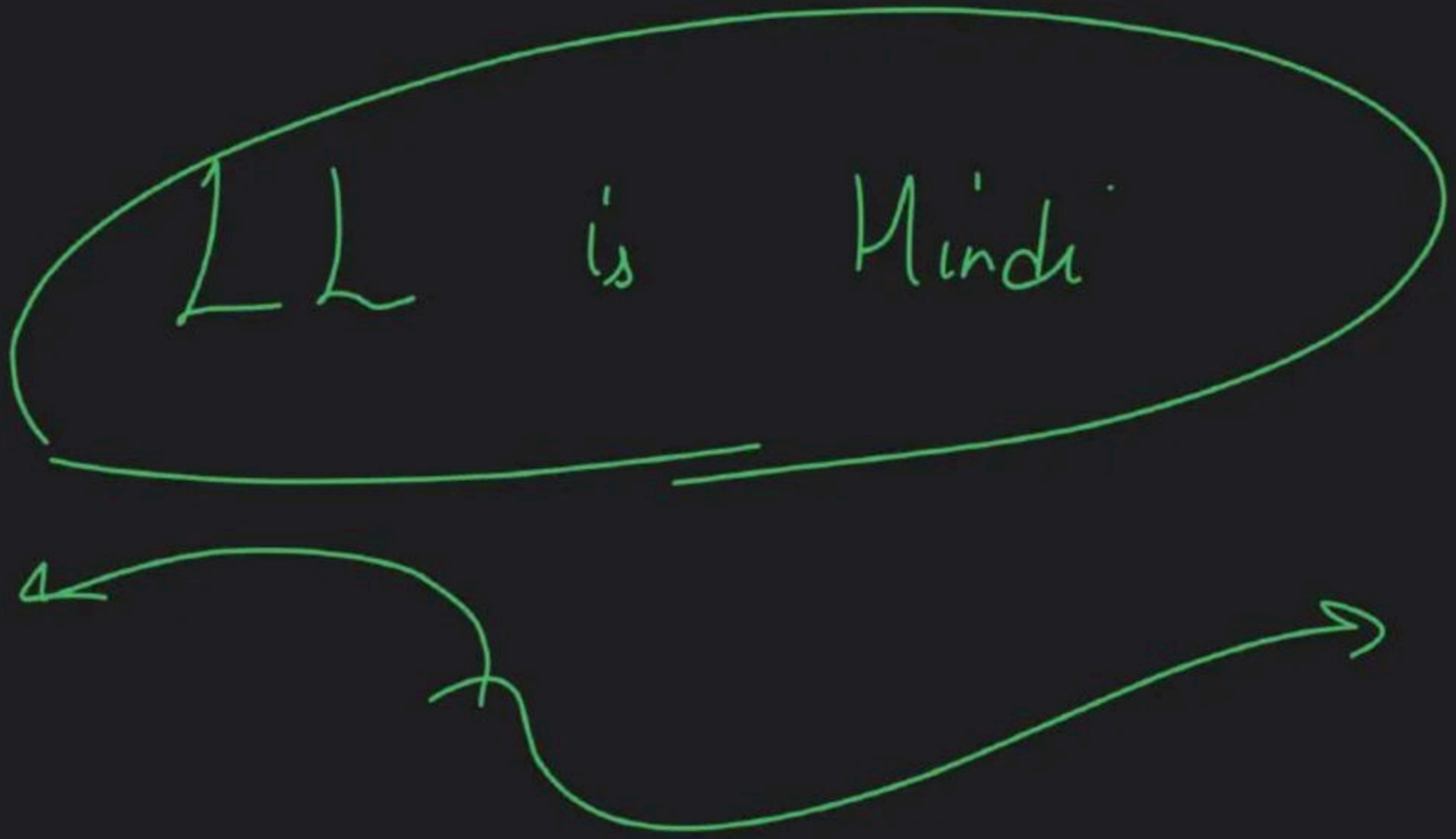
```
prev->next = forward
curr->next = NULL
delete curr;
```

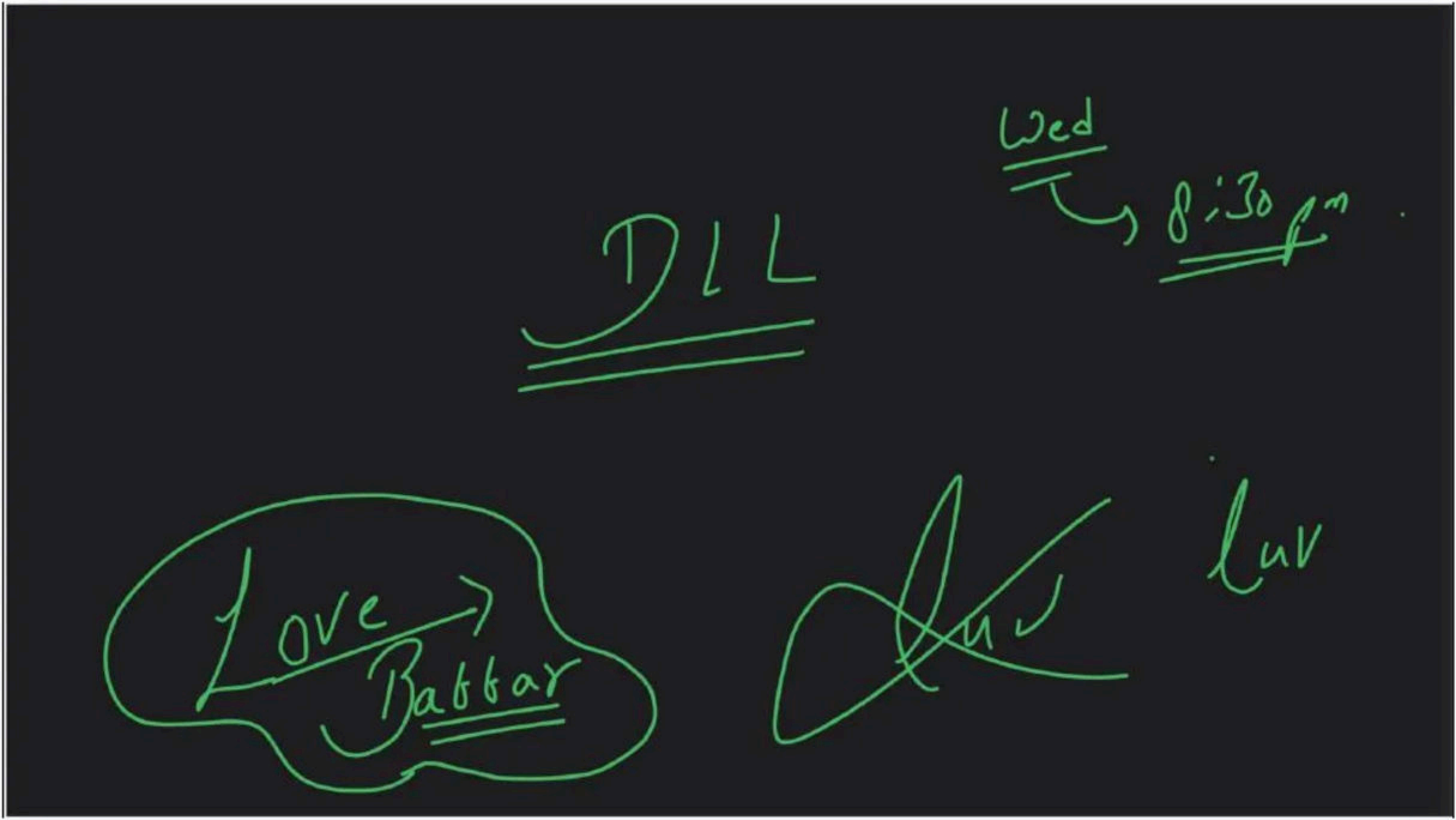


deletePos(5)

pos-2


Singly LL
node
Creation → Head
Tail
Pos
delete(pos)
4/4
delete (value)
traverse
search
display


LL is Hindi

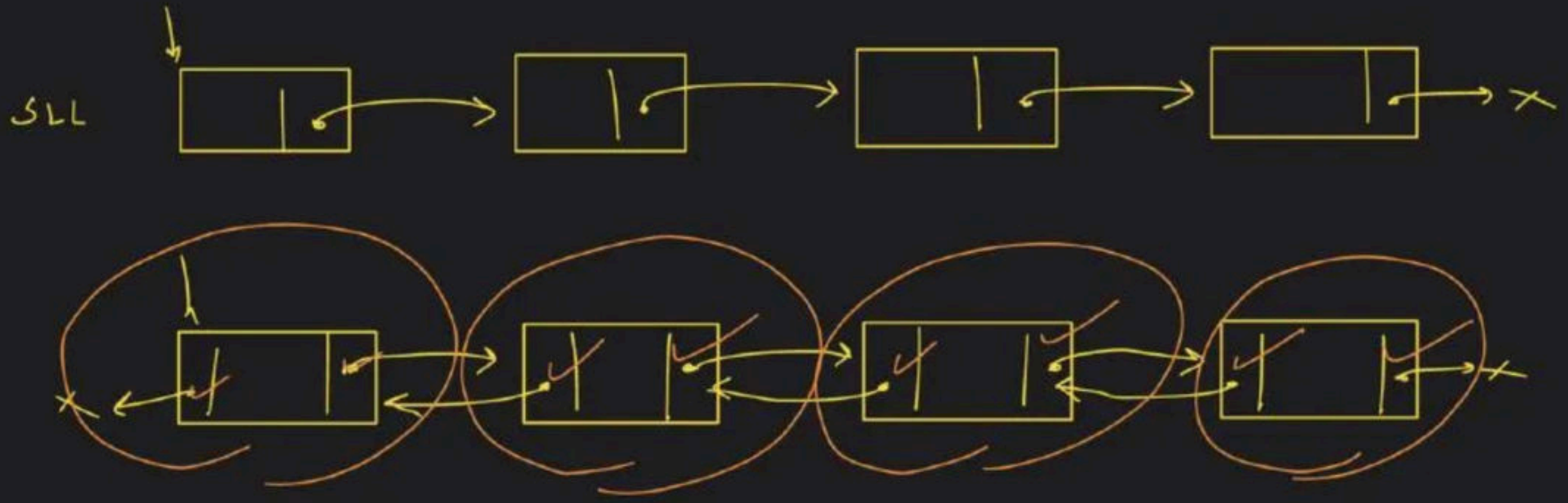

Wed
8:30 pm
DLL
Love
Babbar
luv

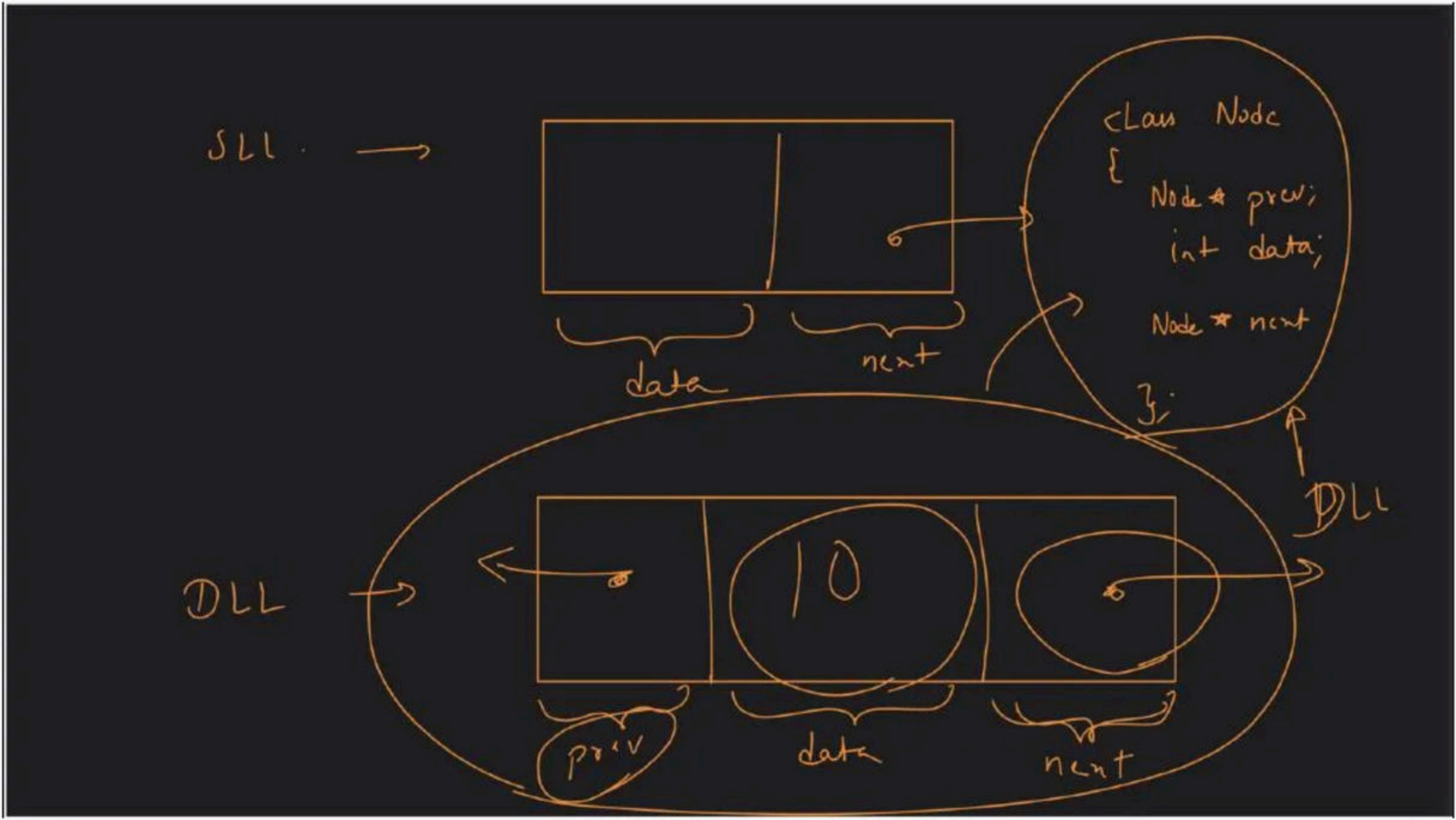
Linked List - Class 2

Special class

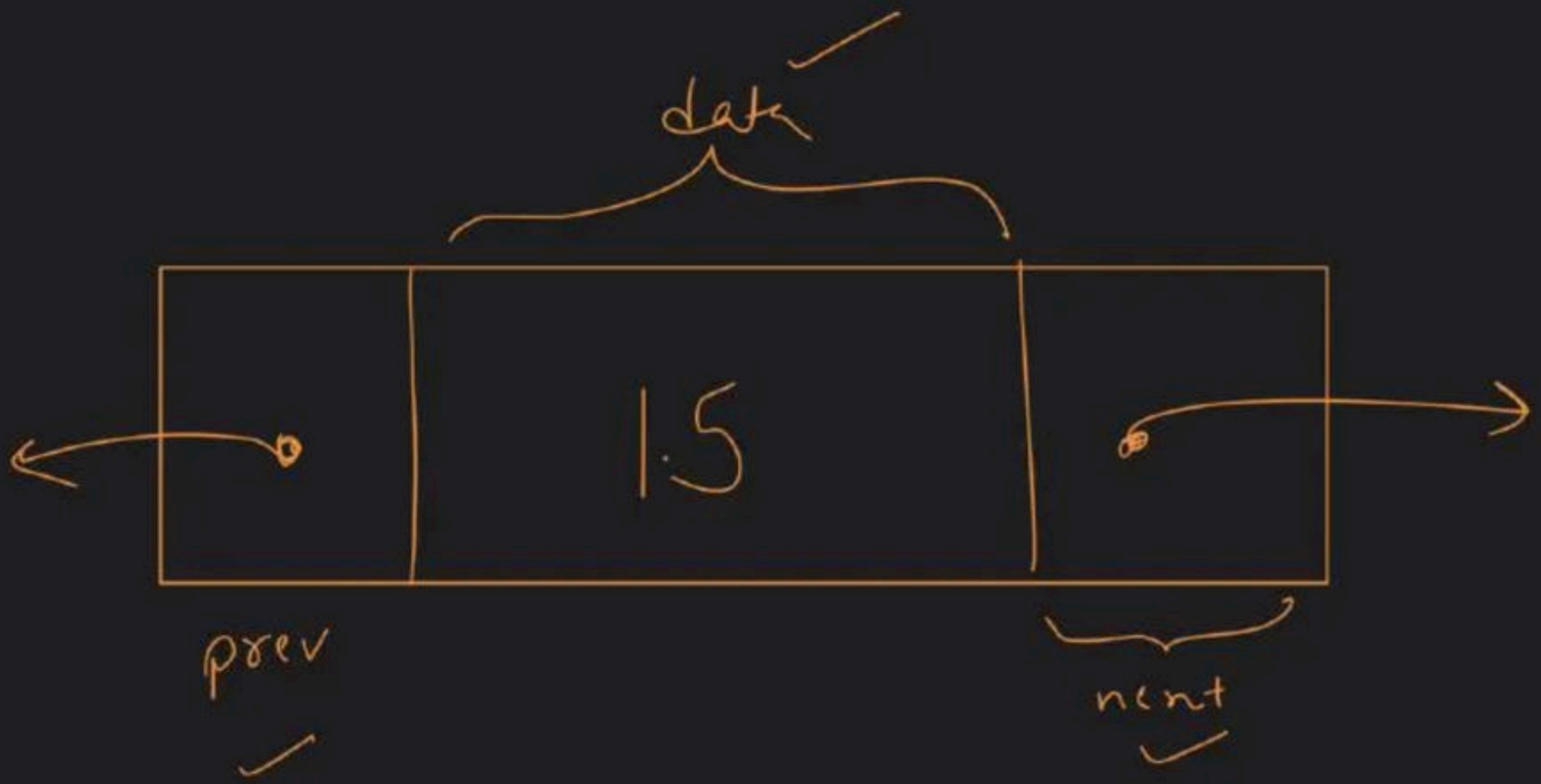
Love Babbar • July 17, 2024

Doubly linked list → diff - 1 node -)

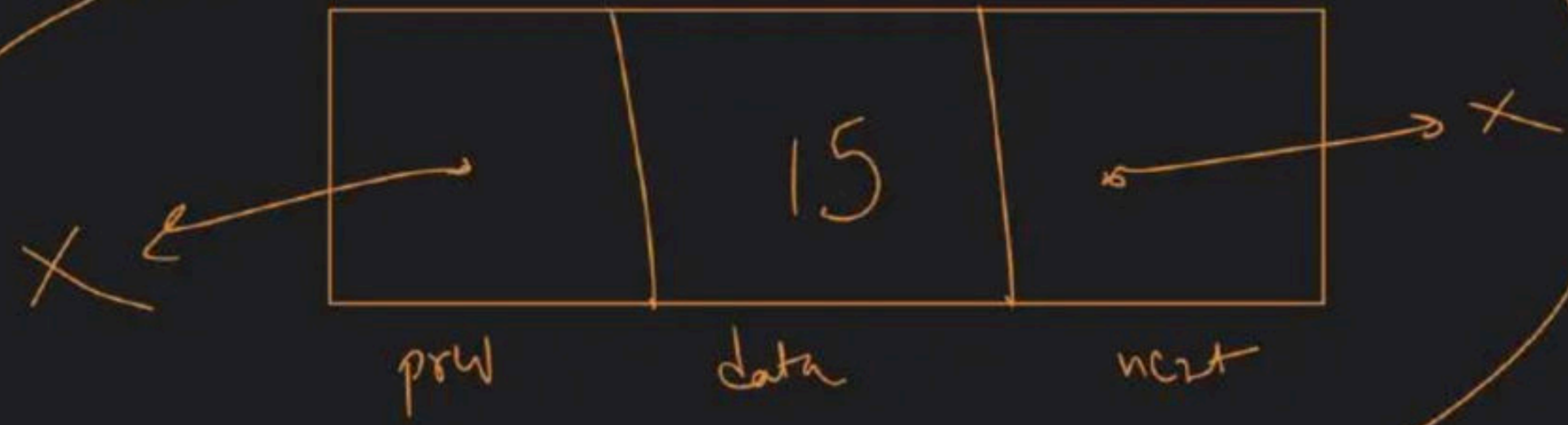



SLL
data
next
class Node
{
Node * prev;
int data;
Node * next
};
DLL
DLL
10
prev
data
next

Node:-

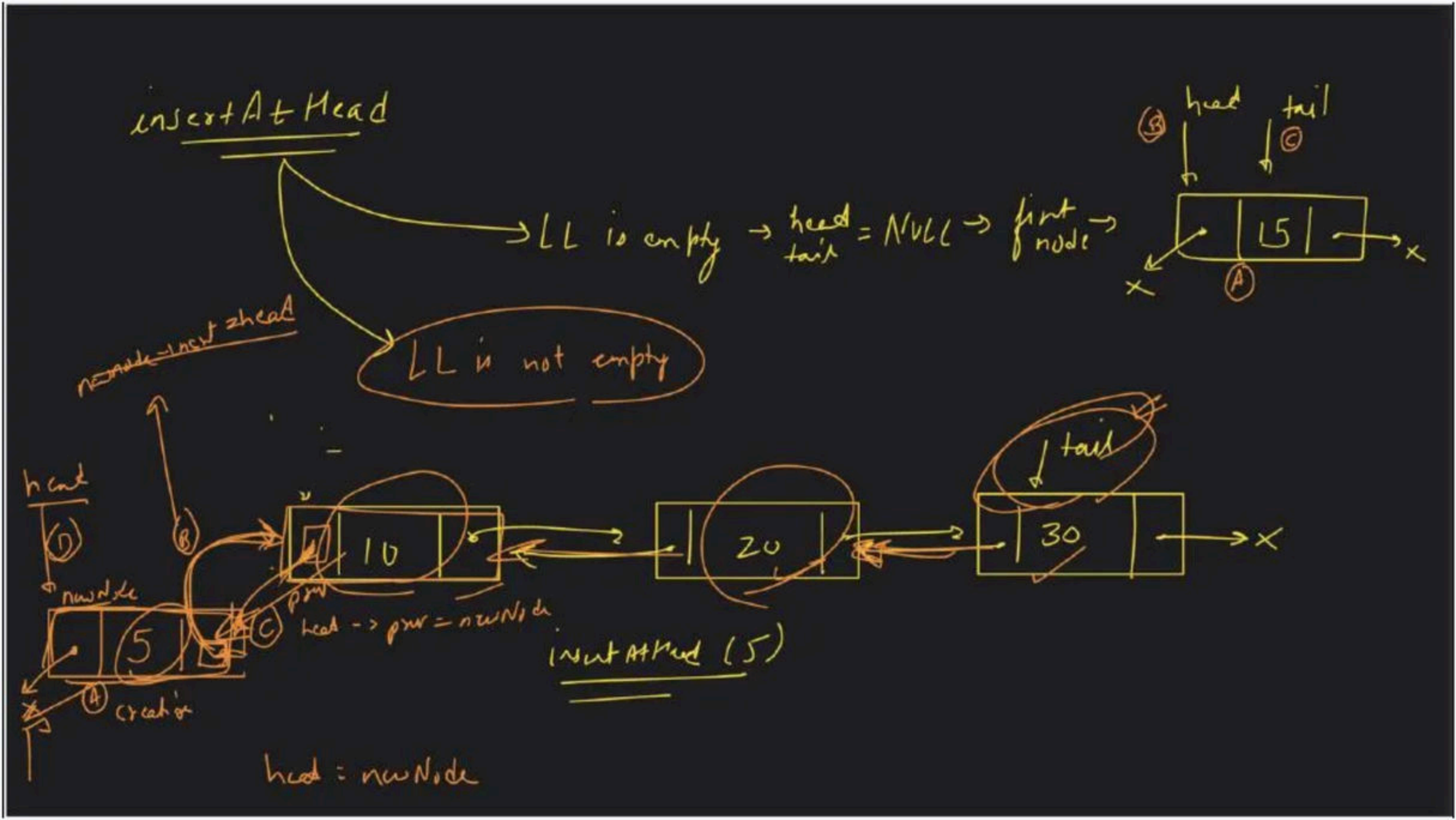


Node * newNode = new Node(15);



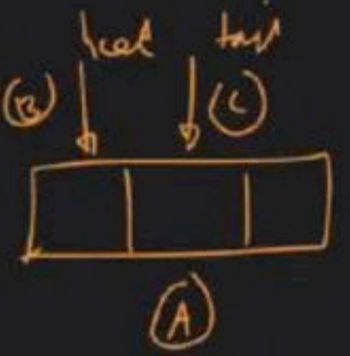
Insertion -

- insertAtHead ( )
- insertAtTail ( )
- insertAtPosition ( )

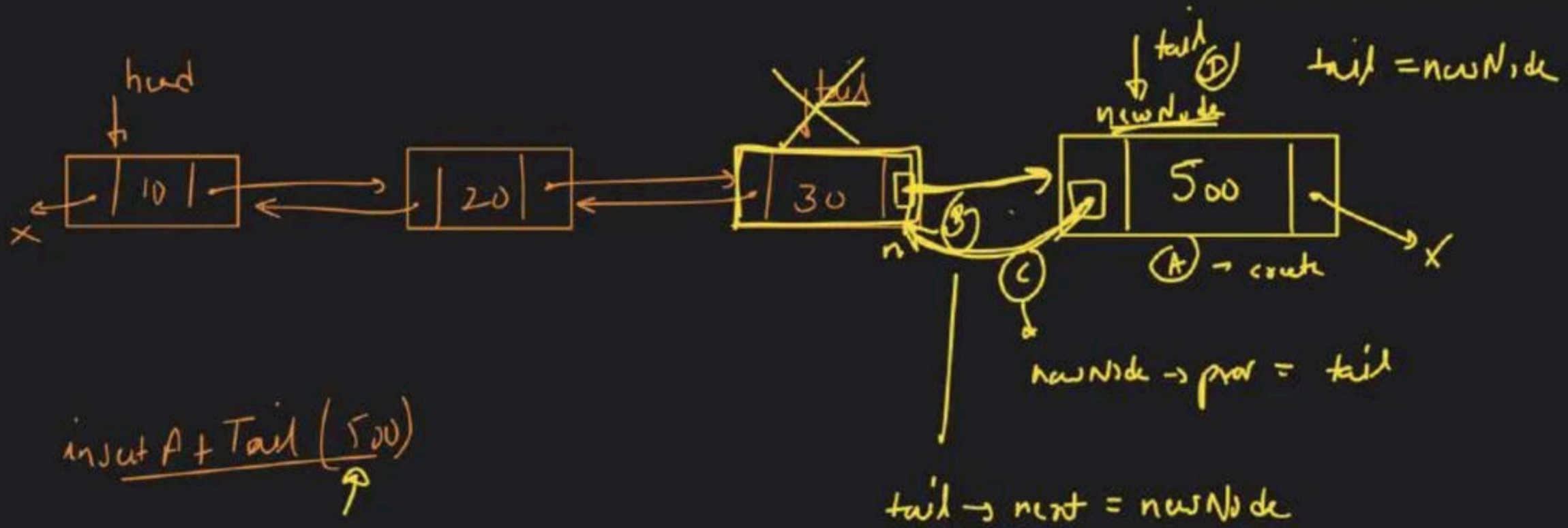

insertAtHead
LL is empty → head = NULL → first node →
head
tail
LL is not empty
newNode = insertAtHead
head
10
20
30
tail
newNode
5
pptr
head -> pptr = newNode
insertAtHead (5)
head : newNode

→ insertAtTail( )

LL is empty → head = tail = NULL → first node created →



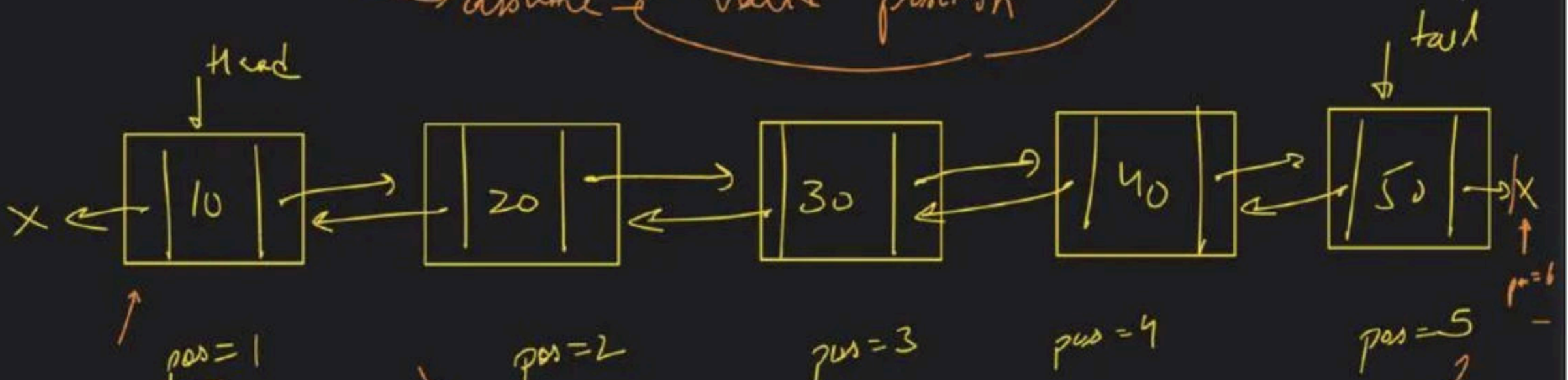
LL is not empty



insertAtTail(500)

→ insertAtPosition (value)

→ assume → valid position

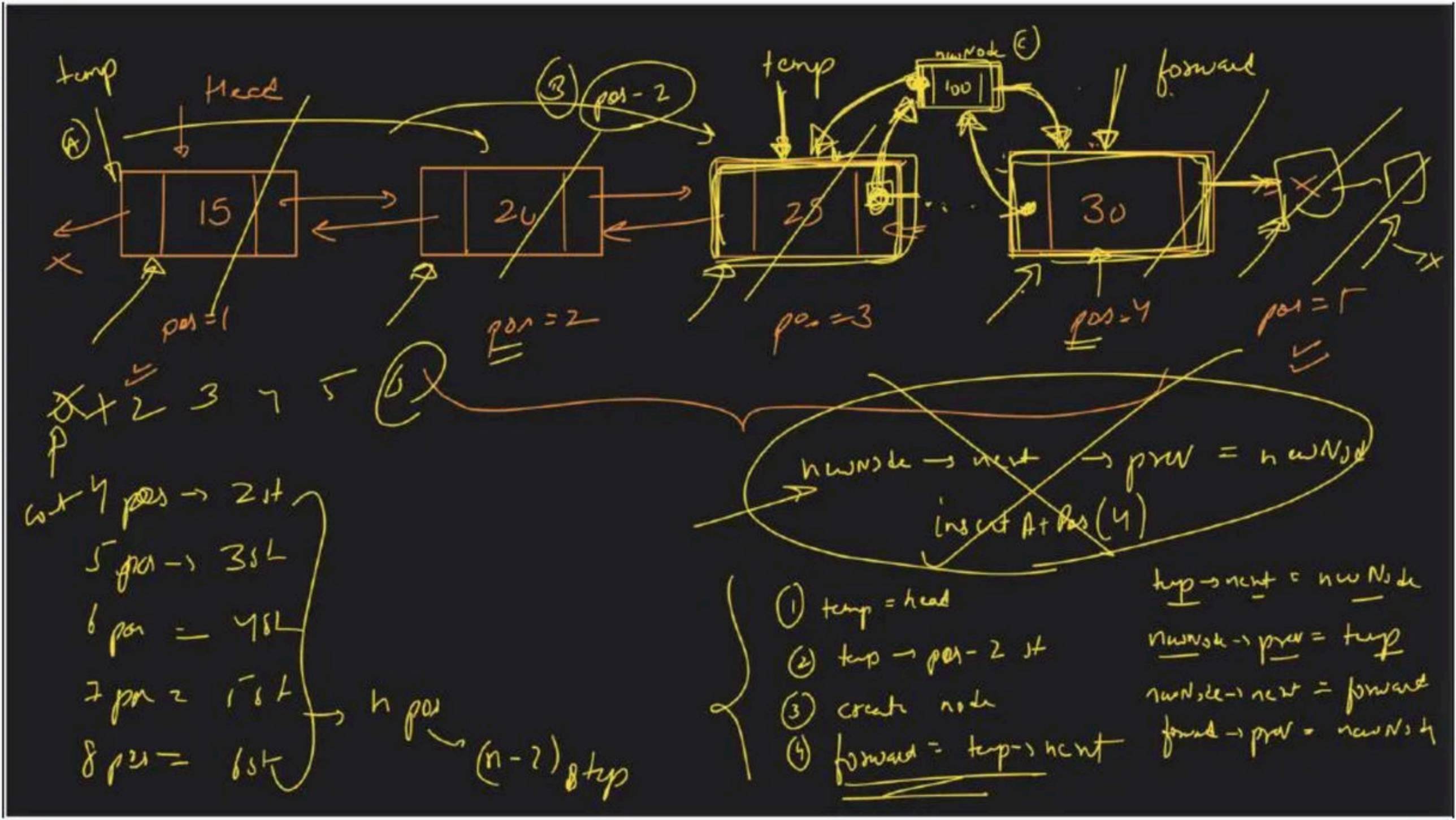


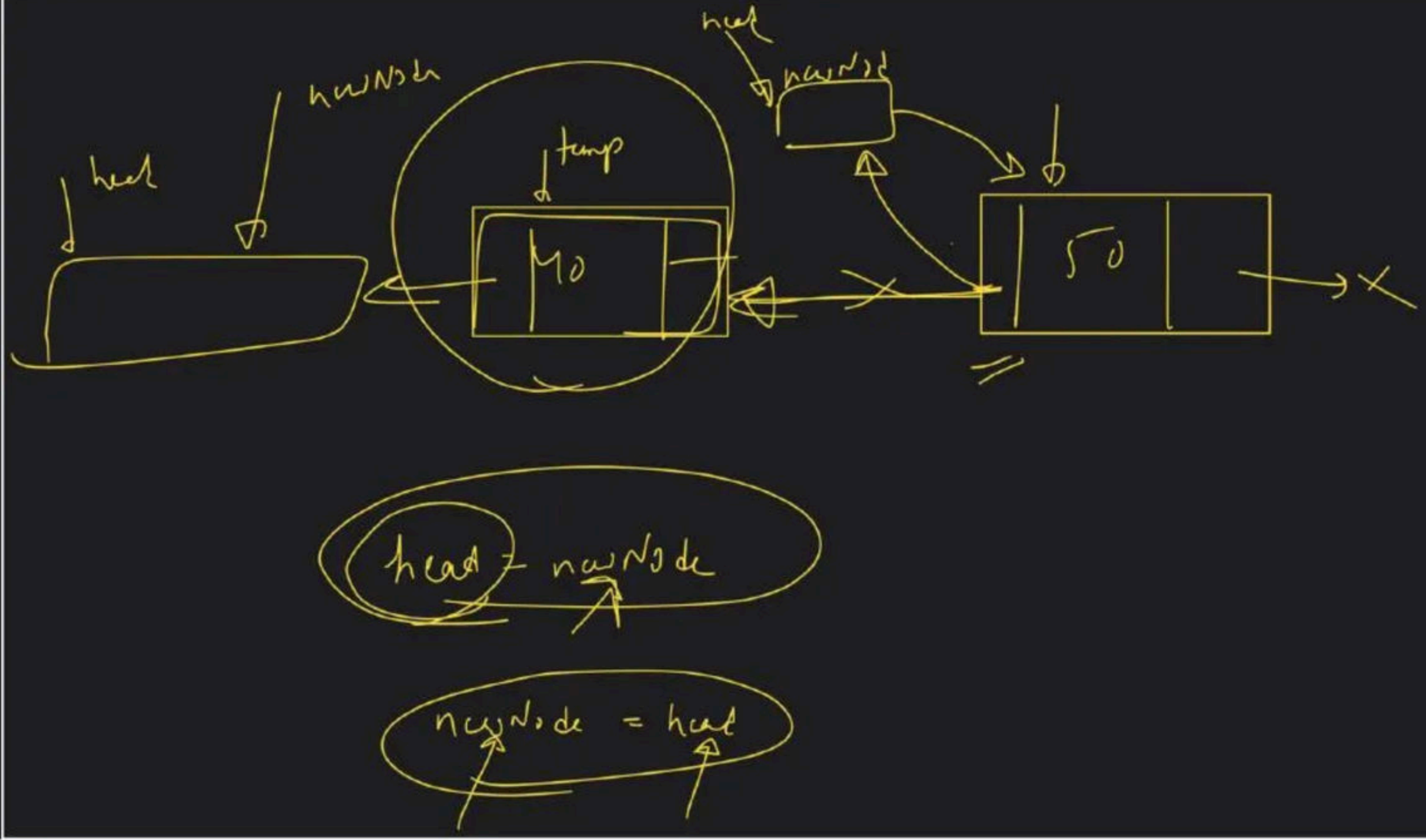
insertAtPos (1) → insertAtHead ( )

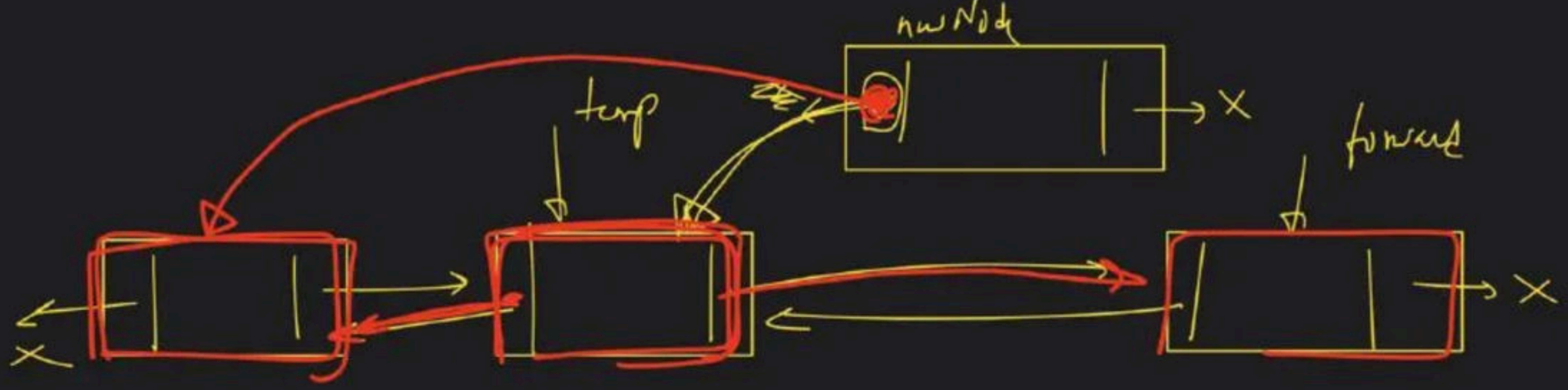
insertAtPos (6) → insertAtTail ( )

→ reuse

→ logic

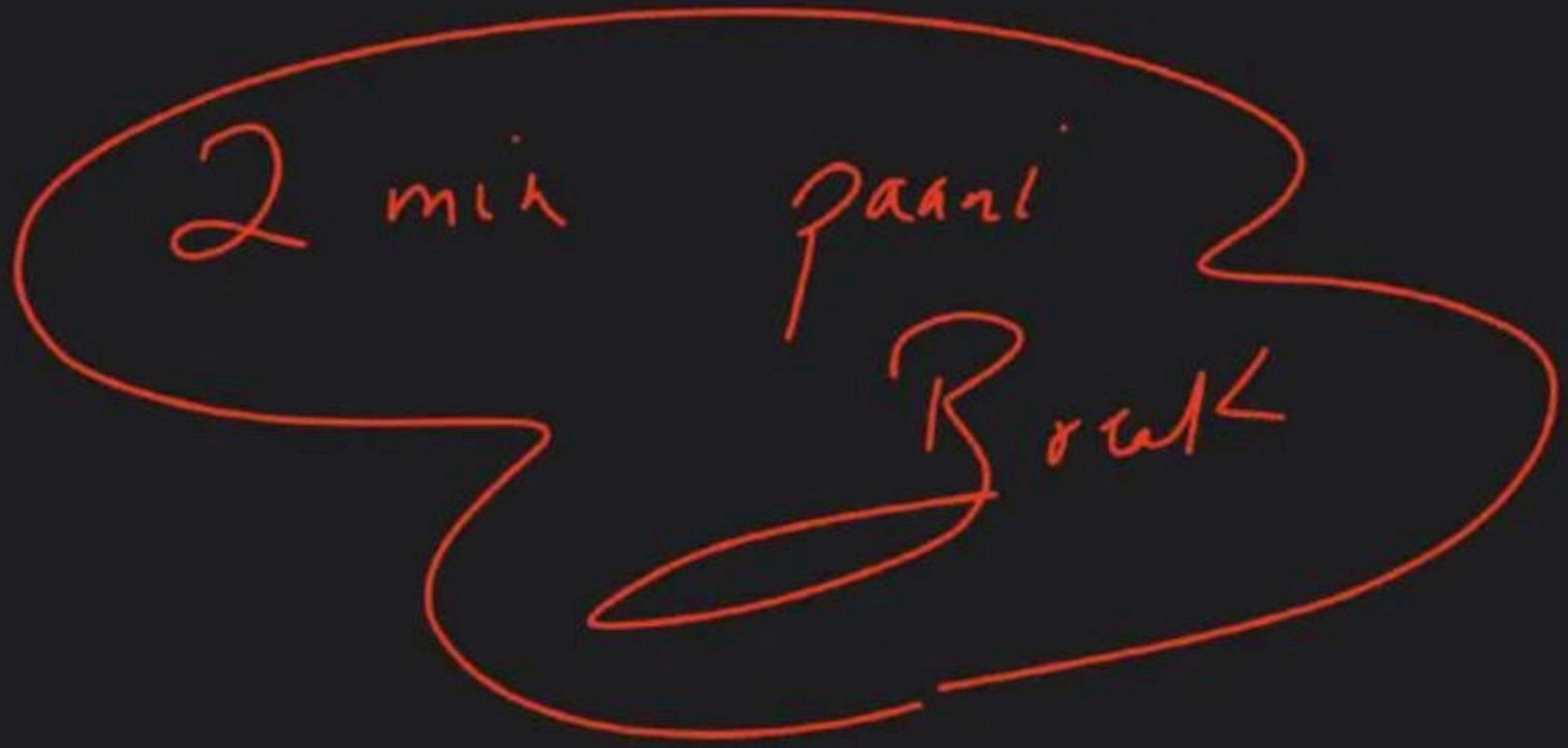

temp
head
pos - 2
temp
newNode
100
forward
15
20
25
30
pos=1
pos=2
pos=3
pos=4
pos=5
newNode -> next -> prev = newNode
insertAtPos(4)
(1) temp = head
(2) temp -> pos - 2 st
(3) create node
(4) forward = temp -> next
temp -> next = newNode
newNode -> prev = temp
newNode -> next = forward
forward -> prev = newNode
1st pos
2nd pos
3rd pos
4th pos
5th pos
6th pos
7th pos
8th pos
n pos -> (n-2) th

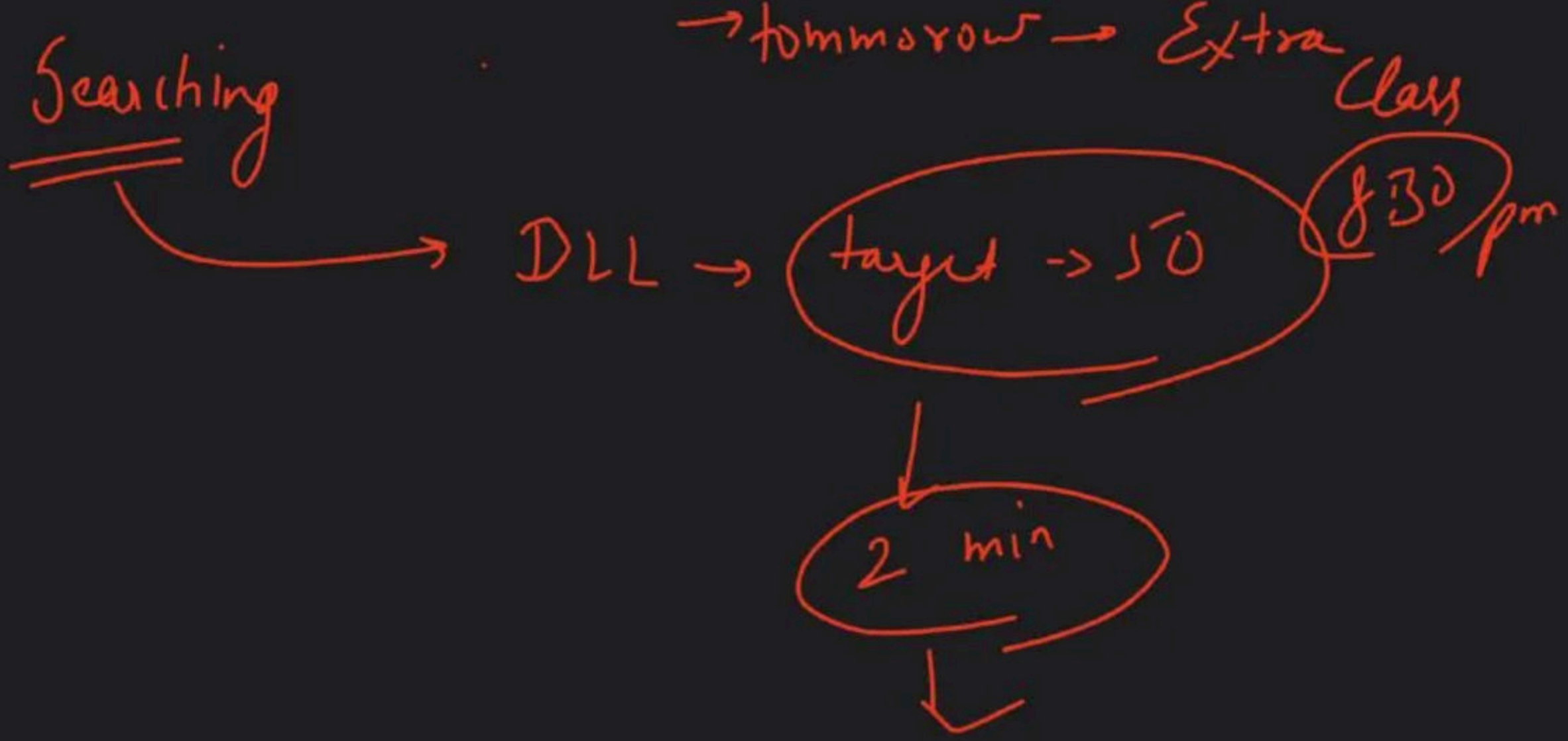

head
newNode
temp
head
newNode
10
50
X
head = newNode
newNode = head



newNode -> prev = temp

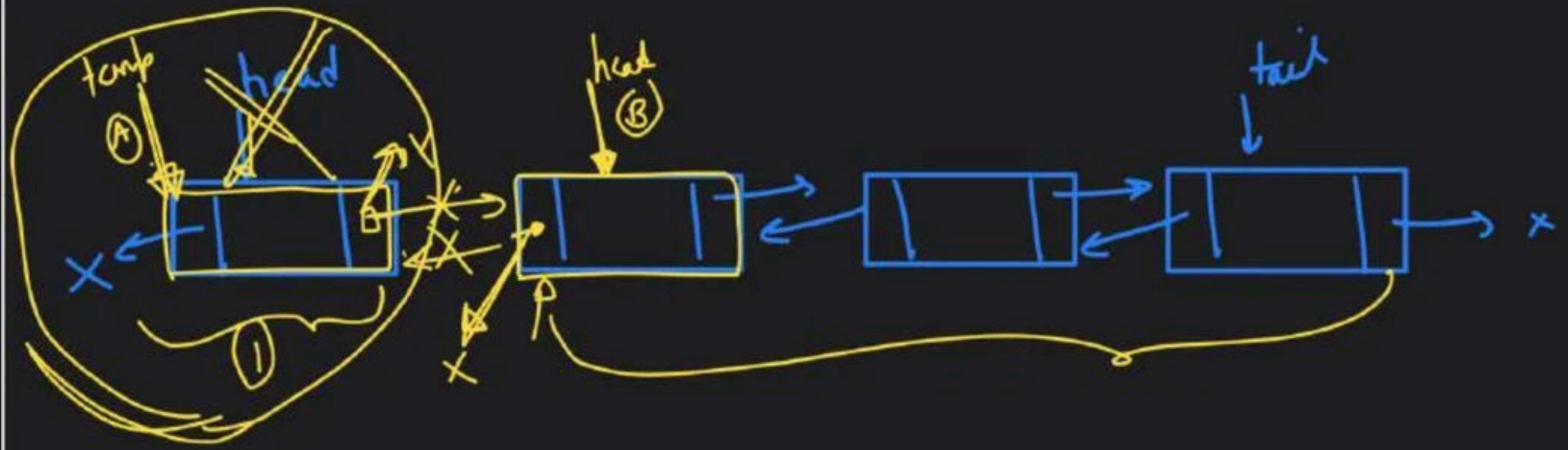
newnode -> prev = temp -> prev


2 min paani
Break


Searching
→ tommorow → Extra Class
DLL →
target -> 50
8 30 pm
2 min

→ Deletion

- LL is empty → delete
- head node → delete
- single node → delete
- ! head → delete

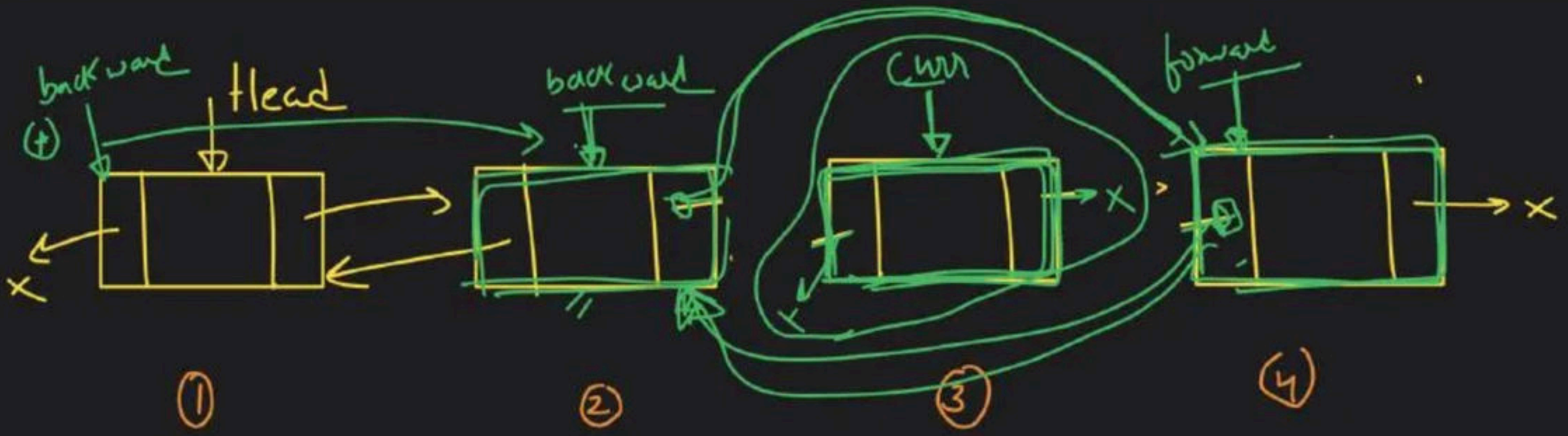


(A) Node * temp = head — (d) temp -> next = NULL

(B) head = head -> next (c) delete temp

(C) head -> prev = NULL

deleteFromFront(1)



(a) backward = head

(b) backward → prev → step

(c) curr = backward → next

(d) forward = curr → next

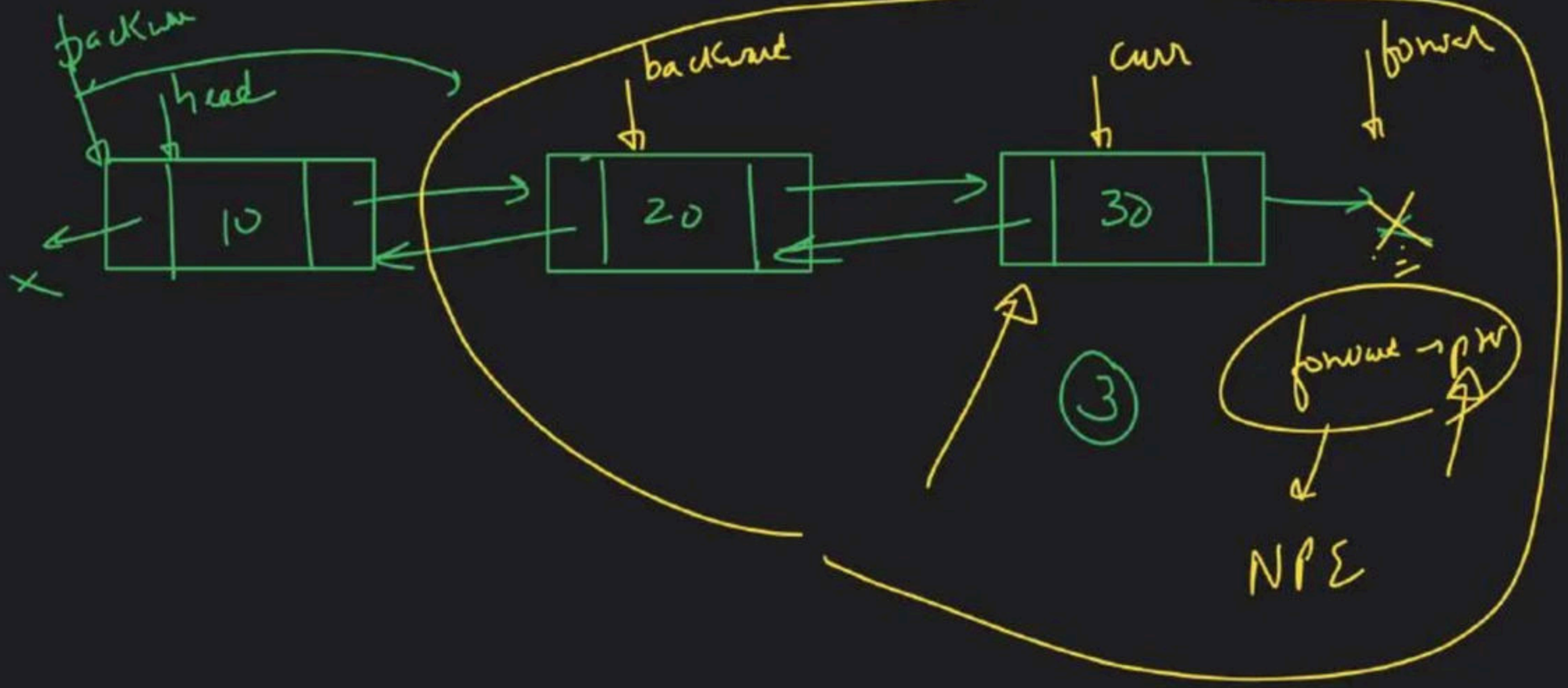
backward → next = forward

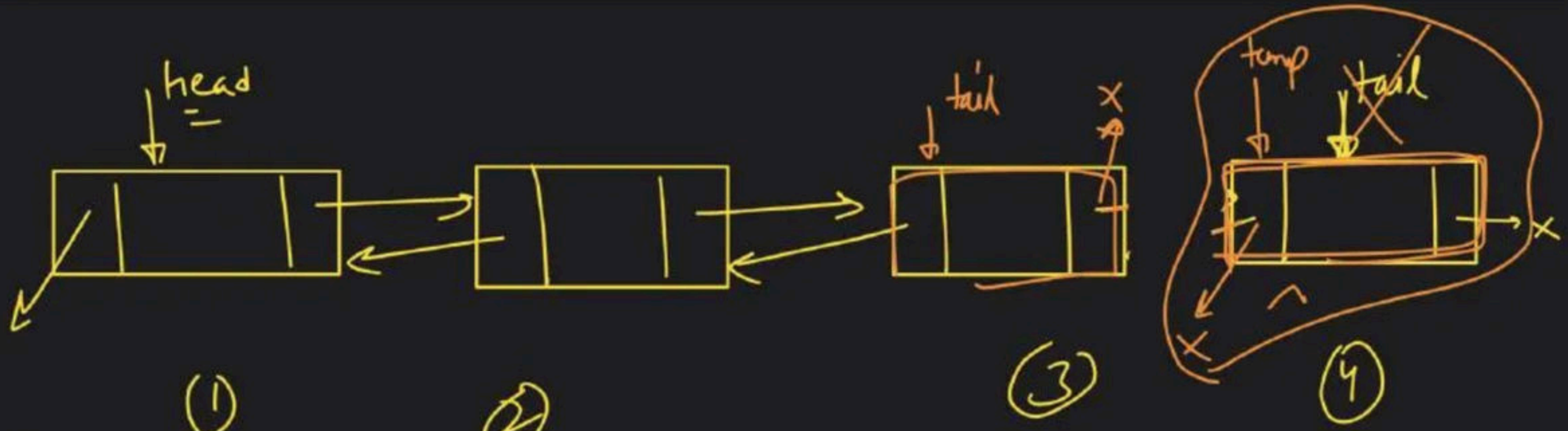
forward → prev = backward

curr → prev = NULL

curr → next = NULL

delete curr;


backward
head
backward
curr
forward
10
20
30
x
3
forward->prev
NPE



(A) temp = tail

(B) tail = temp → prev

(C) tail → next = NULL

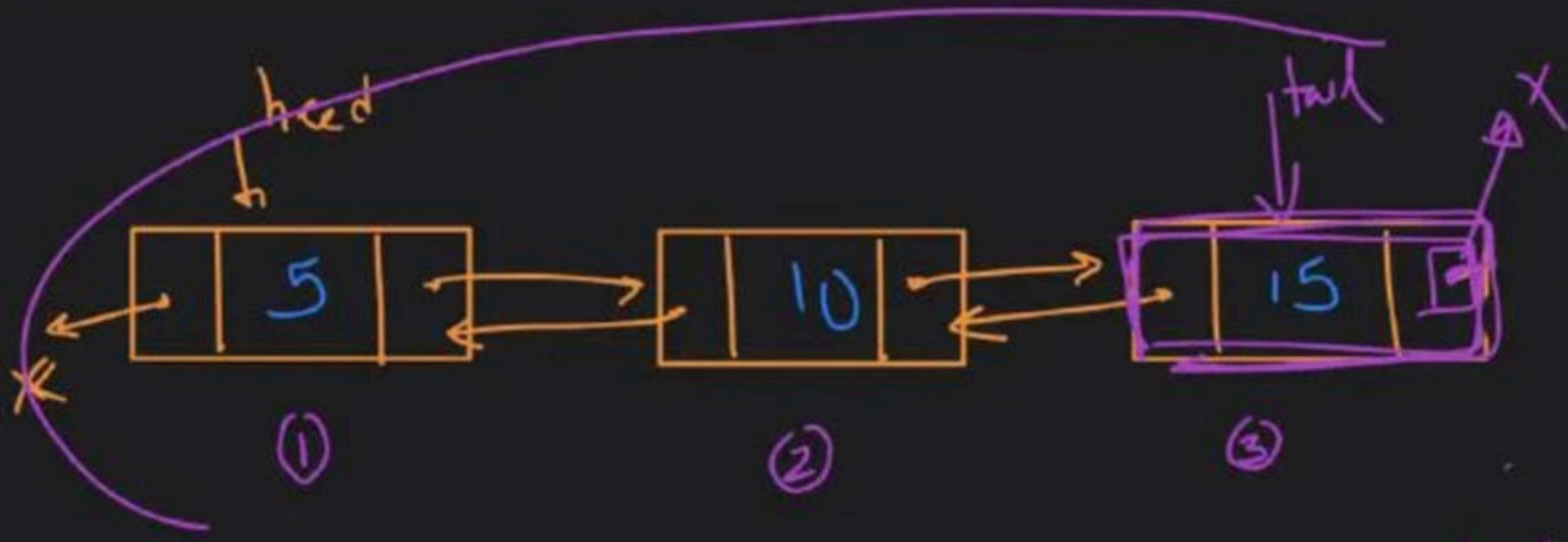
(D) temp → prev = NULL

(C) delete temp

tail delete

deleteFromPos (4)

position == length



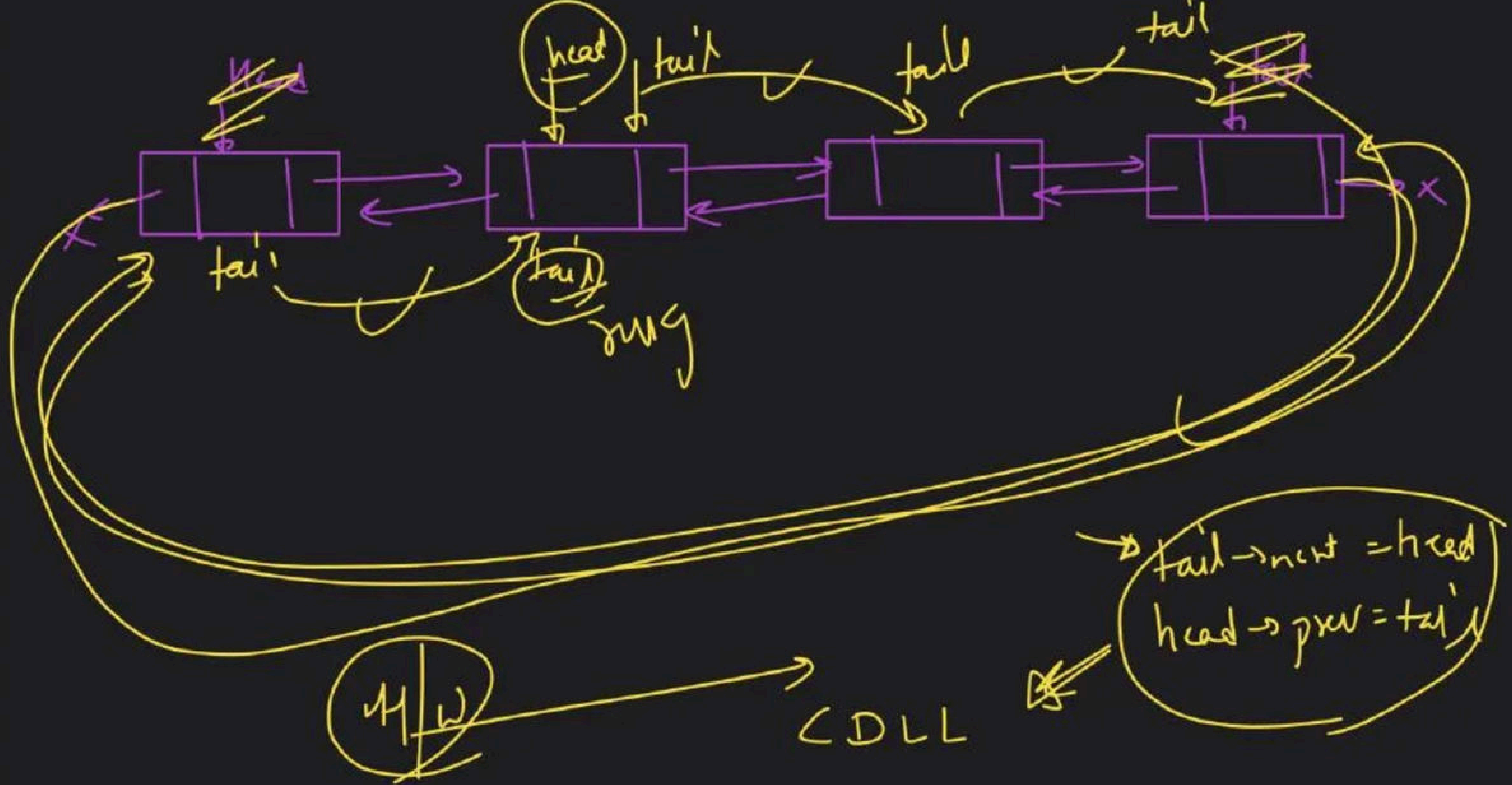
SLL

DLL

CDLL

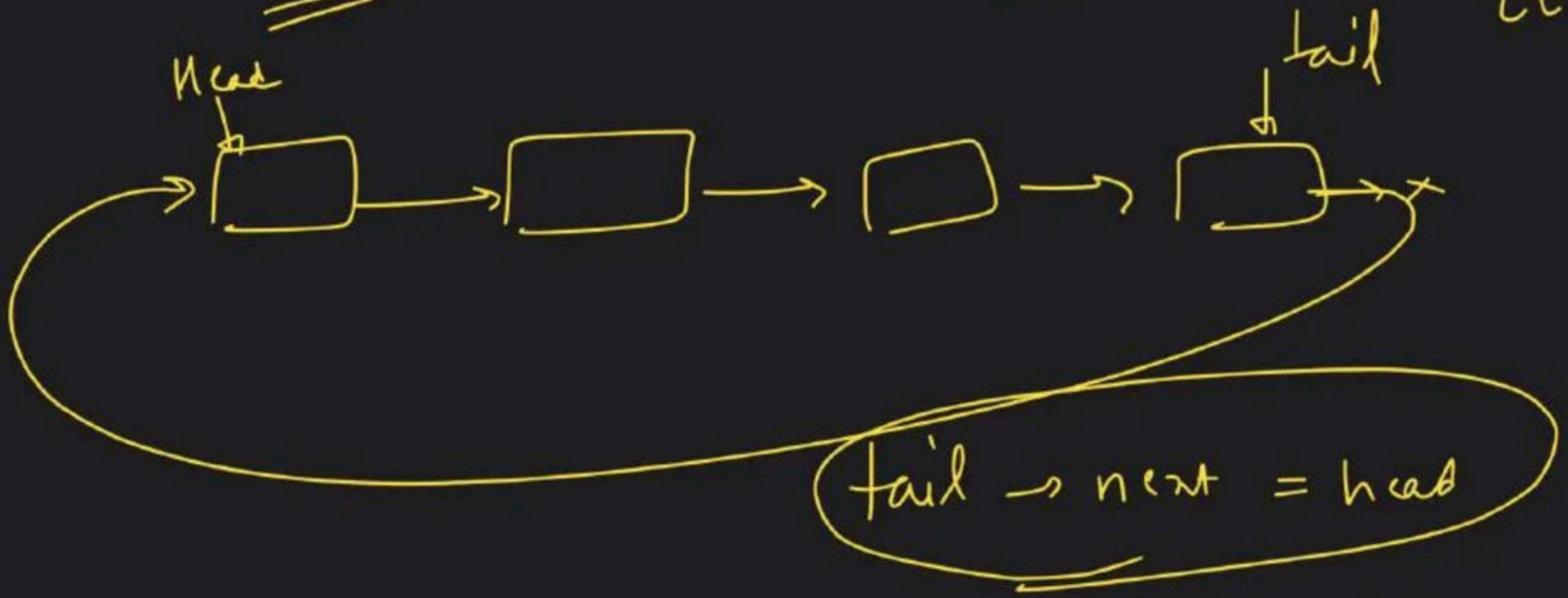
```
Node * temp
tail = temp -> prev
temp -> prev = NULL
tail -> next = NULL
delete temp
```

deleteLast(y)


head
tail
tail
tail
tail
tail
tail->next = head
head->prev = tail
H/W
CDLL

→ C·L·L

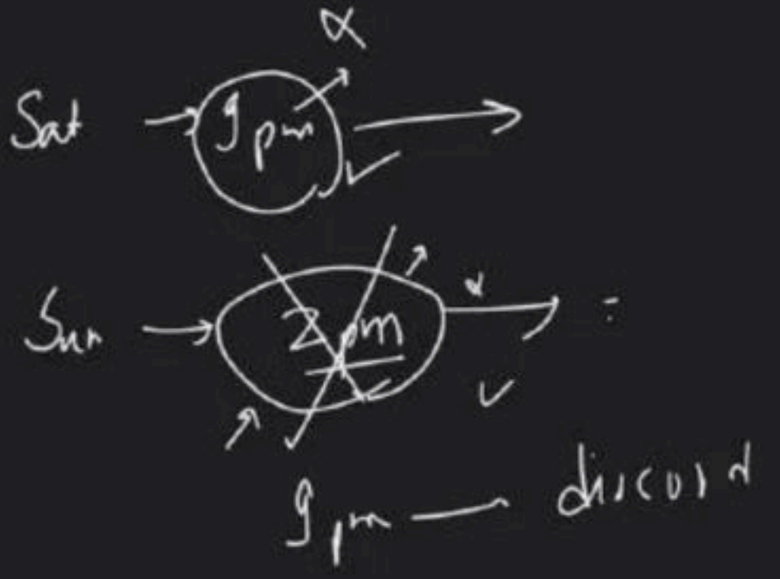
KHUD → length of CLL



S·L·L → C·L·L

Linked List Class-3

Special class

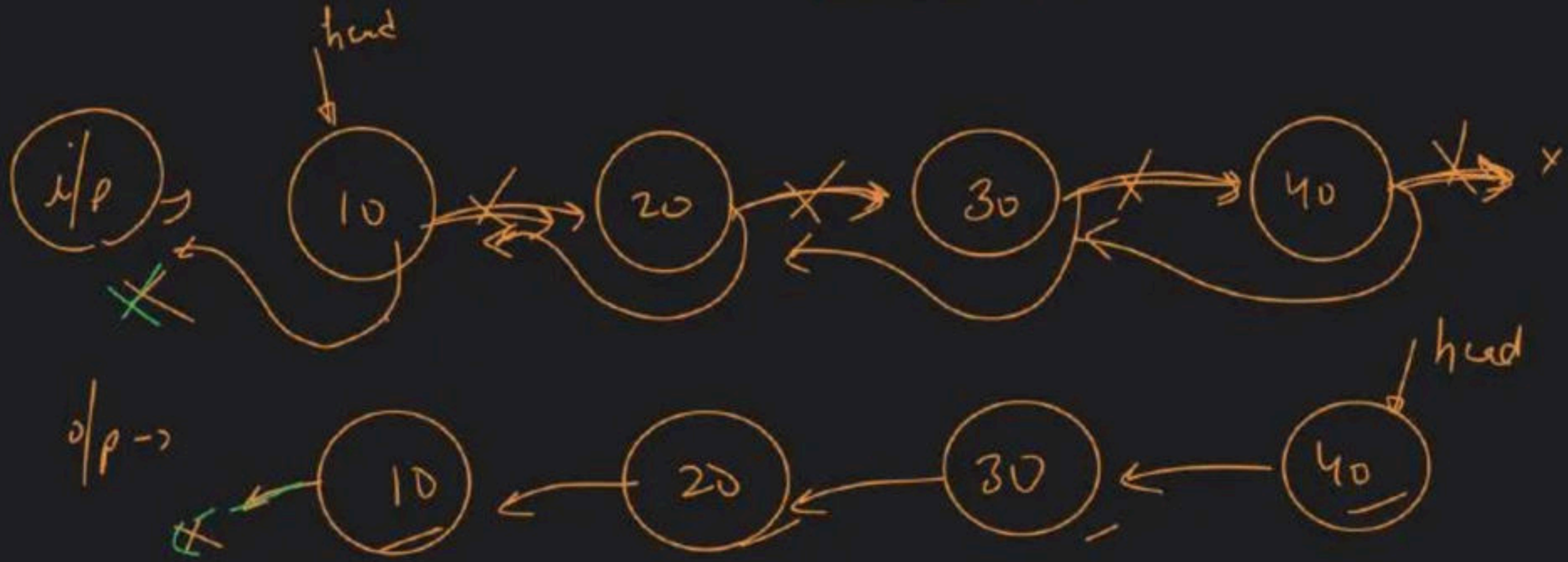


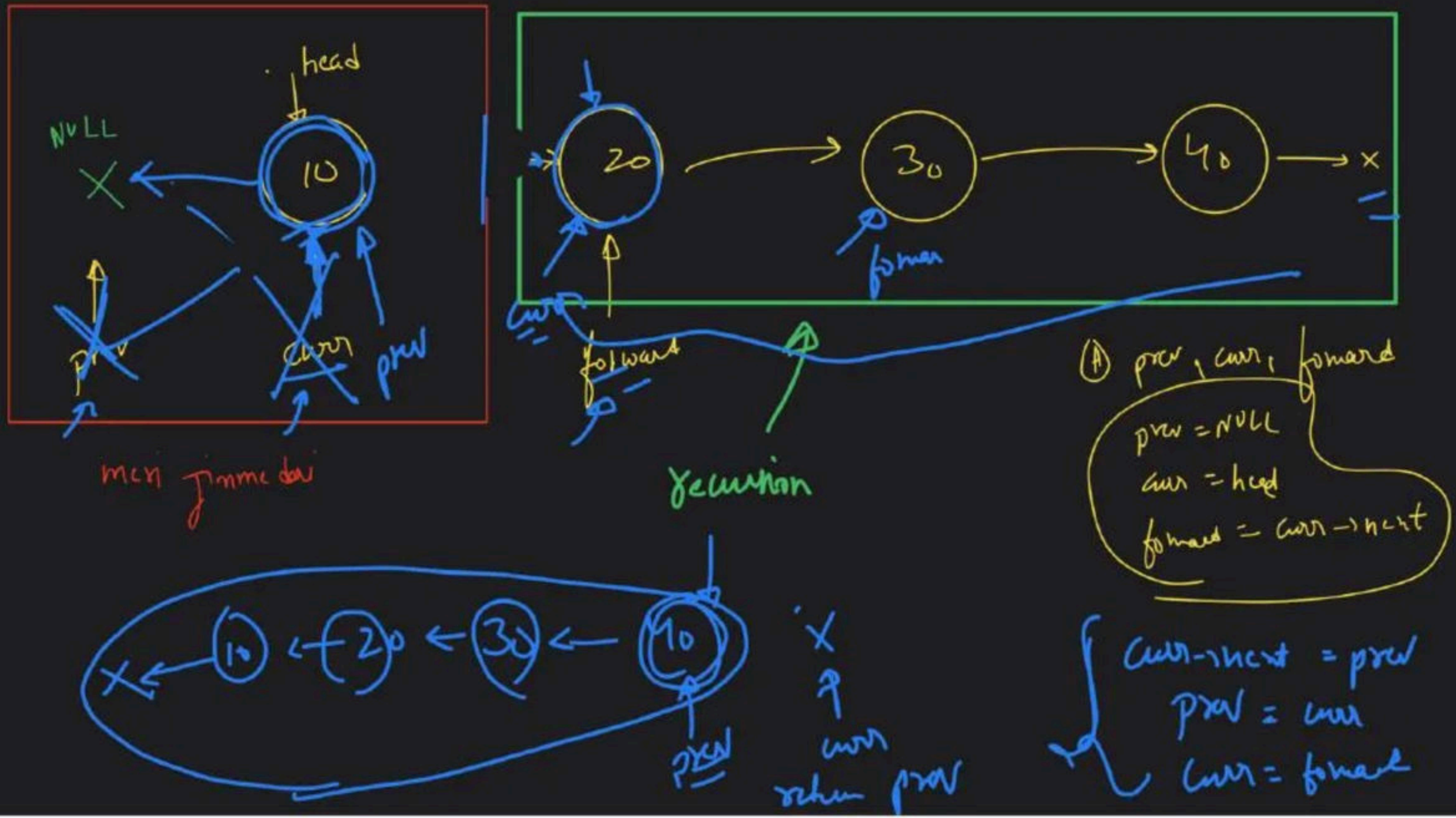
Love Babbar • July 18, 2024

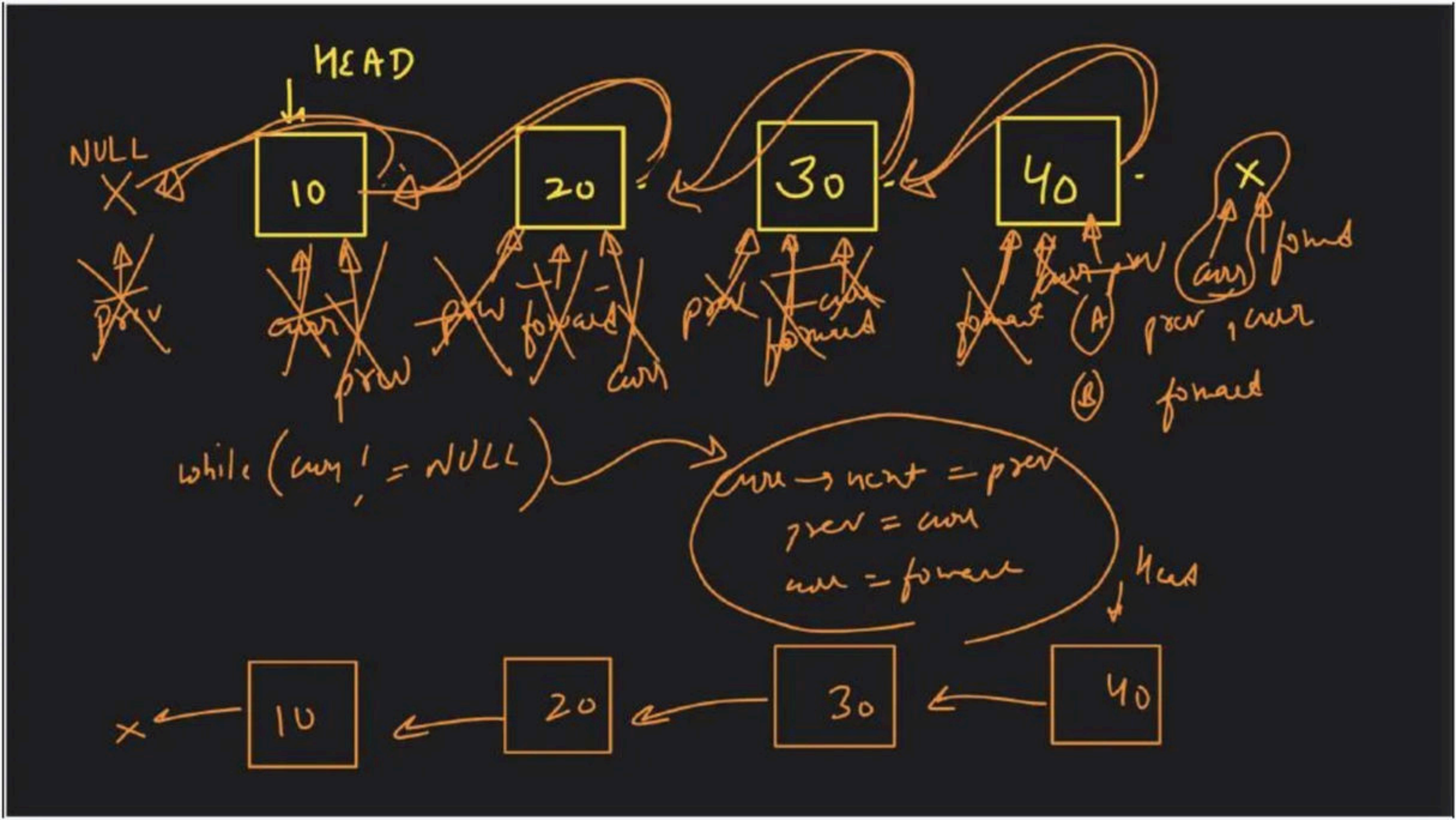
→ Reverse a L.L

→ iterative

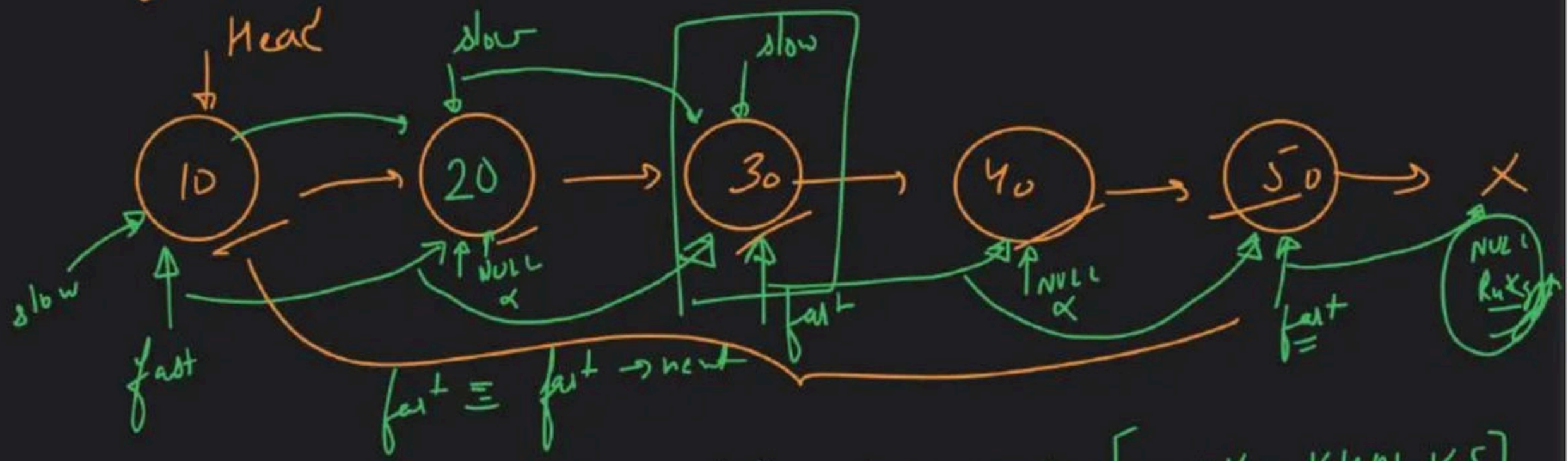
→ recursive




head
NULL
10
20
30
40
X
prev
curr
forward
next immediate
Recursion
A prev, curr, forward
prev = NULL
curr = head
forward = curr -> next
curr->next = prev
prev = curr
curr = forward
return prev


HEAD
NULL
10
20
30
40
X
prev
curr
forward
while (curr != NULL)
curr → next = prev
prev = curr
curr = forward
HEAD
X
10
20
30
40

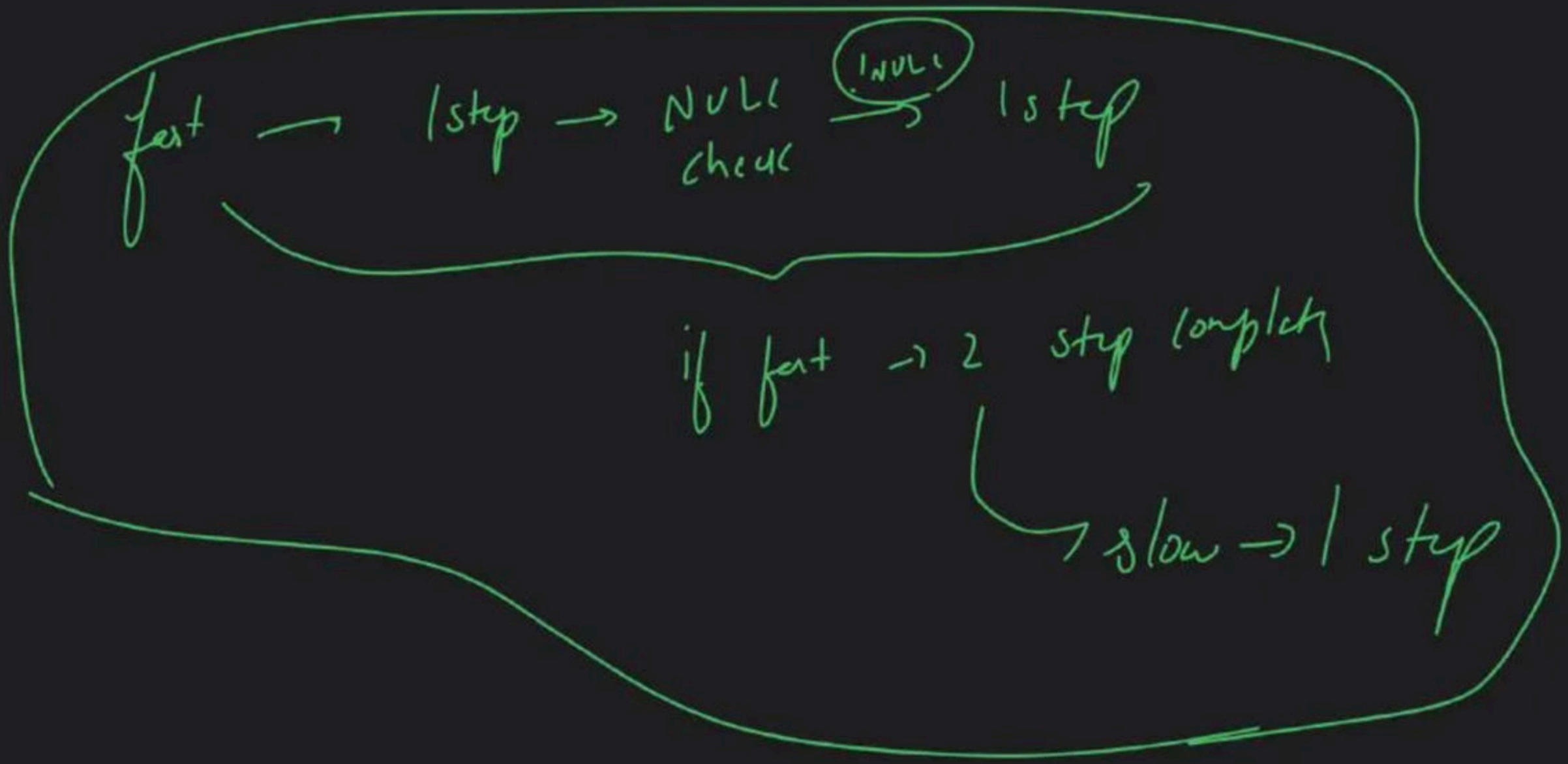
→ find the middle node of L.L

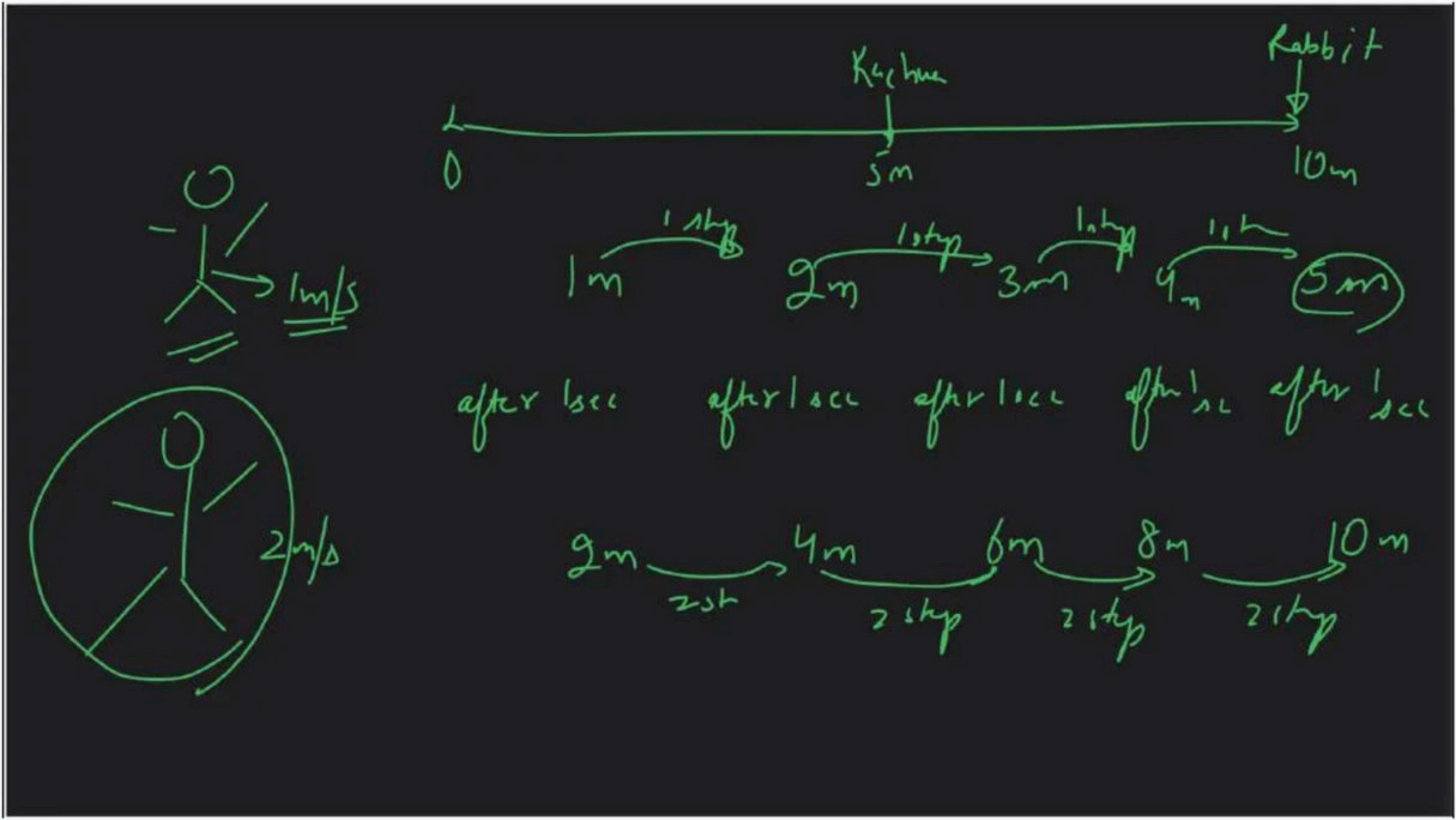


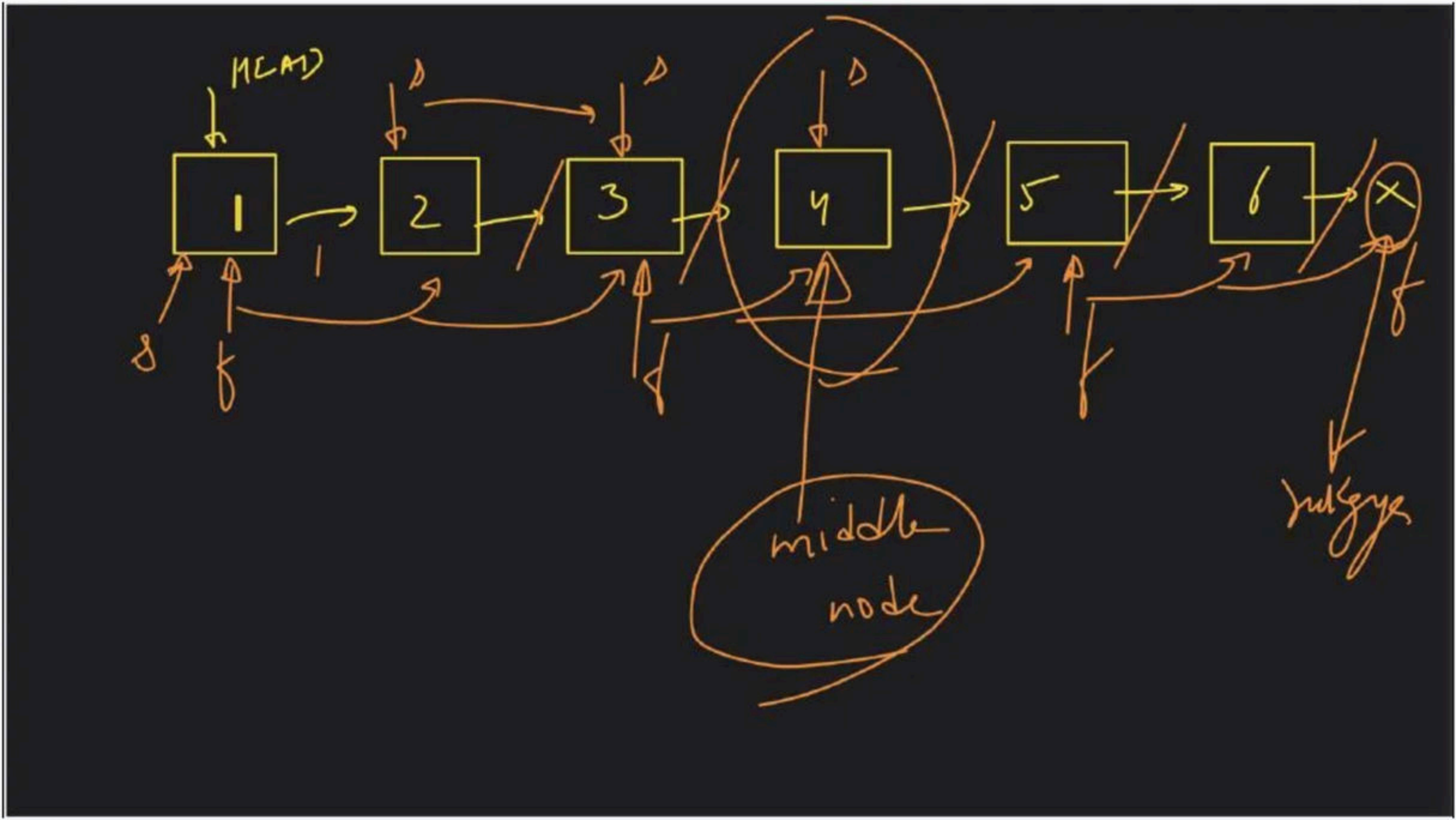
Rabbit → fast → 2 step [aagKe KHOLKE]

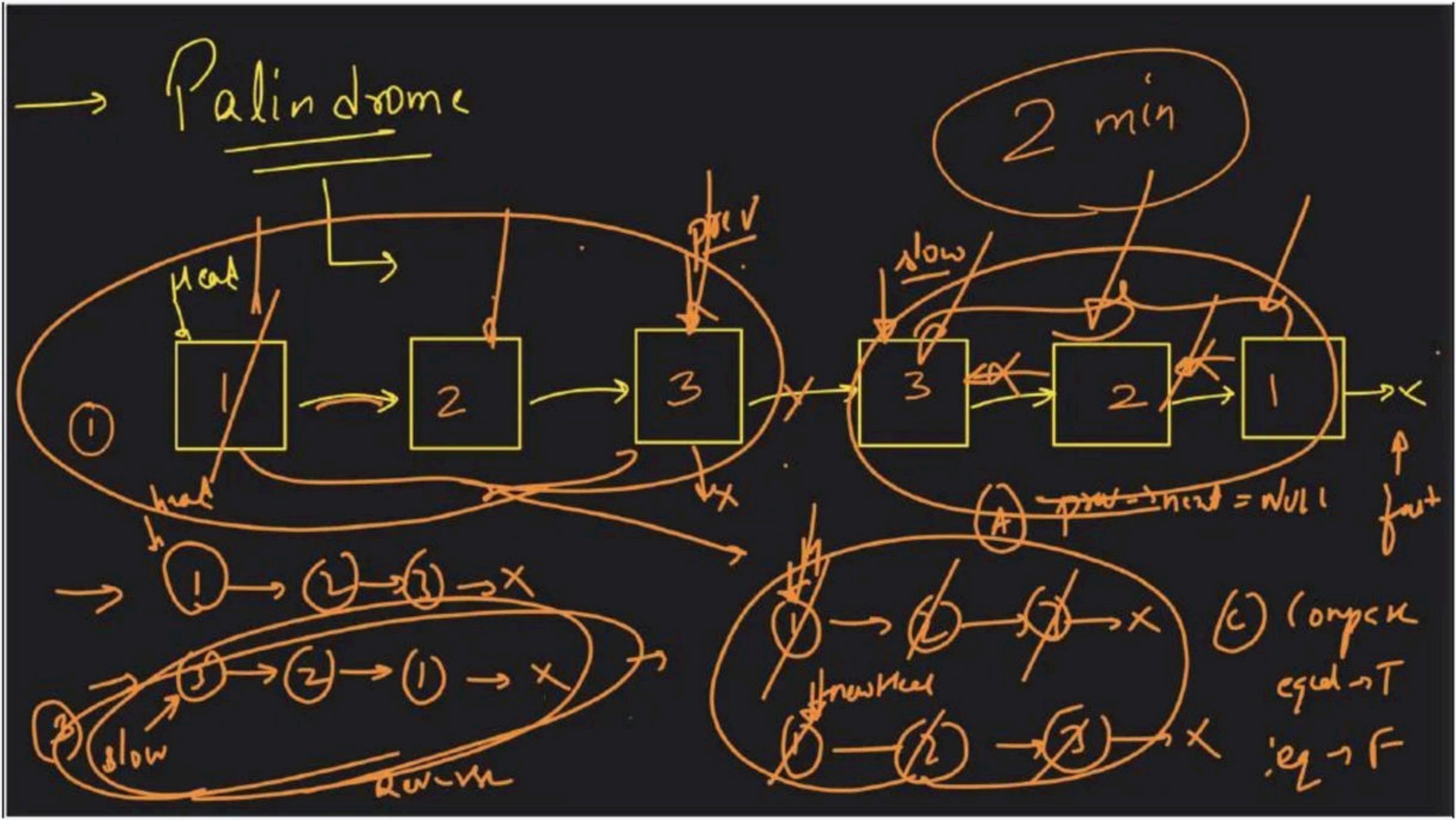
Kachua → slow → 1 step

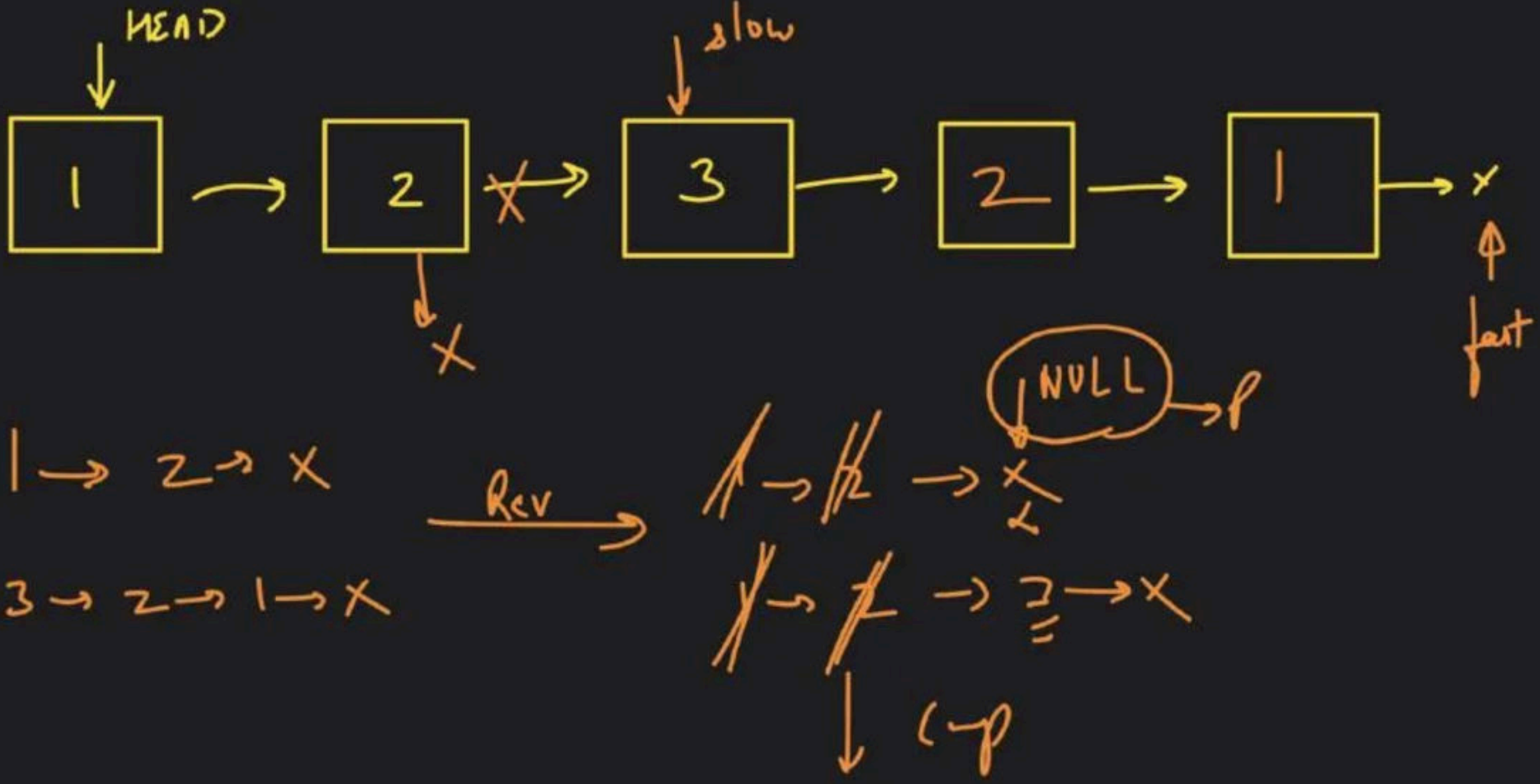

HEAD
ID
X
NULL
fast
slow
Rukya

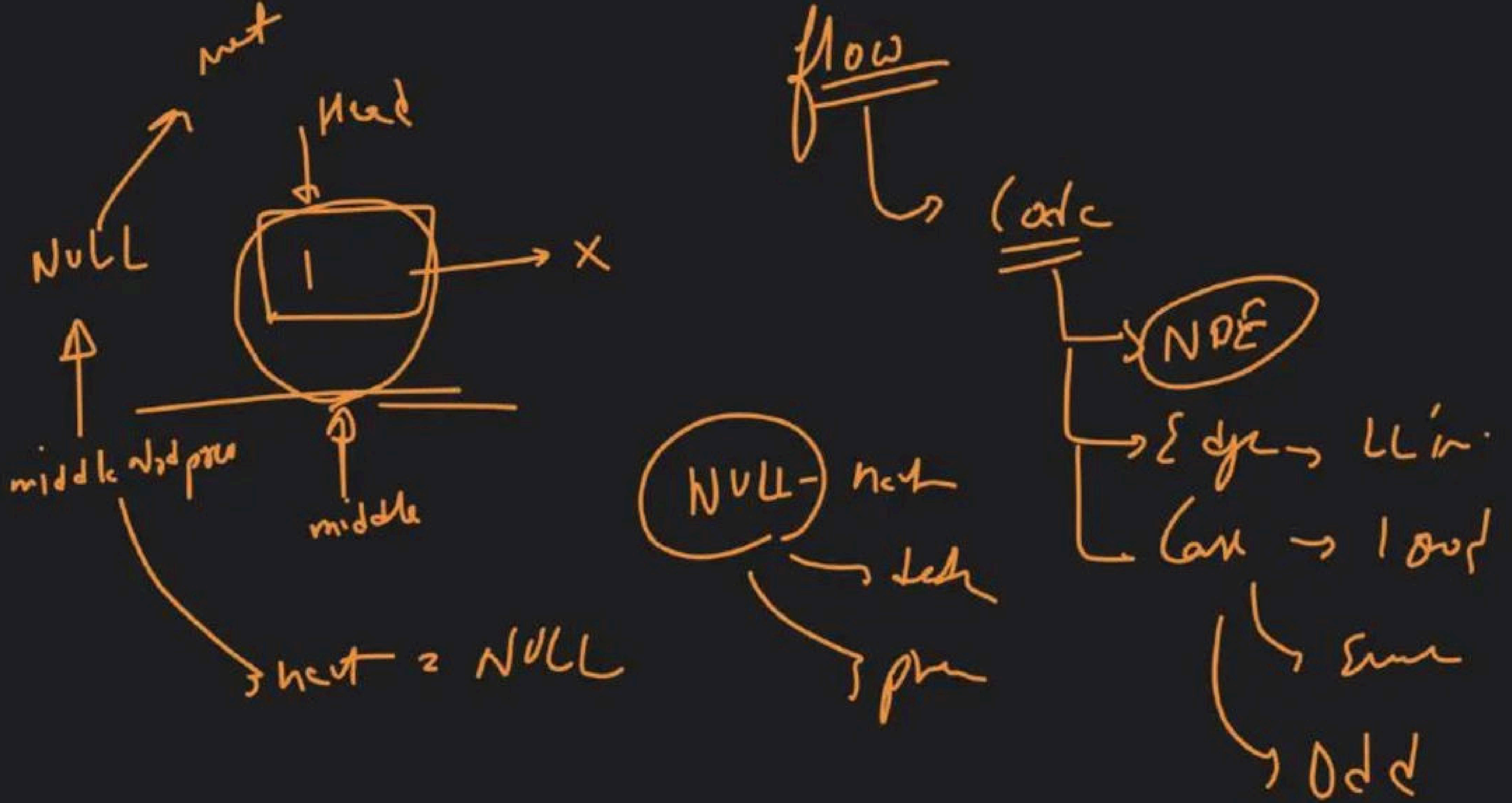

fast → 1step → NULL check
NULL
1 step
if fast → 2 step complete
slow → 1 step


Kuchuu
Rabbit
0
5m
10m
1m/s
1step
1m
2m
3m
4m
5m
after 1sec
after 1sec
after 1sec
after 1sec
after 1sec
2m/s
2m
4m
6m
8m
10m
2sec
2step
2step
2step
2step

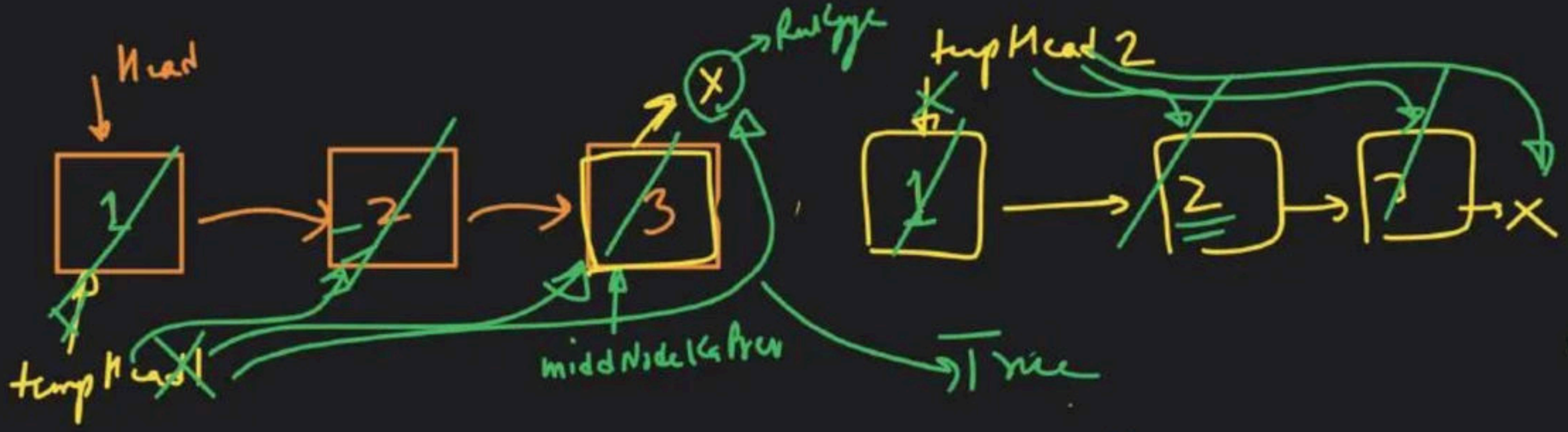

HEAD
1
2
3
4
5
6
X
middle node
Judge


Palindrome
2 min
head
prev
slow
fast
1
2
3
3
2
1
NULL
slow
reverse
newHead
equal → T
!eq → F


HEAD
slow
1
2
3
2
1
X
fast
1 → 2 → X
3 → 2 → 1 → X
Rev
NULL
p
cp



Head
NULL
1
X
middle
next = NULL
flow
Code
NULL
Odd

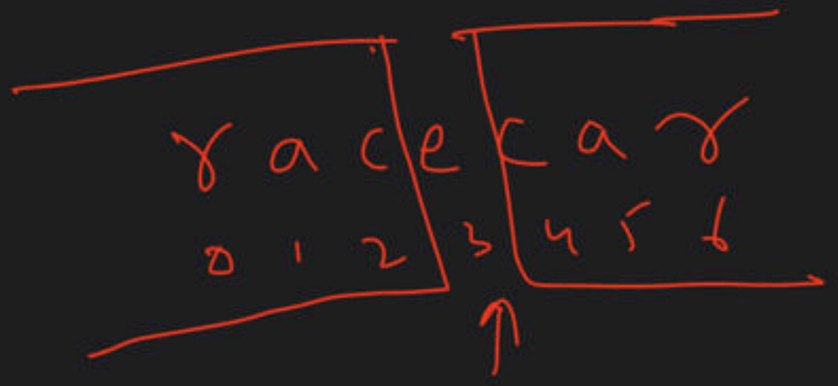


Step1 → find middle Node / middleNodeKaPrev & Break

middleNodeKaPrev → next = NULL

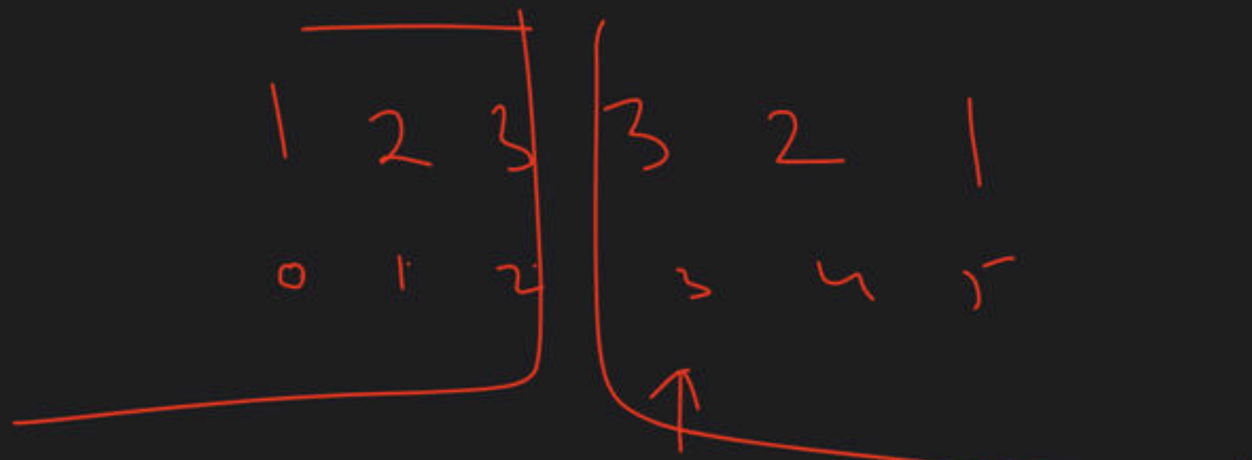
Step 2 → reverse second half LL

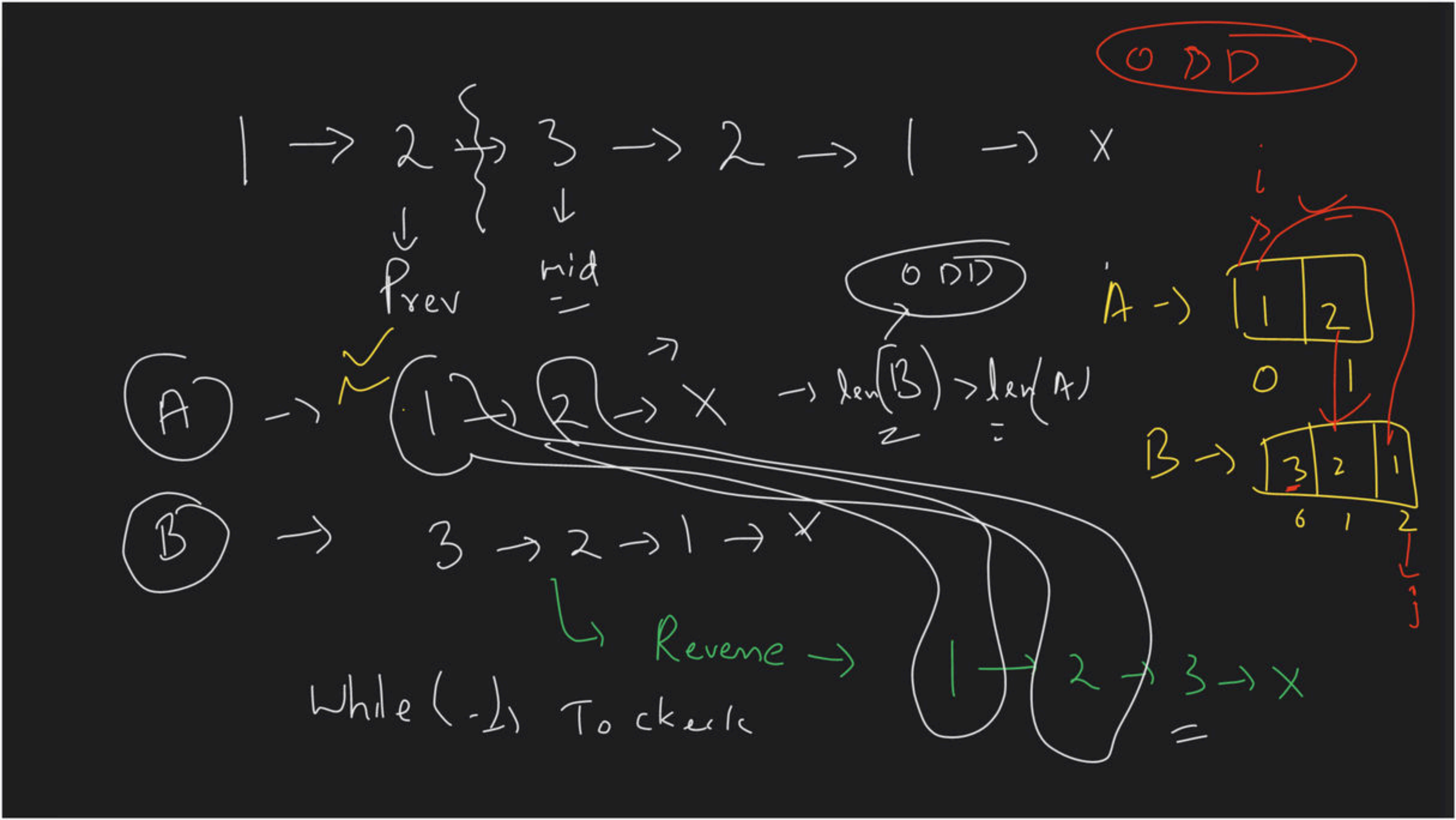
Step 3 → compare


racecar
0 1 2 3 4 5 6


1 2 3


3 2 1


1 2 3
0 1 2
3 2 1
3 4 5


ODD
1 → 2 → 3 → 2 → 1 → X
Prev
mid
ODD
A → 1 2
0 1
B → 3 2 1
6 1 2
i
j
A → 1 → 2 → X → len(B) > len(A)
2
B → 3 → 2 → 1 → X
Reverse → 1 → 2 → 3 → X
While (-1) Tockeck

Even

Prev

6

1 → 2 → 3 → 3 → 2 → 1 → X

-> mid

A

B

1 → 2 → 3 → X

3 → 2 → 1 → X

RE

1 → 2 → 3 → X

Even ⇒ len(A) = len(B)

odd ⇒ len(A) > len(B)

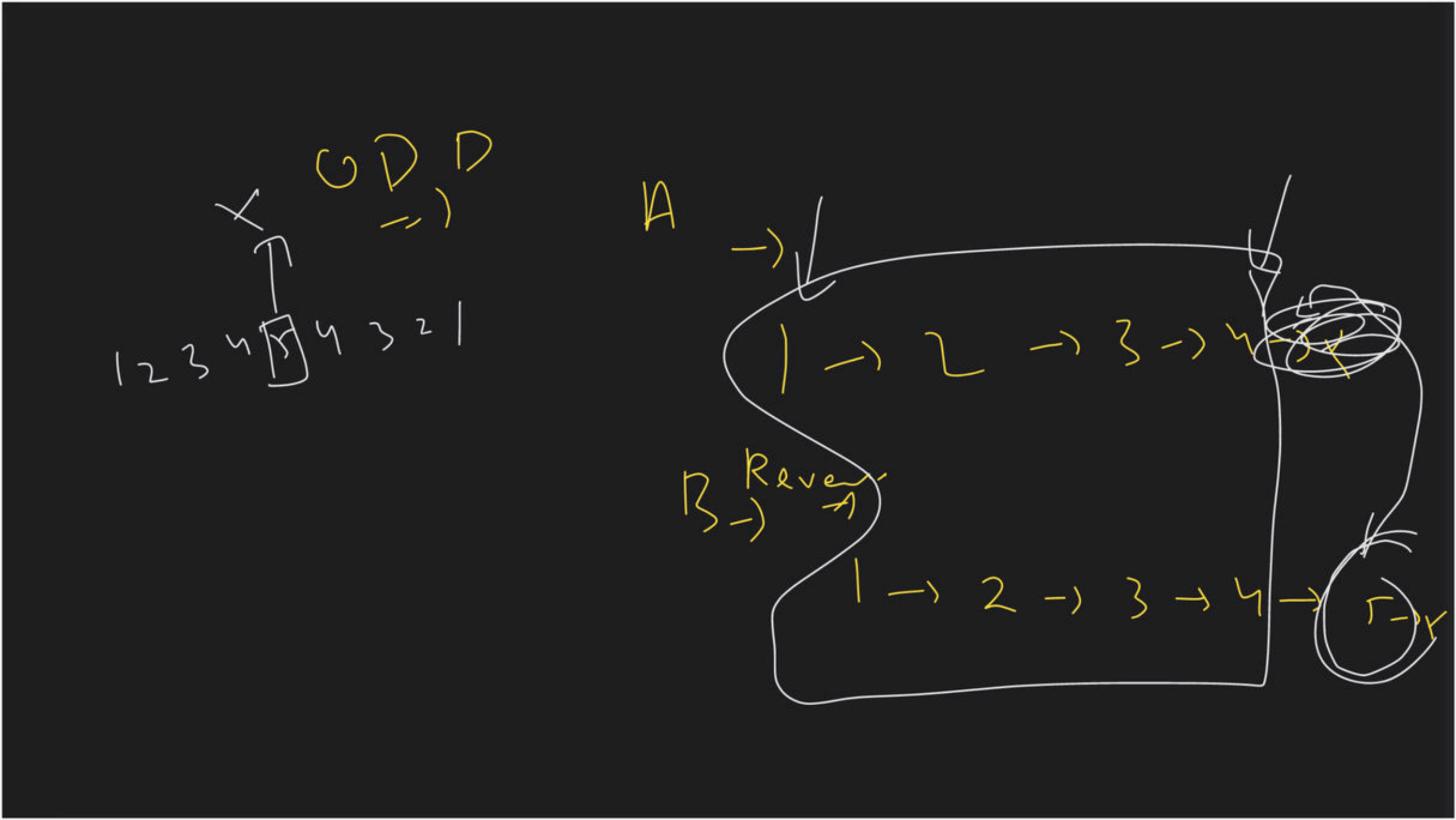
↓

len(A) ≤ len(B)

while (head A)

Rmk gye

}

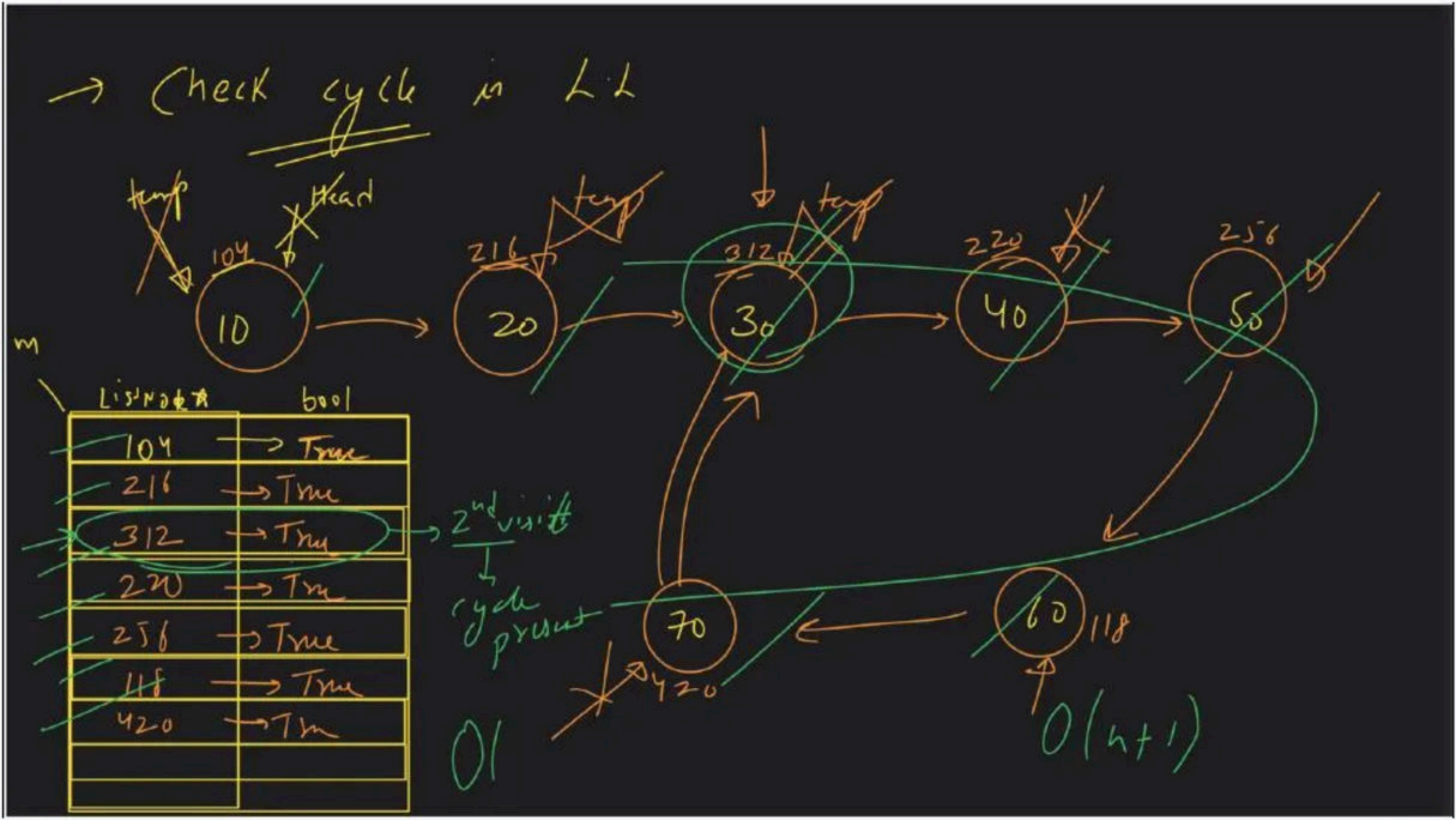

ODD
X
1 2 3 4 5 4 3 2 1
A
1 → 2 → 3 → 4 → X
B
Reverse
1 → 2 → 3 → 4 → Y

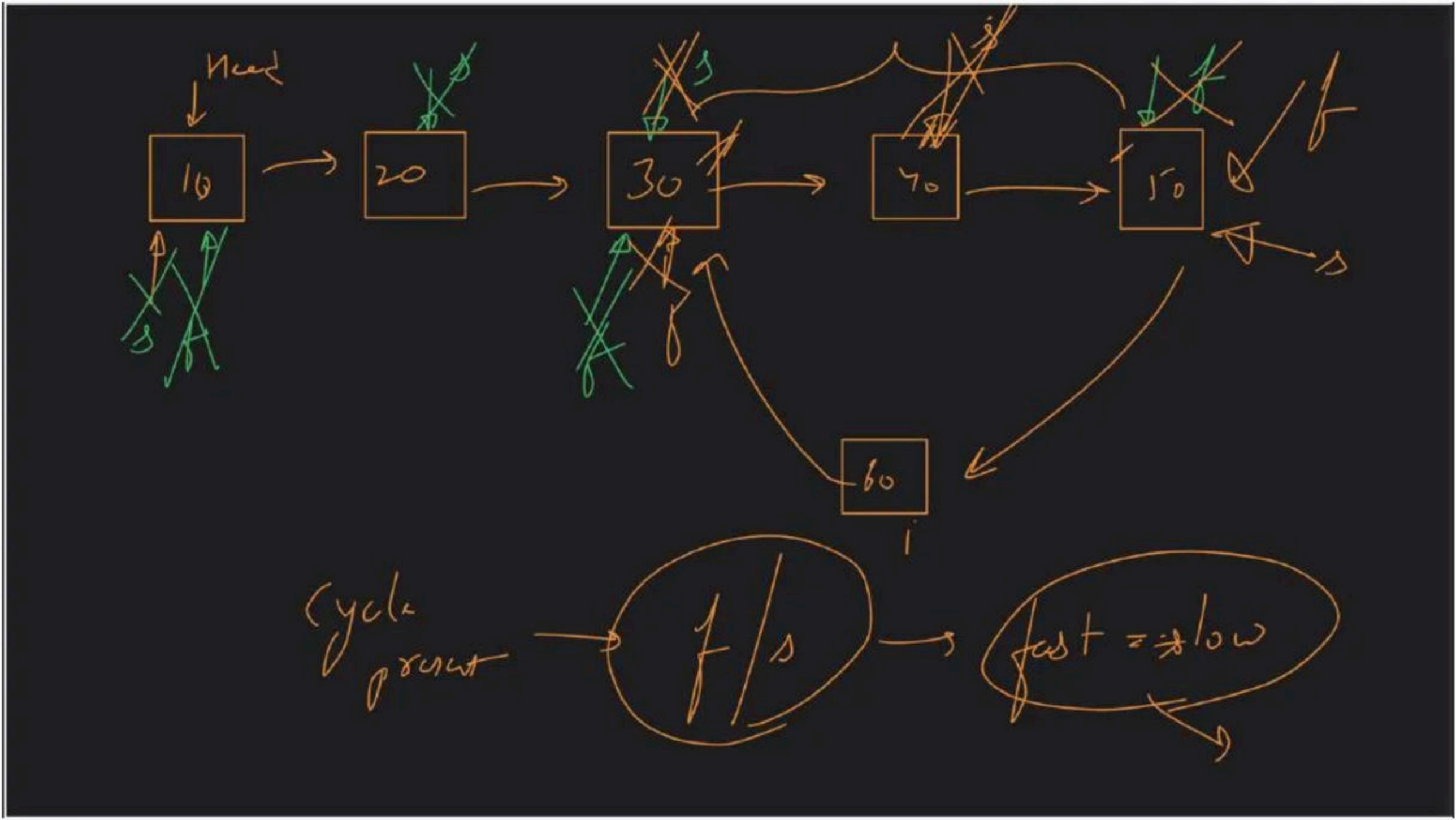
we will continue, agar K question

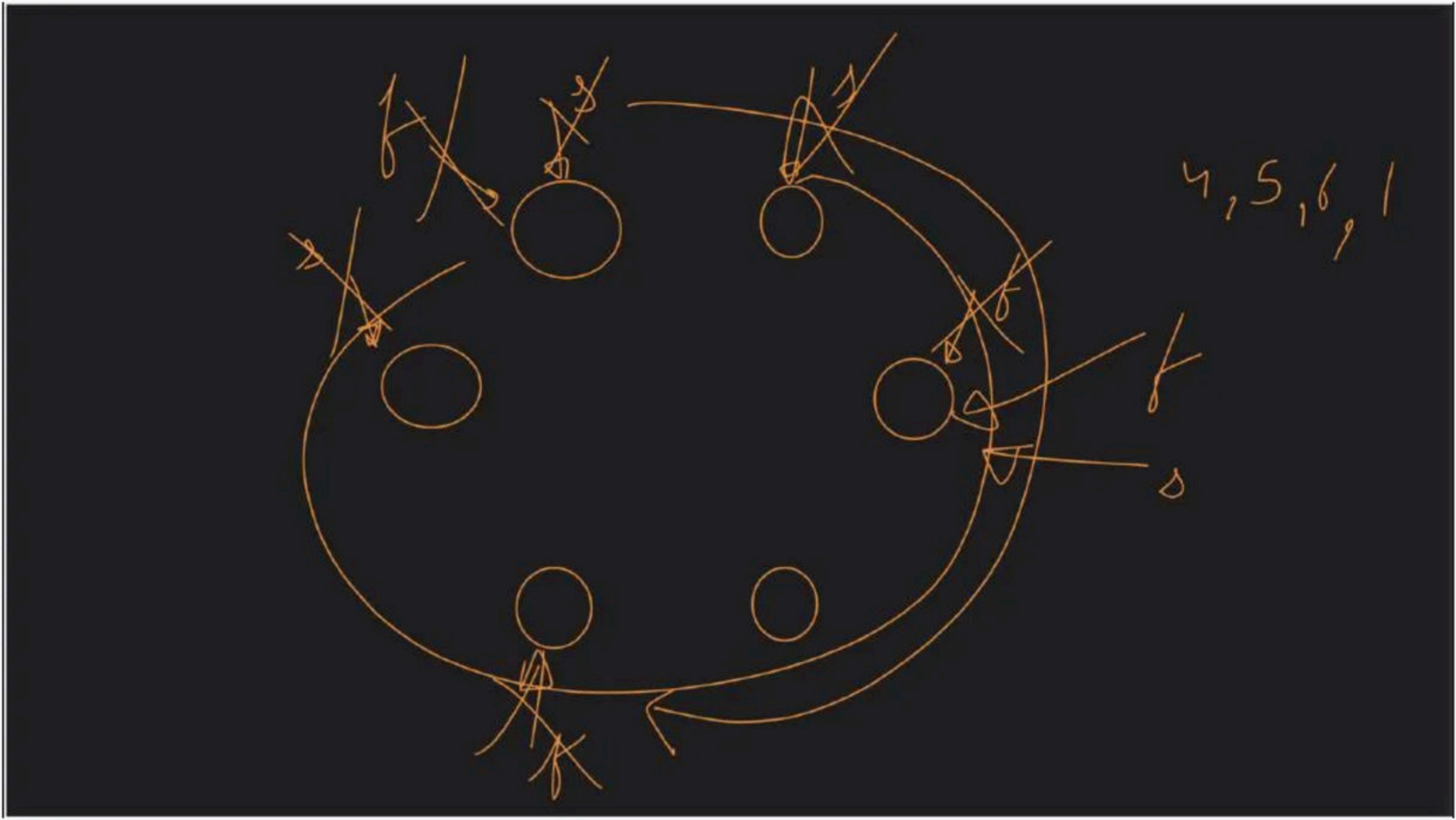
Kal raat 8:30 pm

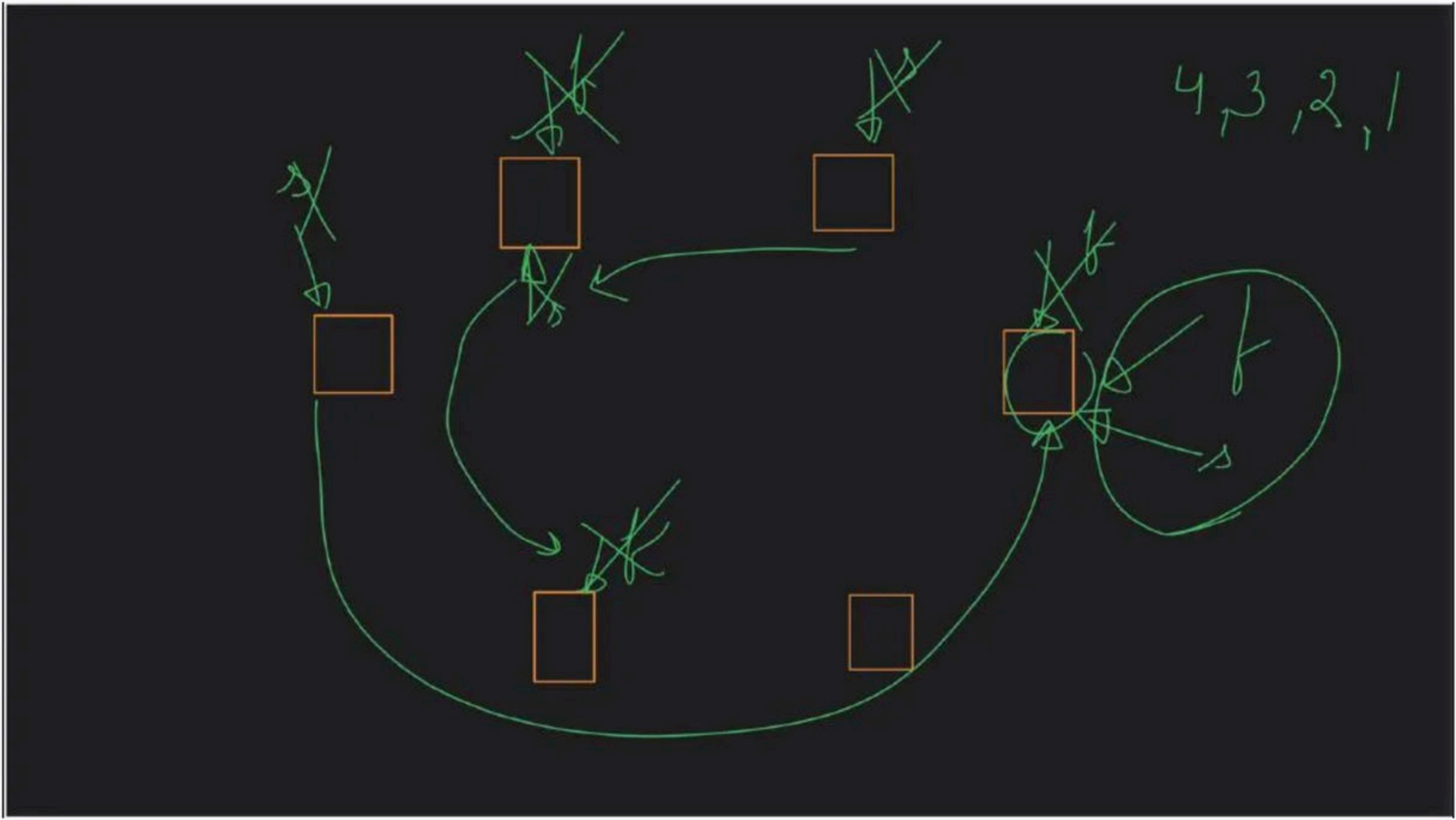
Bye Bye → love you all

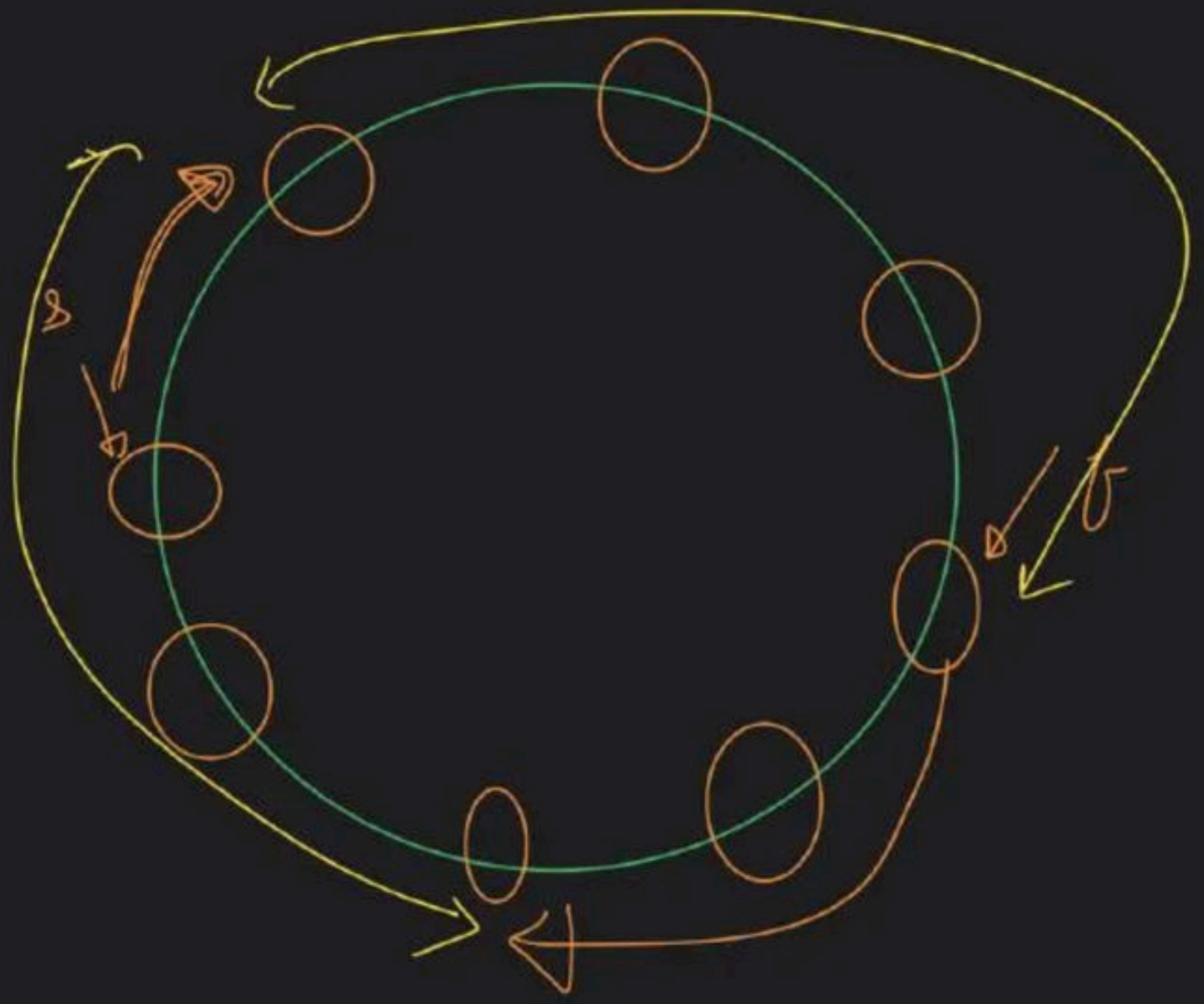

Love Babbar
Linked List Class - 4
Special class
Love Babbar • July 19, 2024

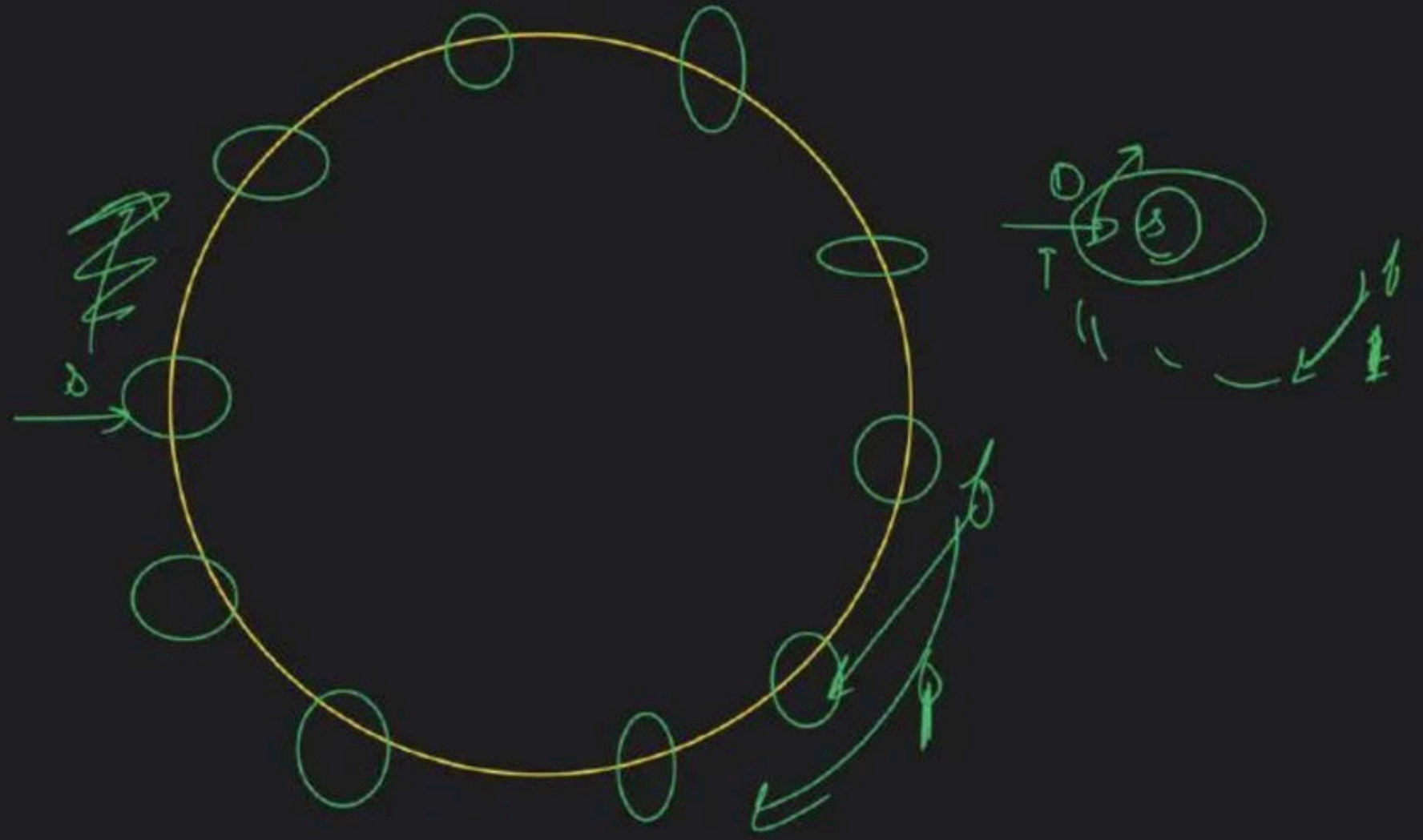

→ Check cycle in LL
temp
head
104
10
216
20
312
30
220
40
256
50
m
ListNode*
bool
104 → True
216 → True
312 → True
220 → True
256 → True
118 → True
420 → True
2nd visit
cycle present
70
60
118
420
O(1)
O(n+1)

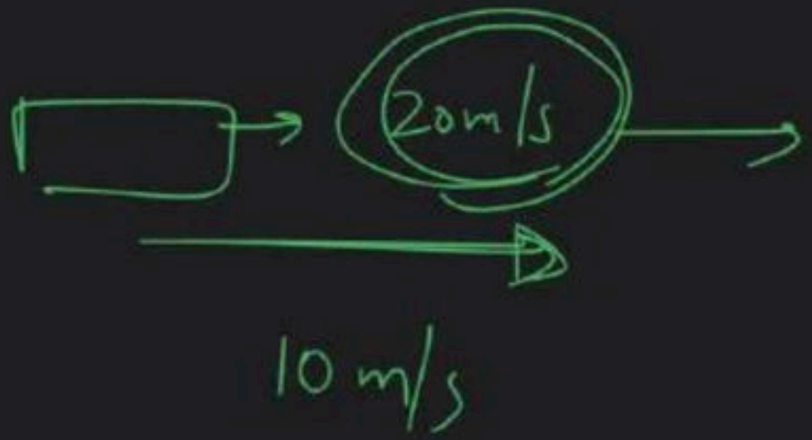

Head
10
20
30
40
50
60
Cycle present
F/S
fast == slow

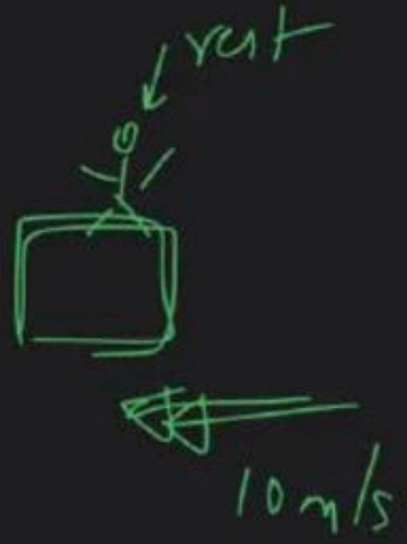

4, 5, 6, 1

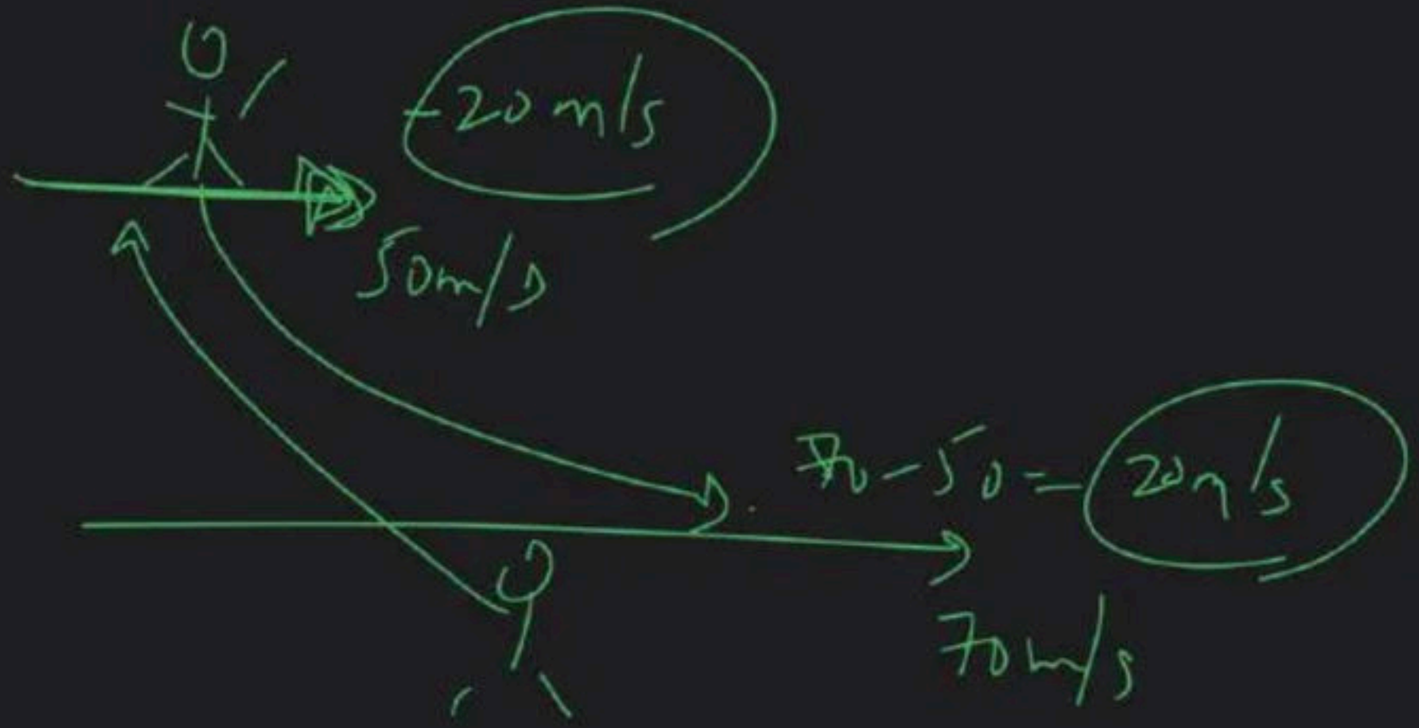

4,3,2,1






20 m/s
10 m/s


rest
10 m/s


20 m/s
50m/s
70 - 50 = 20 m/s
70m/s

temp -> data != prev -> data

↳ prev → 1st

temp → 1st

temp => NULL

Return.

- (I) LL is empty

return Head

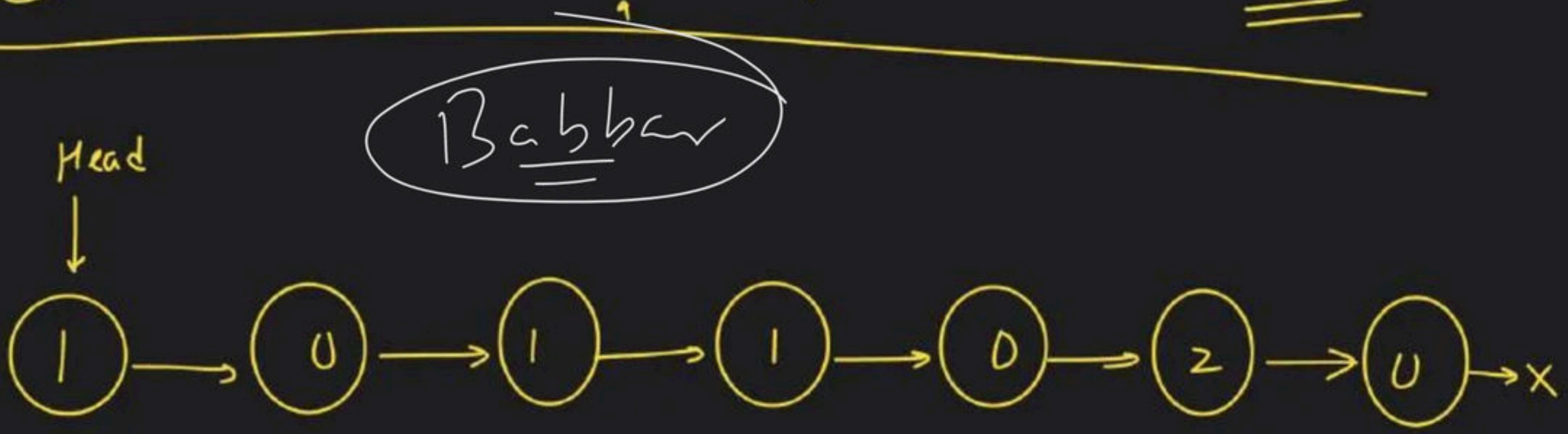
- (II) single Node

return head

→ (III) >= 1 node

→ Sort 0's, 1's, 2's in L.L

Babbar



Approach → counting → New L L → T.C → O(n) S.C -> O(n)

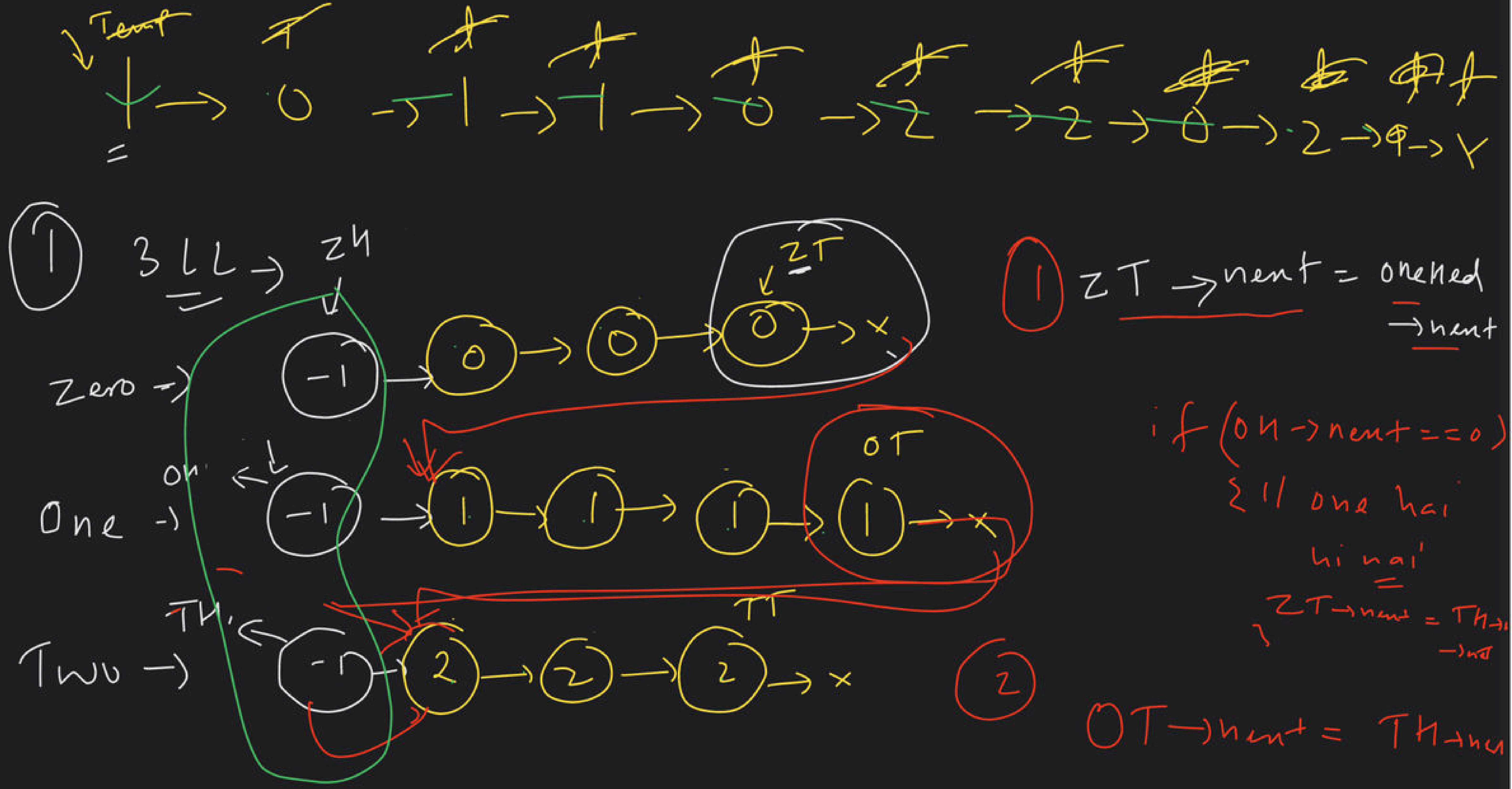
Approach -> count, → overwrite

Approach → Pointers K sath Khelna padega

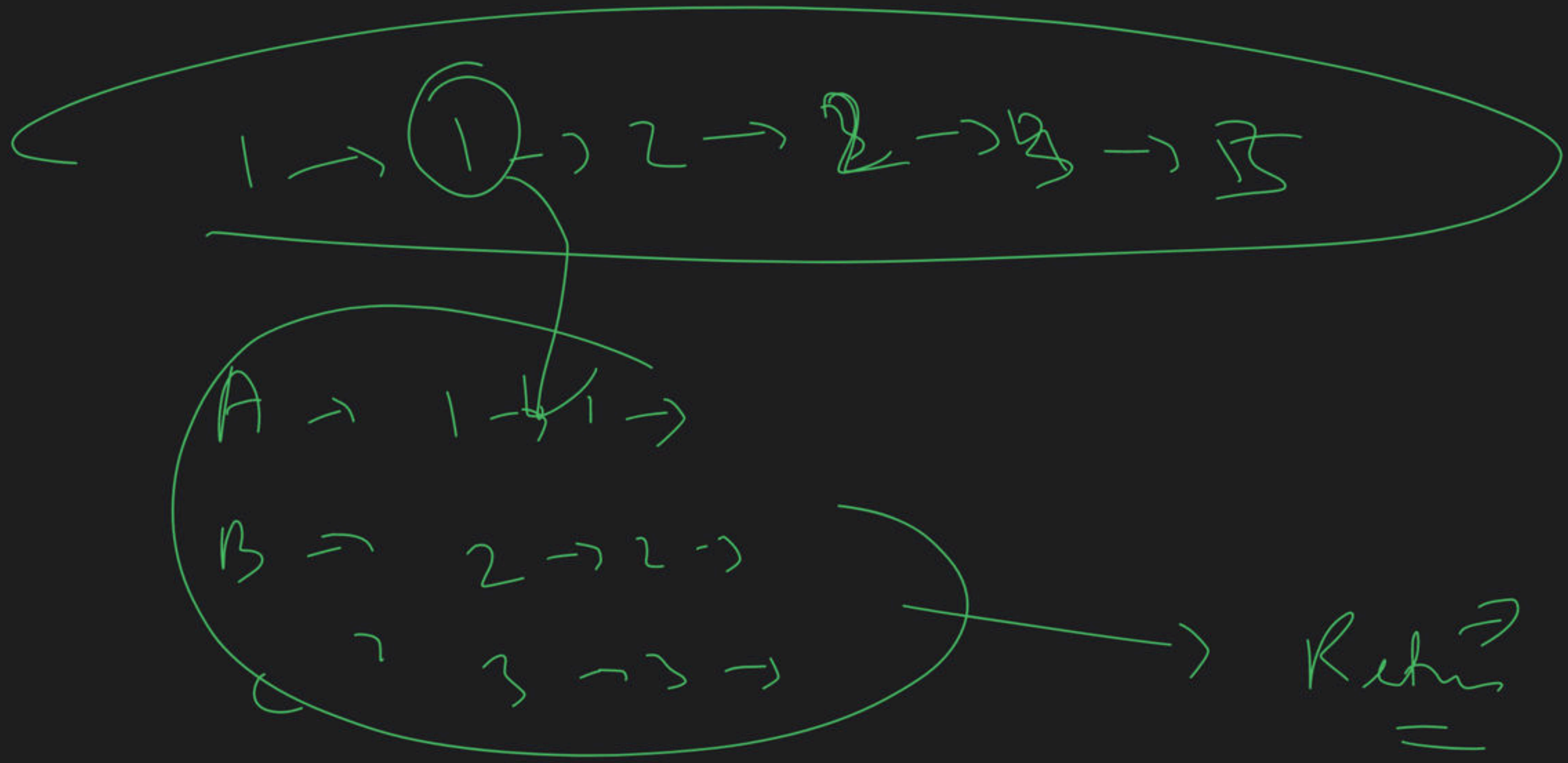
1 → 0 → 1 → 1 → 0 → 2 → 0 → x

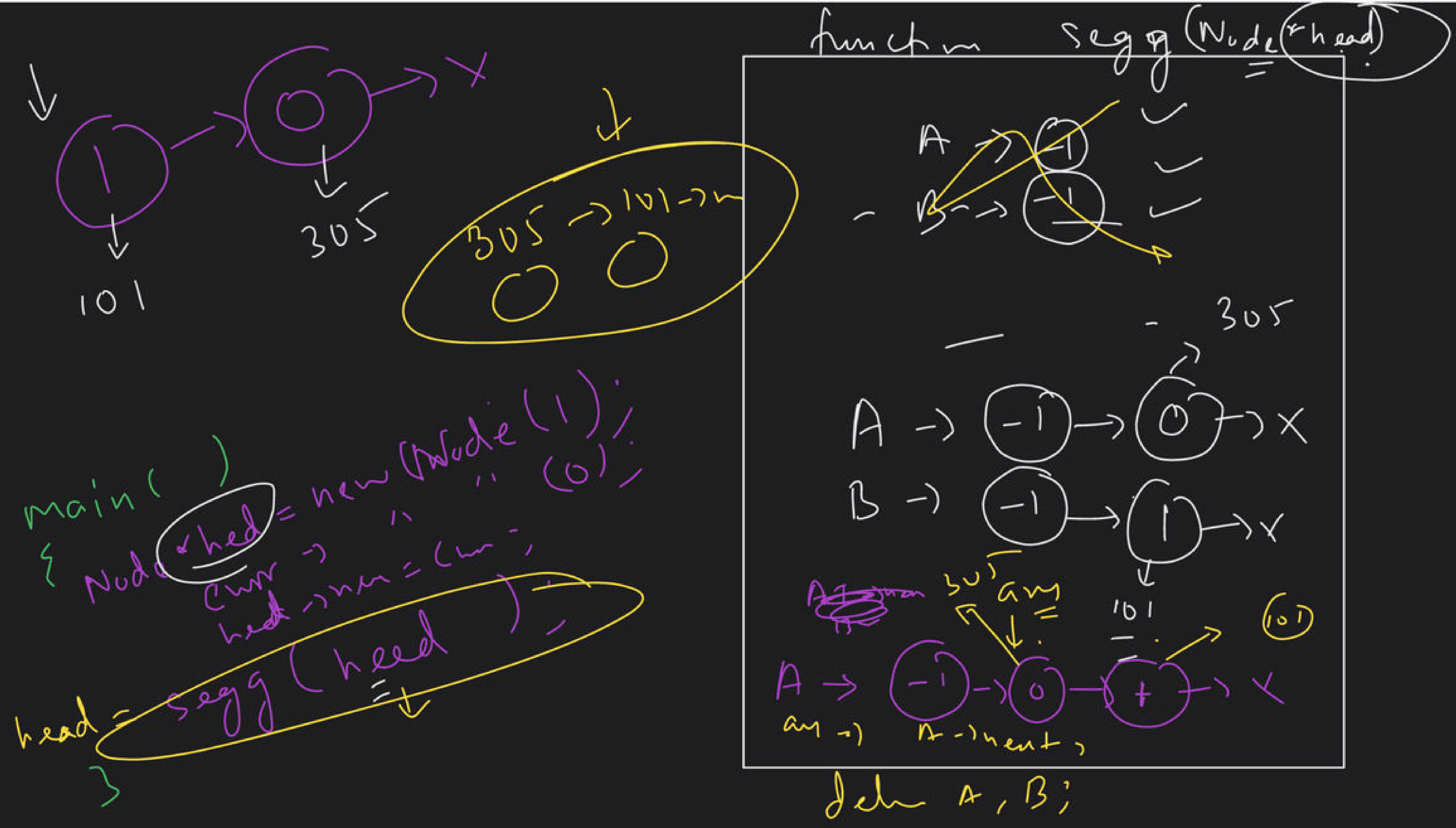
Approach ⇒

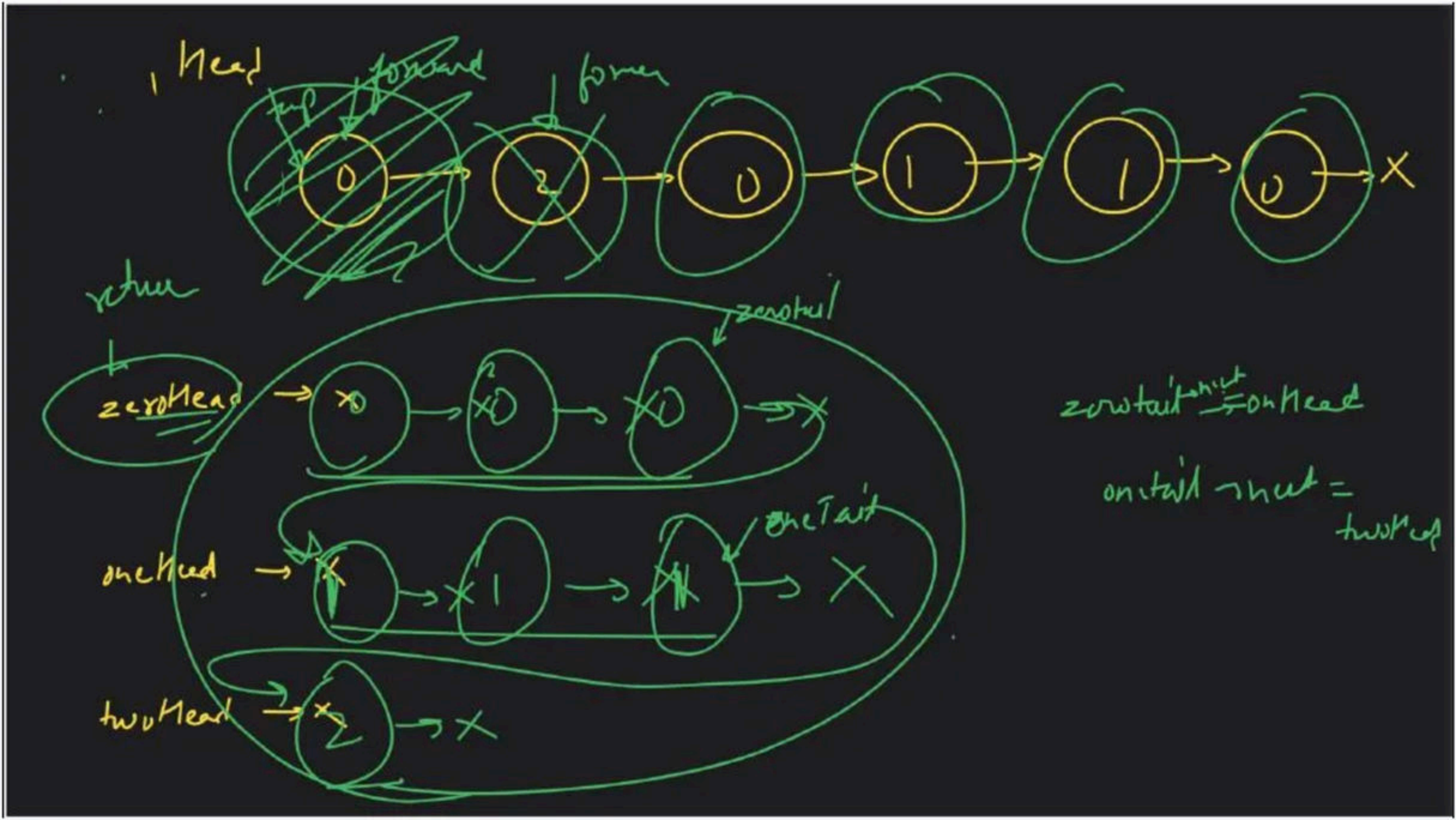
1. Make 3 LL of 0, 1, 2, respectively
2. Merge them
3. Done

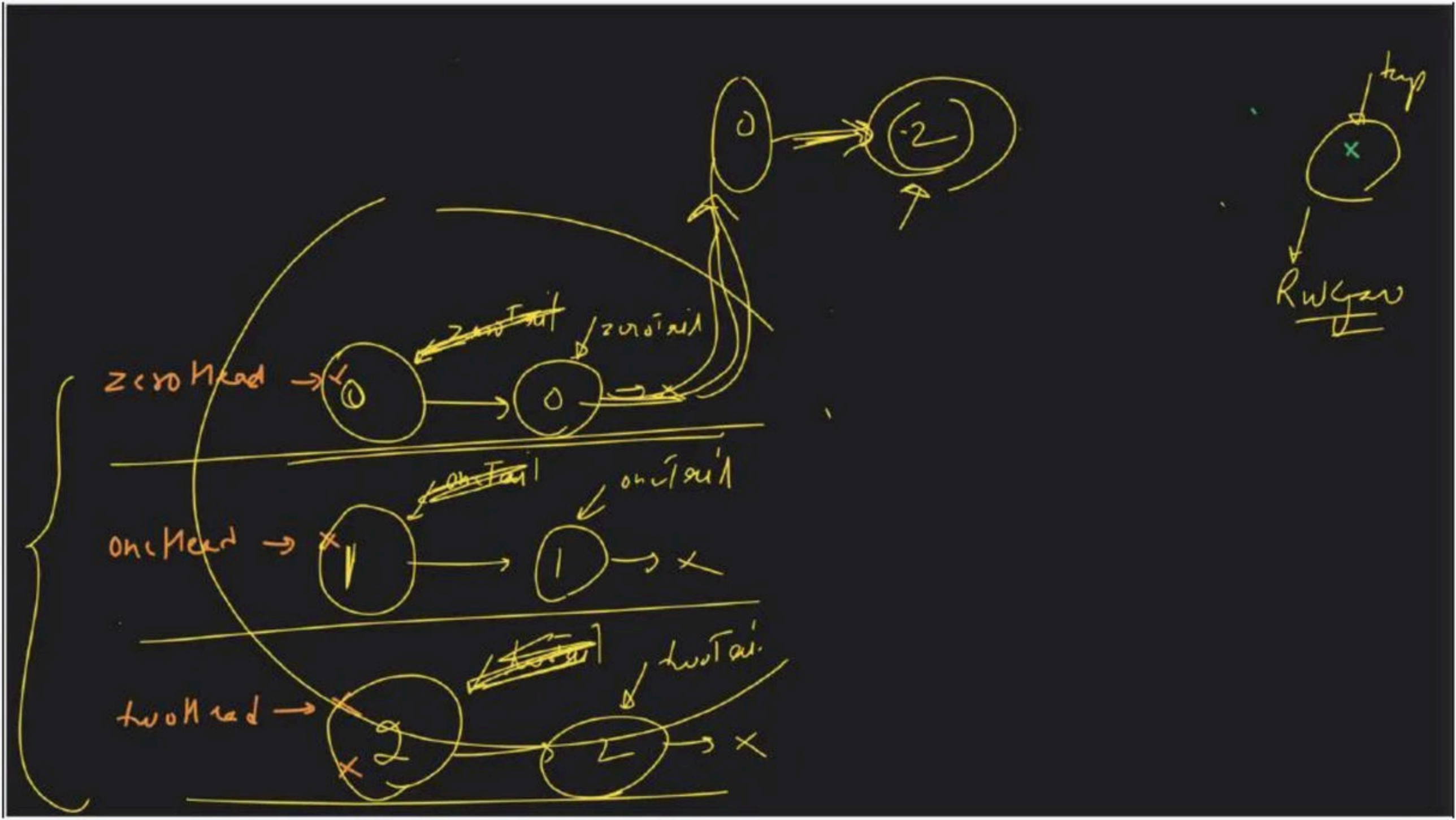


Temp
1 → 0 → 1 → 1 → 0 → 2 → 2 → 0 → 2 → φ → Y
① 3 LL → ZH
Zero → -1 → 0 → 0 → 0 → X
ZT
One → -1 → 1 → 1 → 1 → 1 → X
OH
OT
Two → -1 → 2 → 2 → 2 → X
TH
TT
① ZT → next = oneHead → next
if (OH → next == 0)
{ // one hai ki nai
ZT → next = TH → next
}
② OT → next = TH → next


1 → 1 → 2 → 3 → 5
A → 1 → 1 →
B → 2 → 2 →
3 → 3 →


function segg(Node *head)
A
B
305
101
305 -> 101 -> n
A -> -1 -> 0 -> X
B -> -1 -> 1 -> X
ans
A -> -1 -> 0 -> 1 -> X
ans -> A->next;
dele A, B;
main( )
{ Node *hed = new Node(1);
curr -> " (0)
head->next = curr;
head = segg(heed)
}


head
fast
forward
former
0
2
0
1
1
0
X
return
zeroTail
zeroHead
0
0
0
X
oneTail
oneHead
1
1
X
twoHead
2
X
zeroTail.next == oneHead
oneTail.next =
twoHead

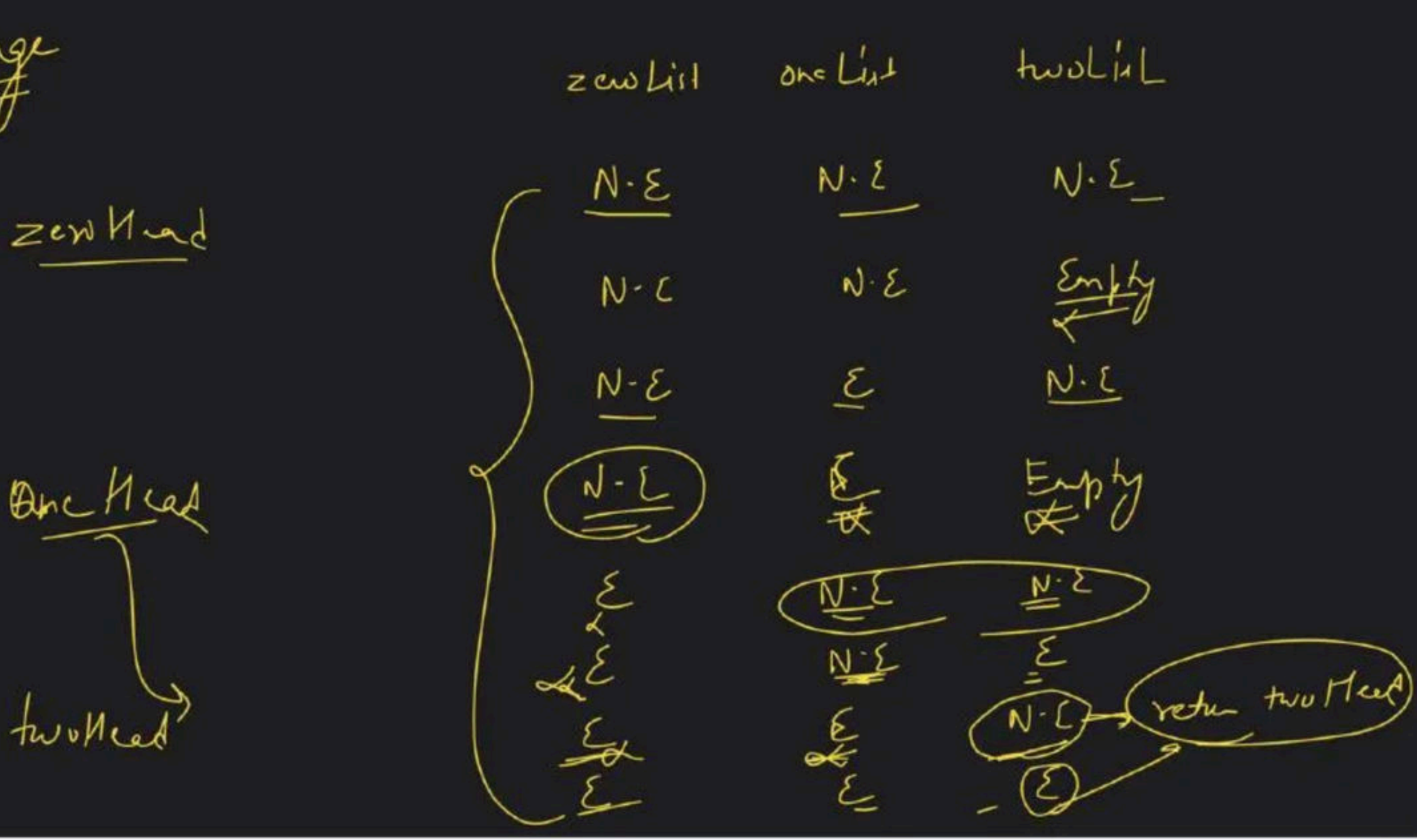

0
2
temp
x
RWLock
zeroHead
zeroTail
oneHead
oneTail
twoHead
twoTail
1
2
x

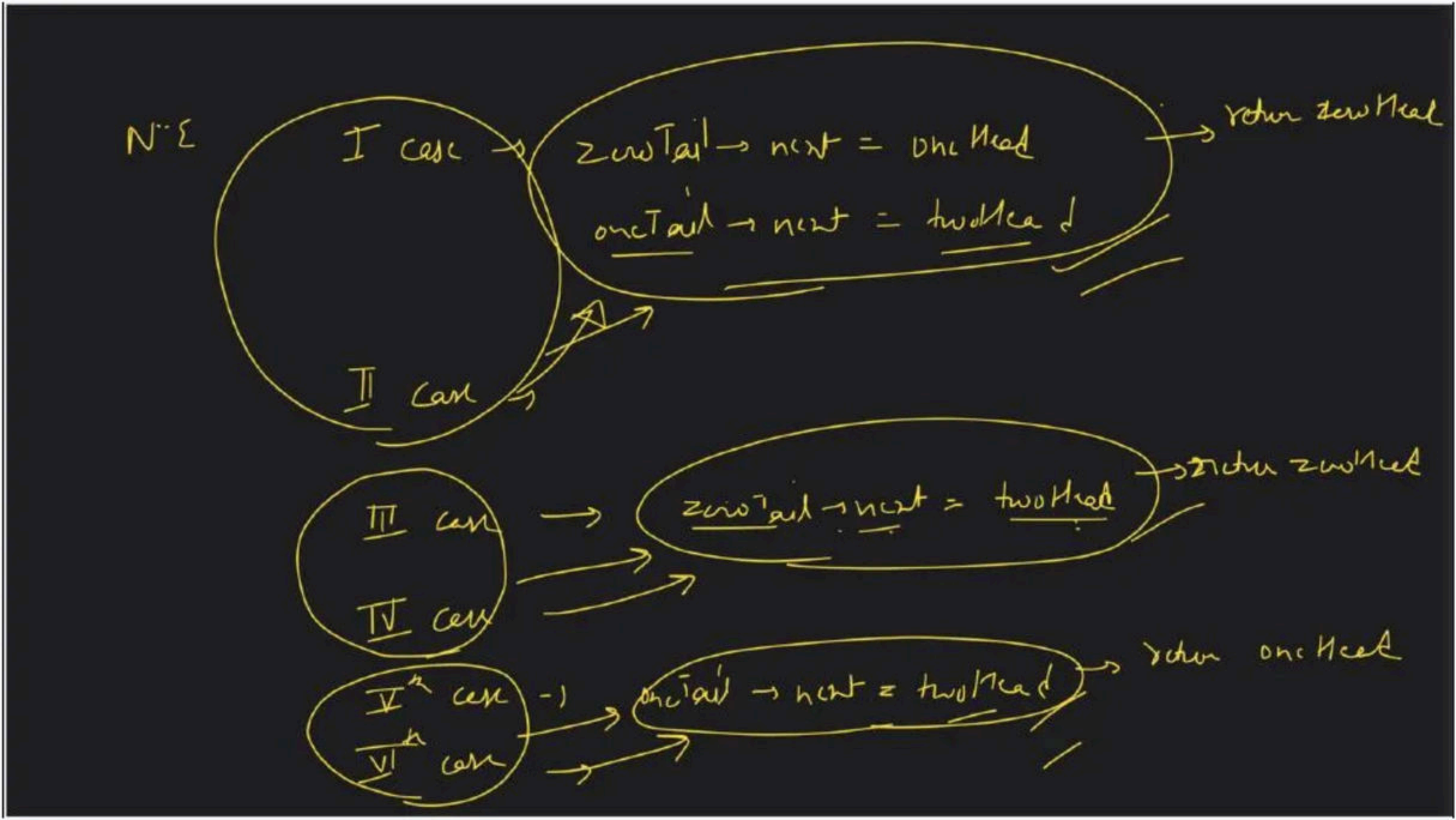
Merge

zeroHead

oneHead

twoHead




N-1
I case
II case
zeroTail → next = oneHead
oneTail → next = twoHead
return zeroHead
III case
IV case
zeroTail → next = twoHead
return zeroHead
V^k case - 1
VI^k case
oneTail → next = twoHead
return oneHead

→ Add 1 to Linked List :→

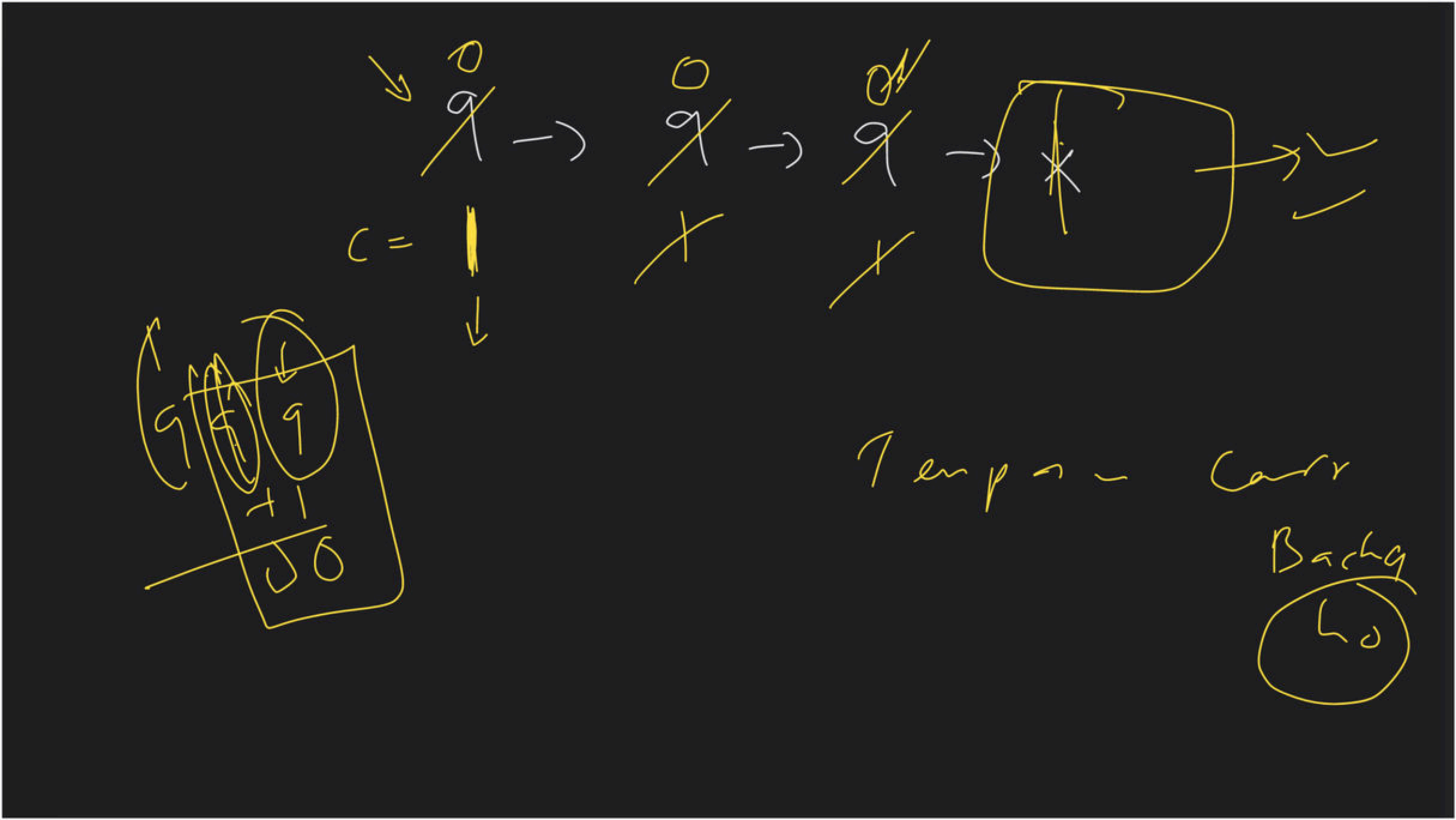
2 min

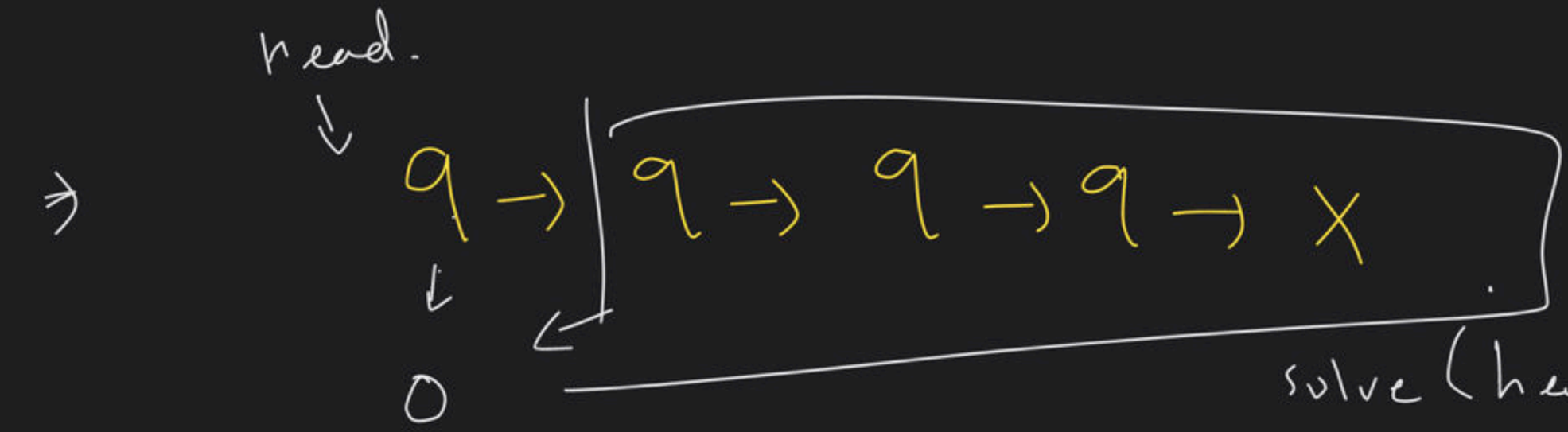
approach

(1) convert karlo integer → +1 → convert LL ∠

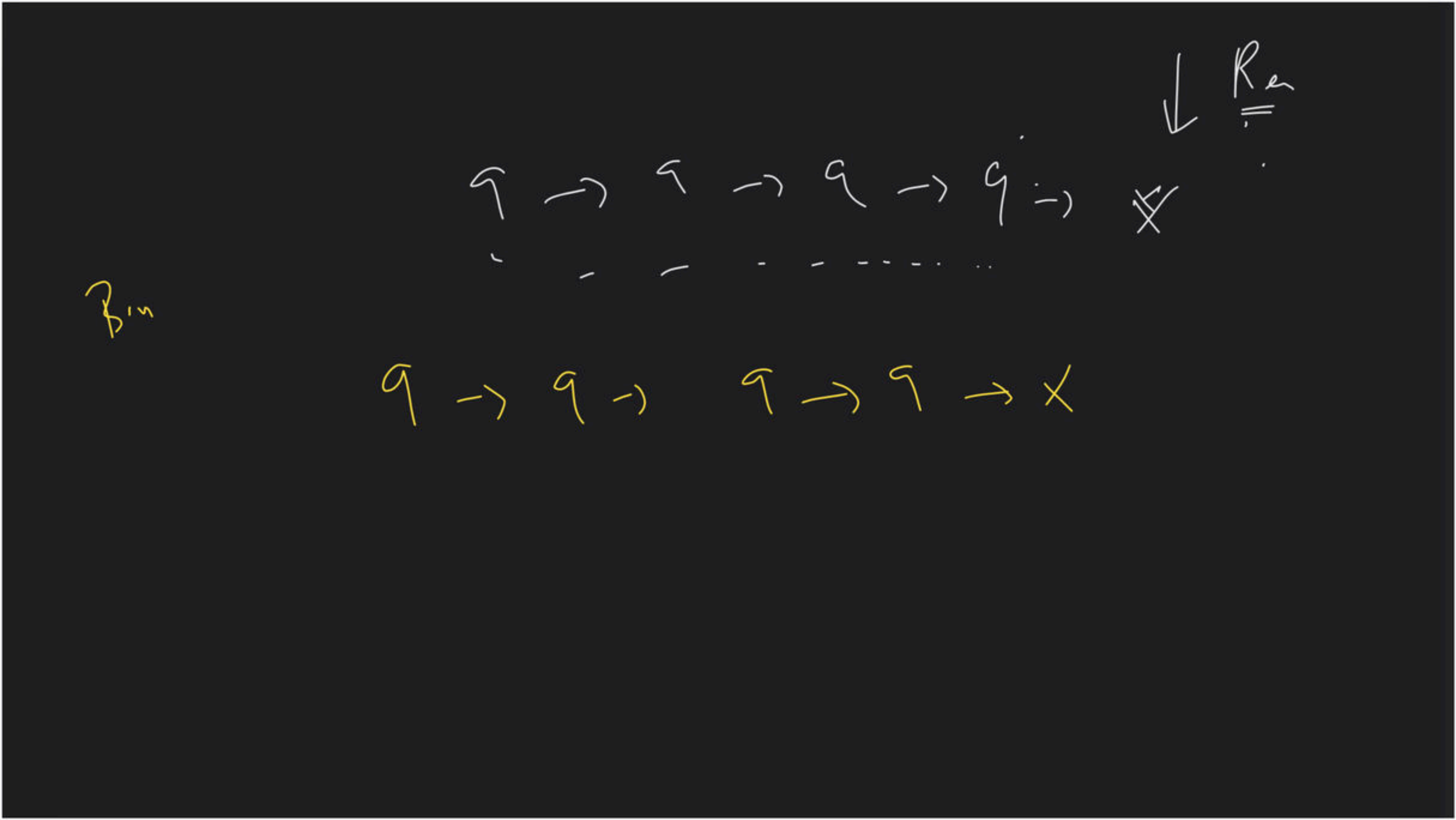
(2) convert karlo string → +1 → convert LL ∠

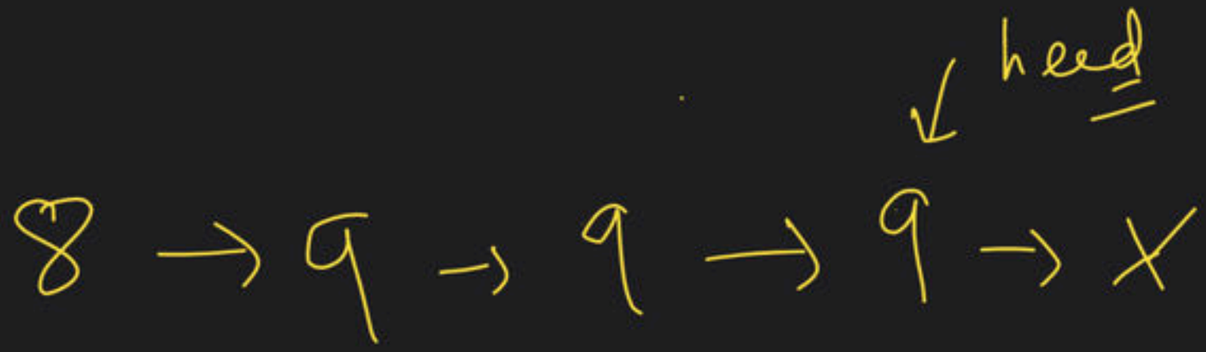
(3) reverse karke → +1 → reverse


Temp → Carry
Backg




```
Cal =
    Sum = carry + head -> data;
    digit => Sum %. 10;
    Carry -> sum/10;
    Head -> data = digit;
head->next = solve (head->next, car)
```


R_n
$q \to q \to q \to q \to$
Σ_{in}
$q \to q \to q \to q \to X$



0 #1

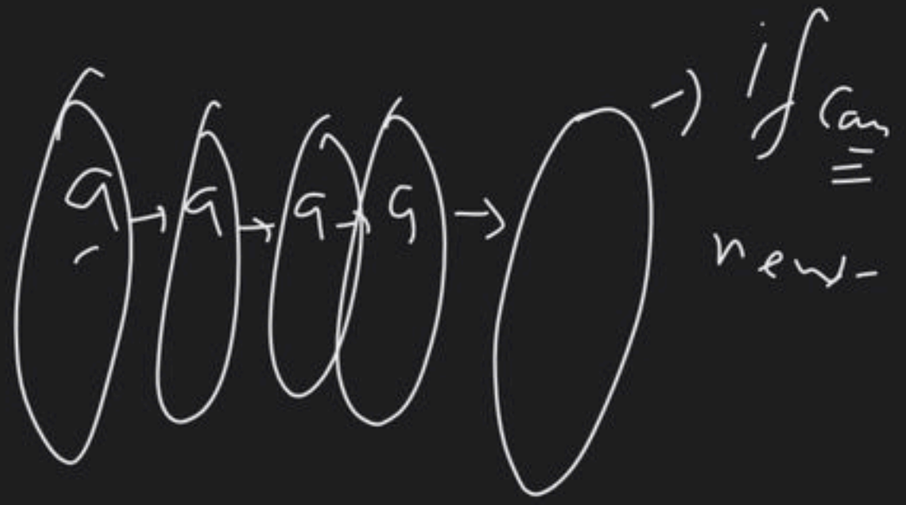
Mega Clan

By Ref

① Reverse

② add → Rev

③ Rever



$q_1, q_2, q_3, q \to Y$

$0 \to 0 \to 0 \to 0 \to X$

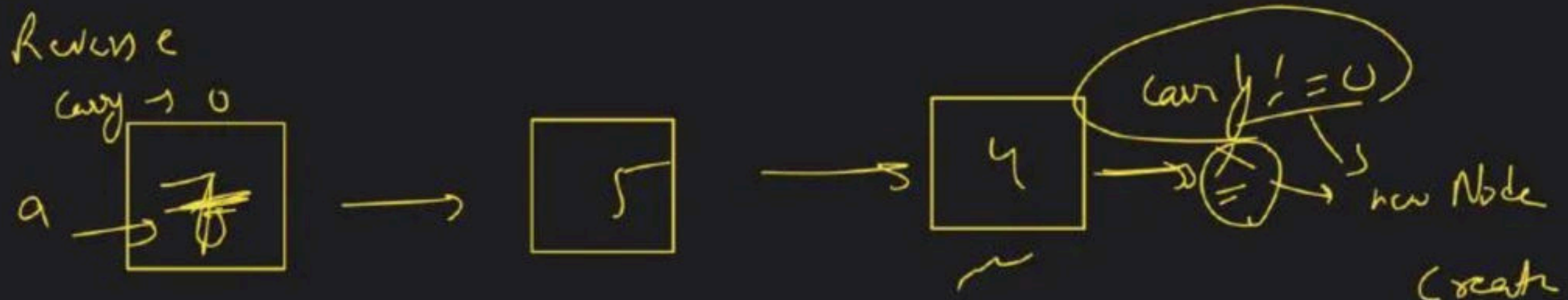
```
main()
{
    Reverse(head, carry);
    if(carry)
    {
        insert at head
    }
}
```

By Ref.

Reverse

(1) Reverse

carry → 0



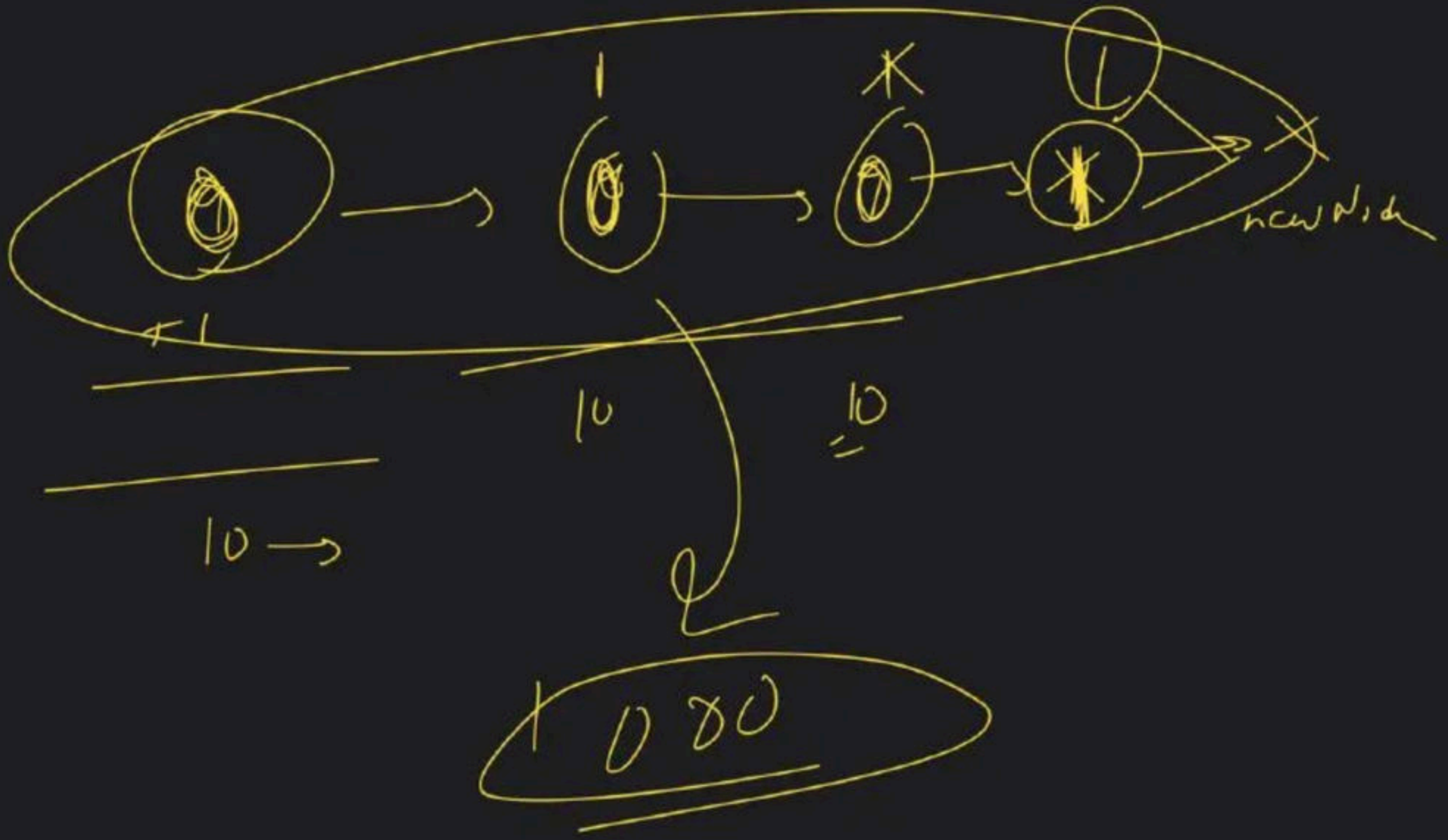
b → 1

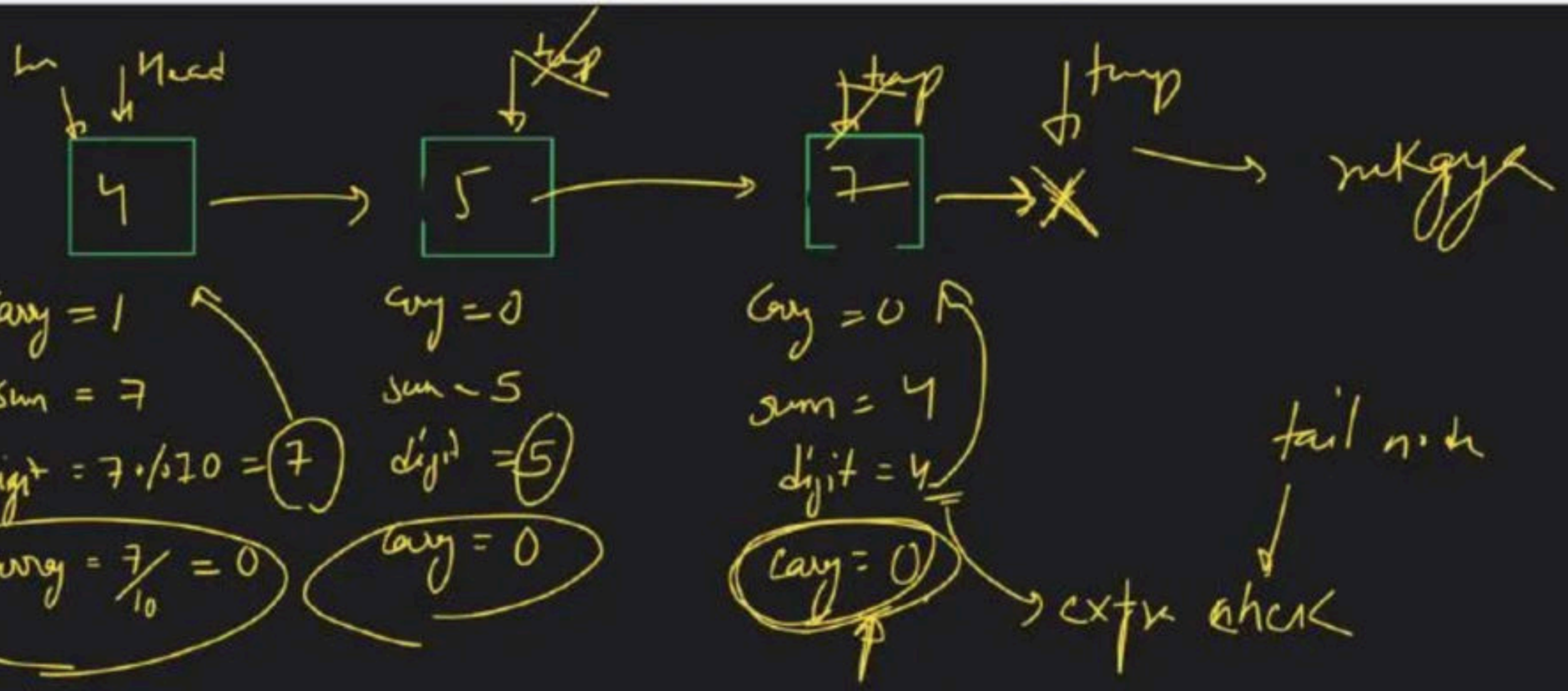
sum = (7), carry → 0

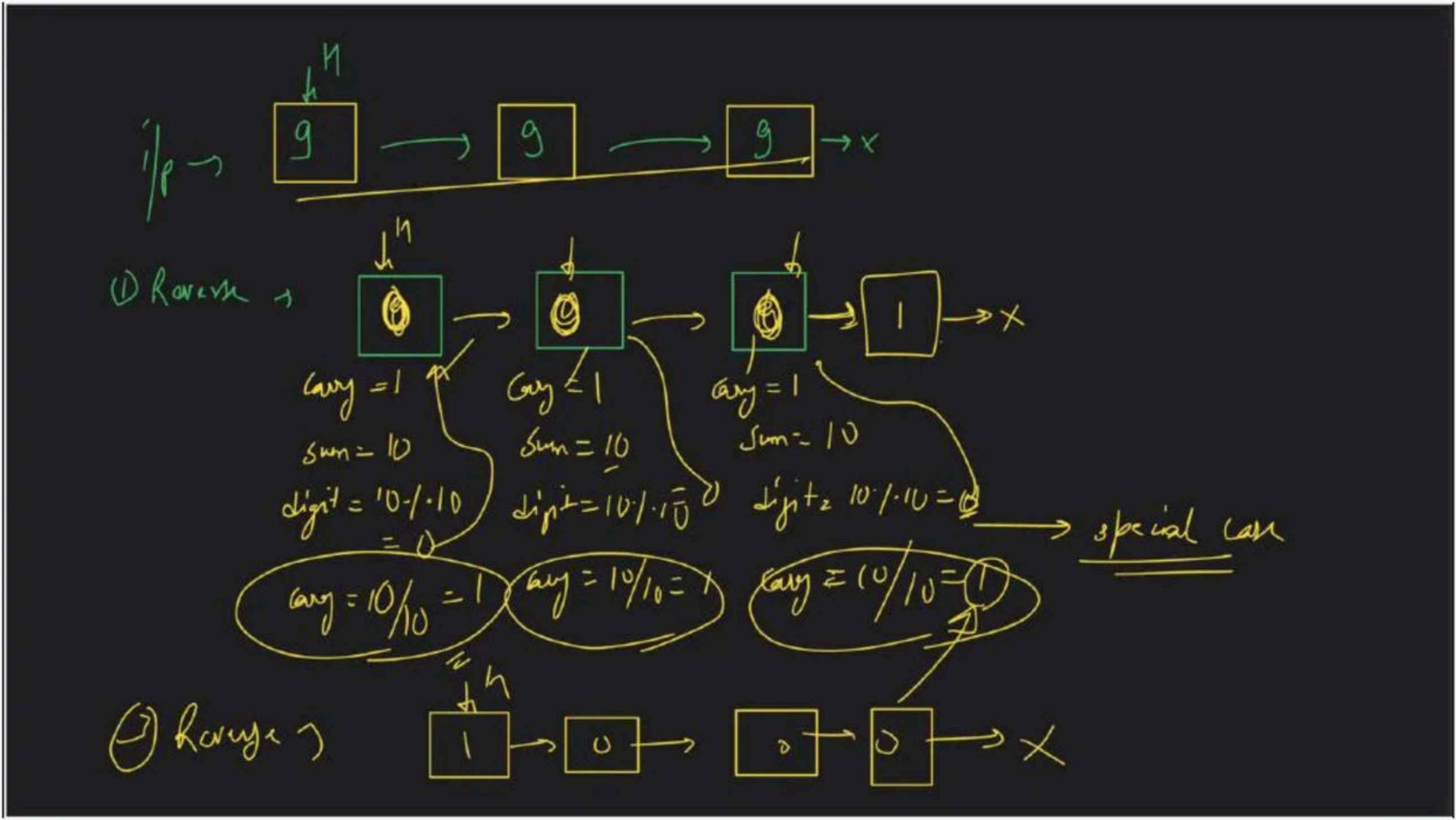
↳ aage kuch

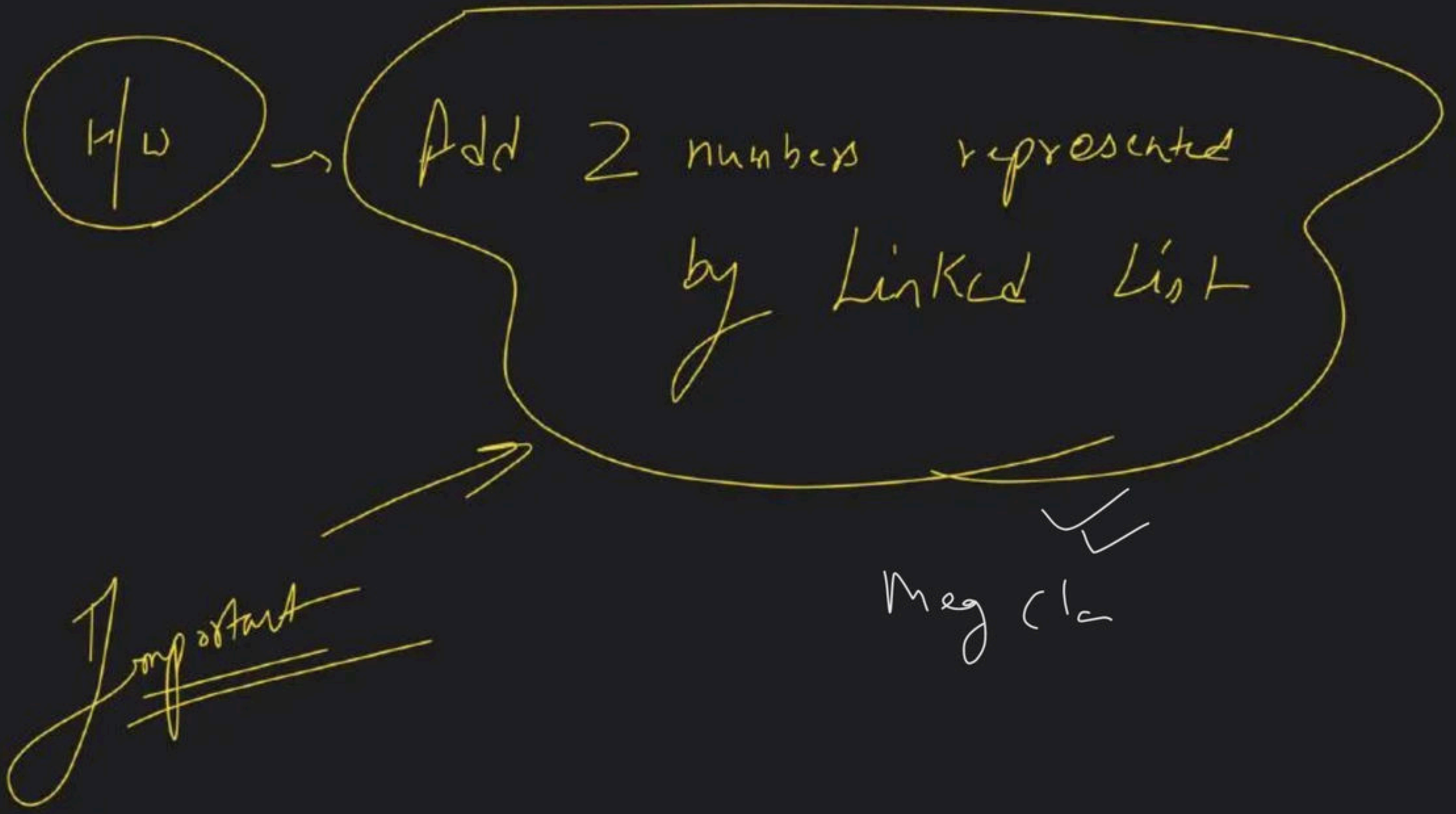
carry != 0 → next node pr plus karo

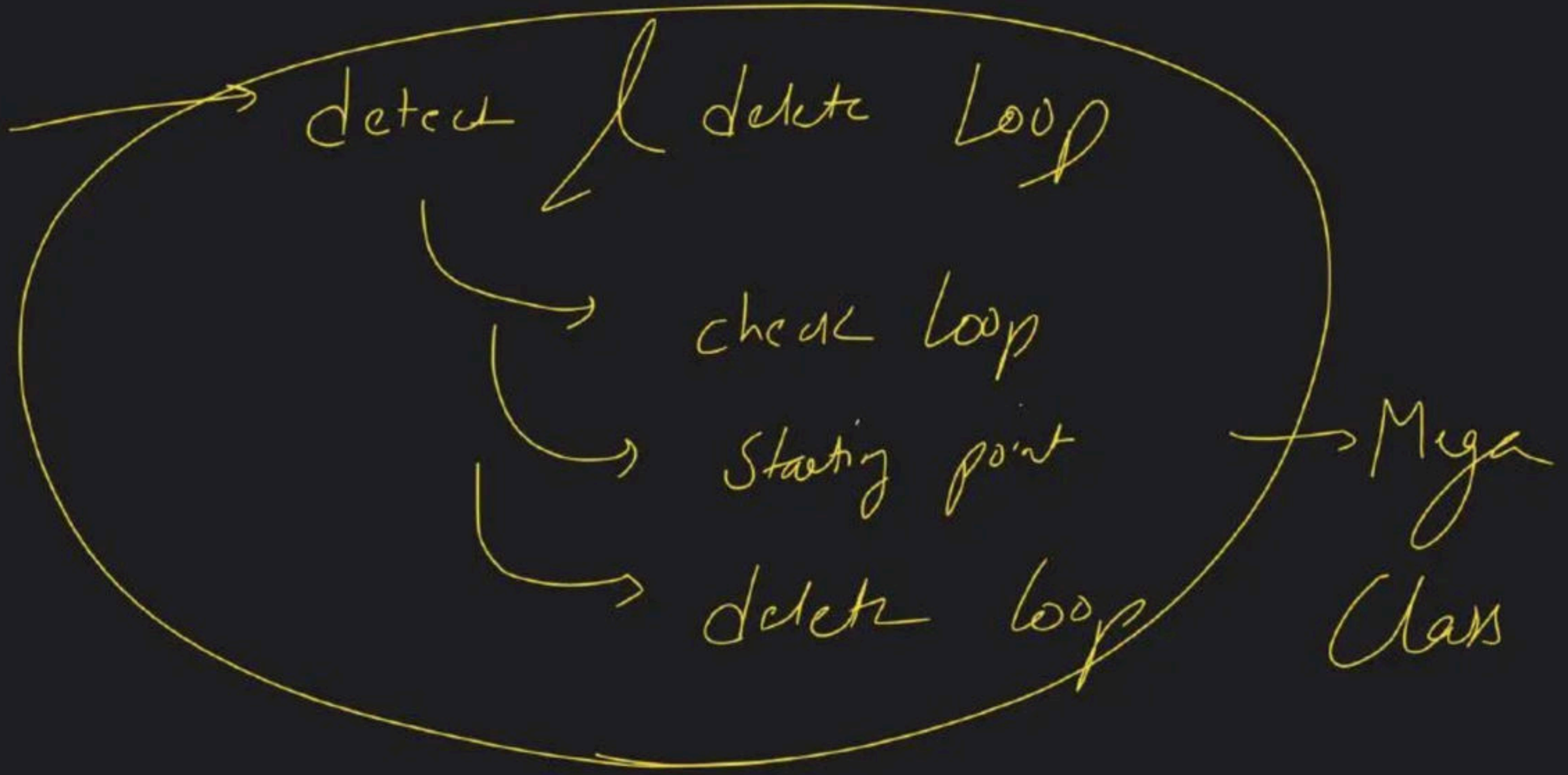
(3) Reverse


1
*
0
0
0
1
*
new Node
1
10
10
10 →
1 0 0


l1
Head
temp
temp
temp
4
5
7
ruk gaya
carry = 1
sum = 7
digit = 7%10 = 7
carry = 7/10 = 0
carry = 0
sum = 5
digit = 5
carry = 0
carry = 0
sum = 4
digit = 4
carry = 0
extra check
tail node

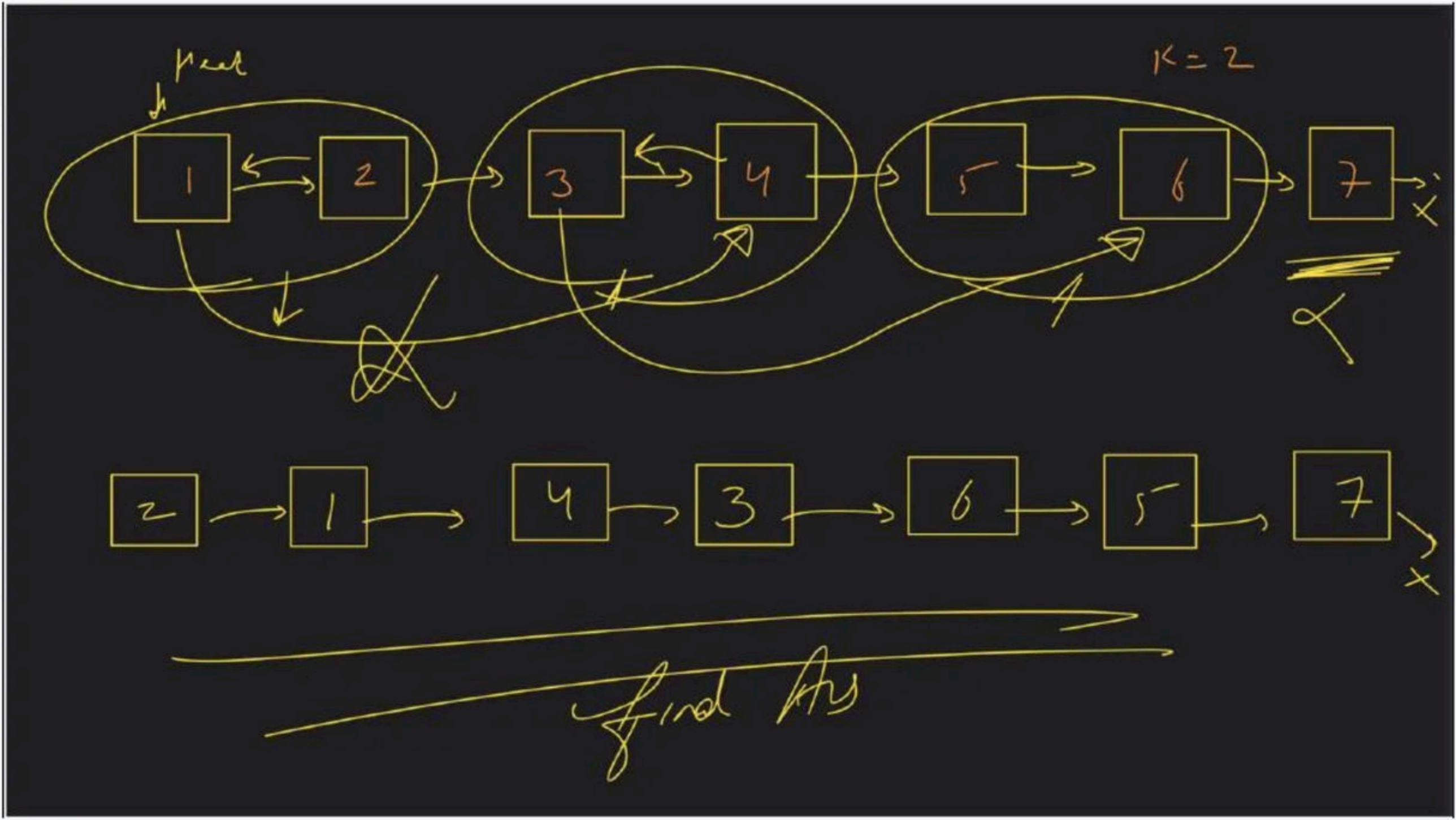

i/p →
9
9
9
→ X
① Reverse →
0
0
0
1
→ X
carry = 1
sum = 10
digit = 10 % 10 = 0
carry = 10/10 = 1
carry = 1
sum = 10
digit = 10 % 10 = 0
carry = 10/10 = 1
carry = 1
sum = 10
digit = 10 % 10 = 0
carry = 10/10 = 1
special case
② Reverse →
1
0
0
0
→ X

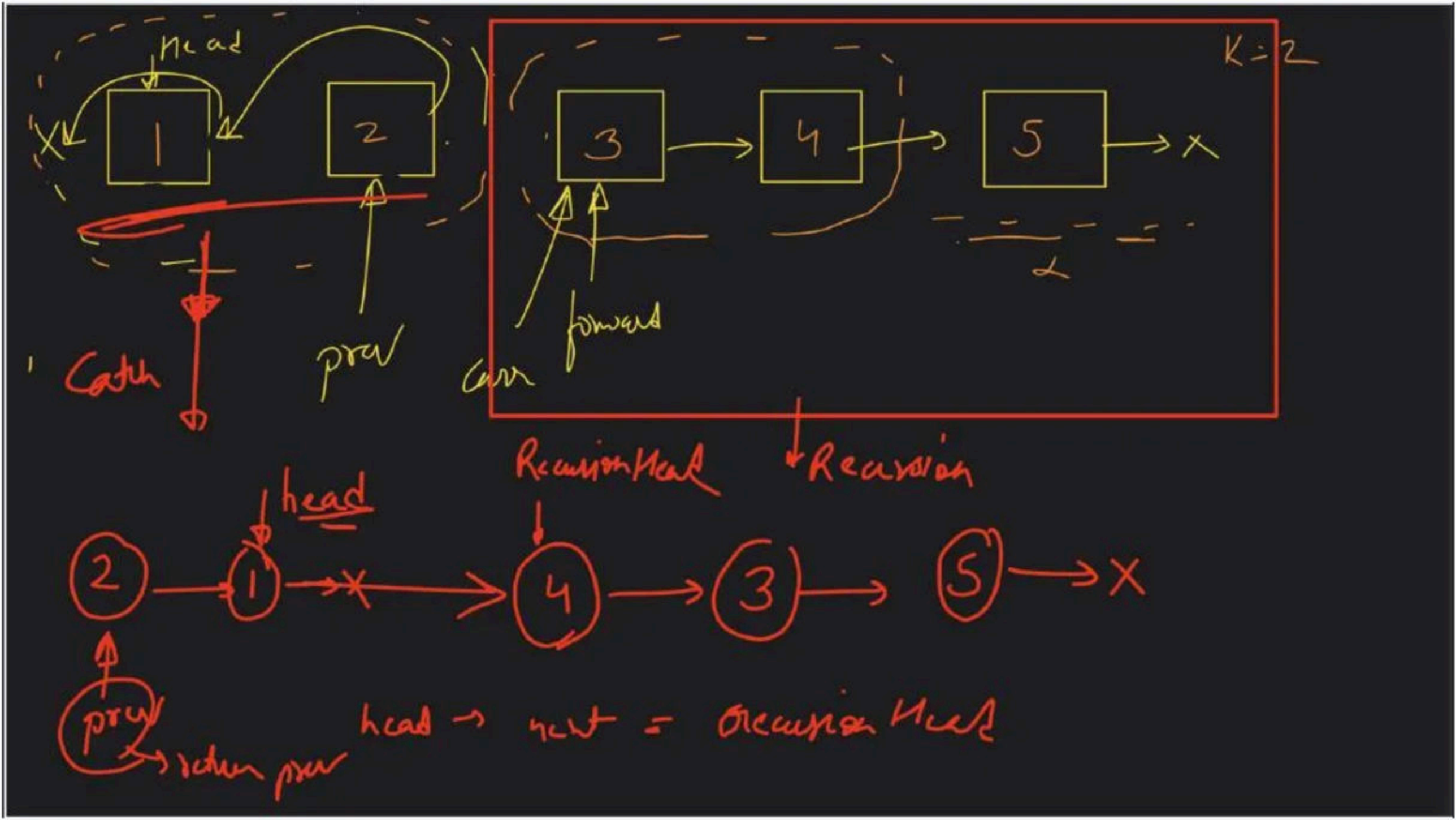

H/W
Add 2 numbers represented
by Linked List
Important
Mag cl

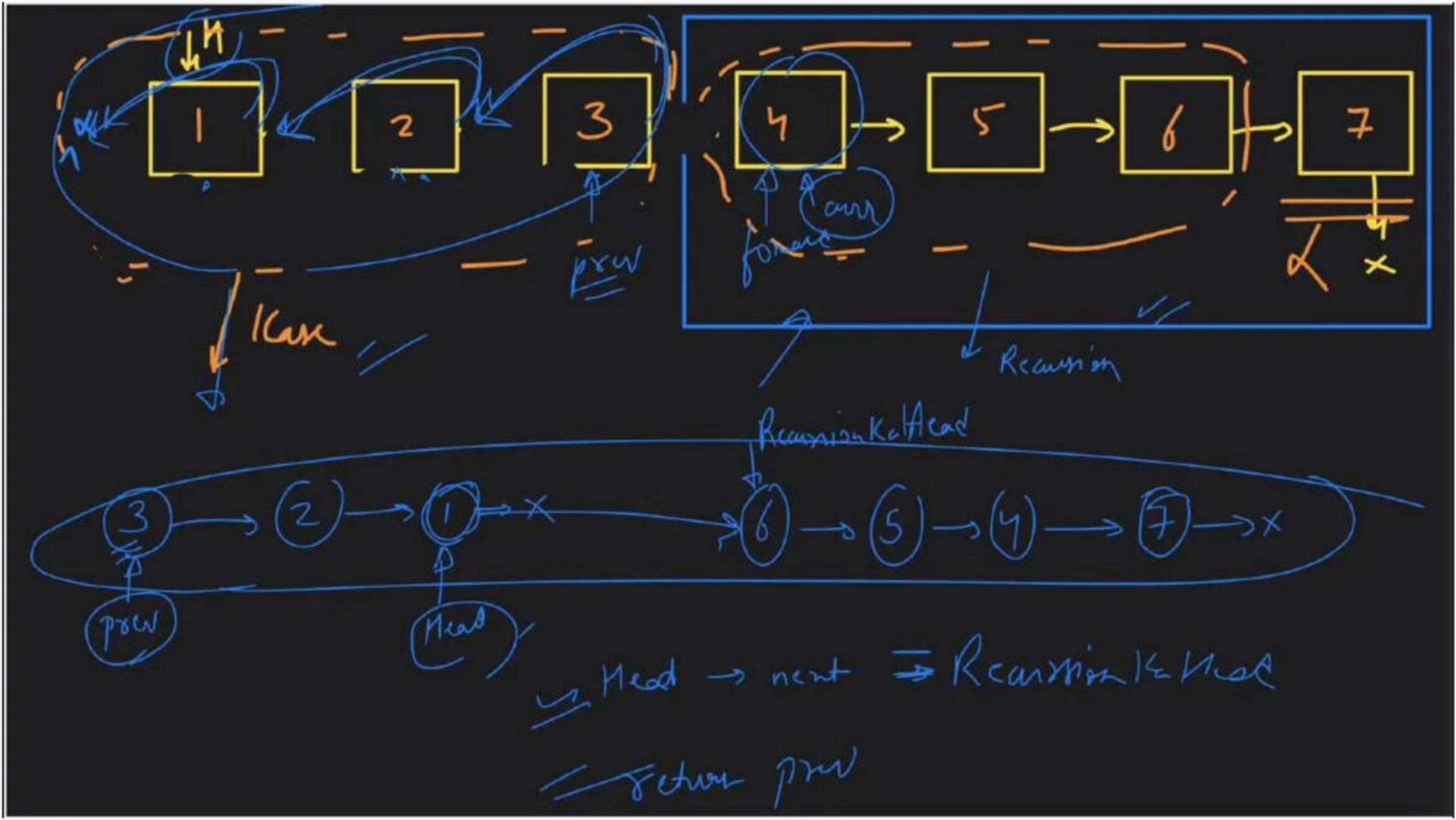

detect & delete Loop
check Loop
Starting point
delete loop
Mega
Class

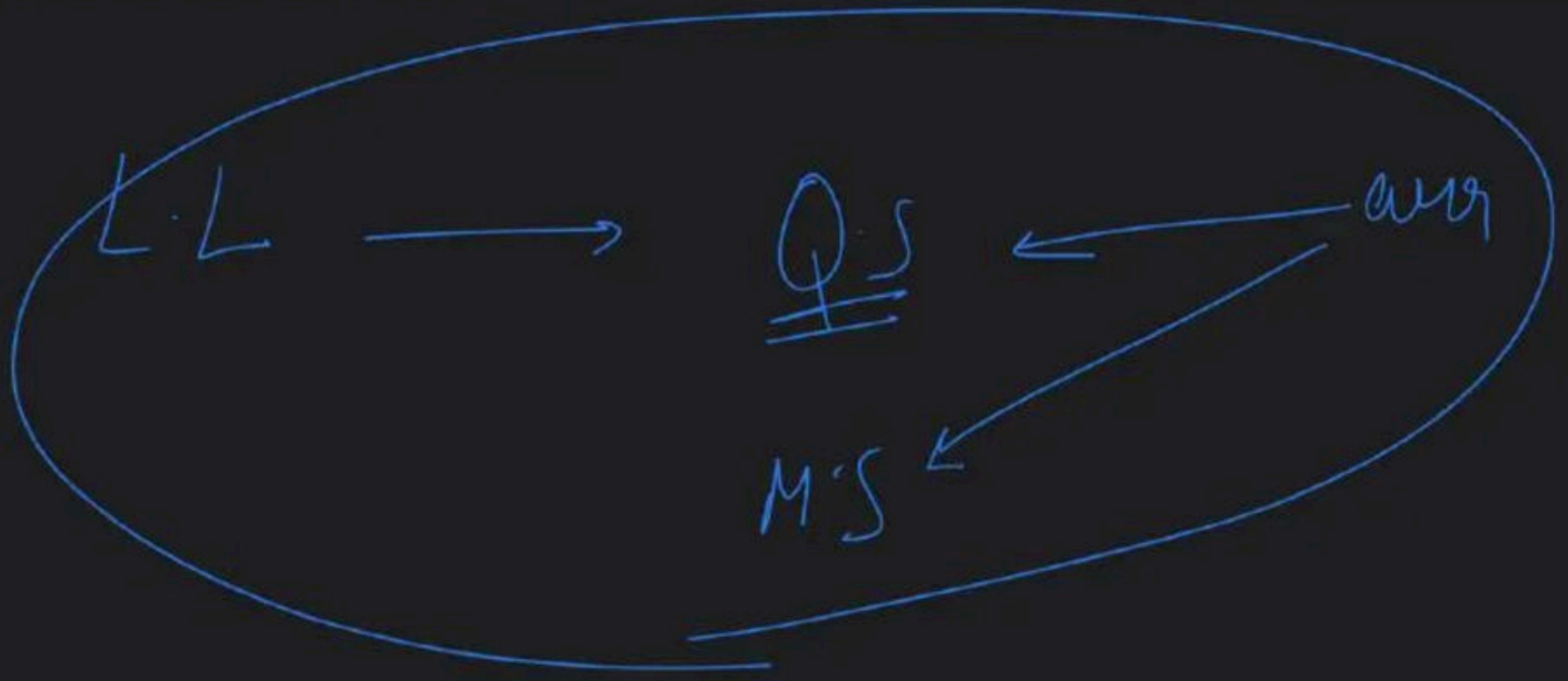
M.S → Online assessment

↳ reverse in groups of k


head
K = 2
1
2
3
4
5
6
7
2
1
4
3
6
5
7
find Ans


head
1
2
3
4
5
K=2
curr
forward
prev
RecursionHead
Recursion
head
2
1
4
3
5
X
prev
head → next = RecursionHead
next prev


1
2
3
4
5
6
7
curr
prev
x
Recursion
Recursion ka Head
3
2
1
x
6
5
4
7
x
prev
Head
Head → next ⇒ Recursion ka Head
return prev

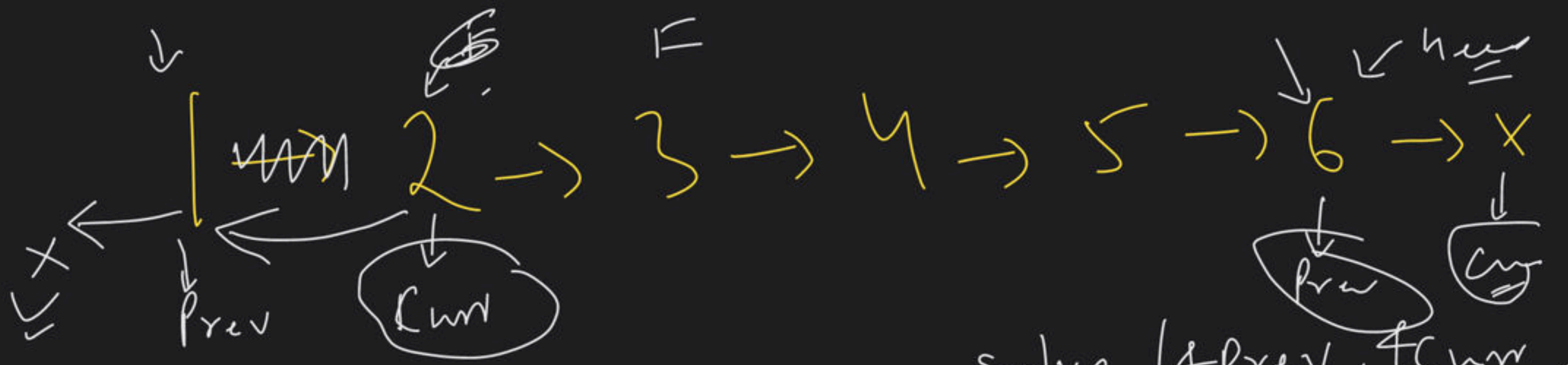

L·L
O·S
M·S

⇒ kal milenge

2 PM

↓

Mega class



① Node* forward = curr->next;

→ 2 Curr->next = prev;

prev = curr;

Curr = F;

Solve (prev, curr)

Solve (&prev, &Curr)

main →

Mega Clu