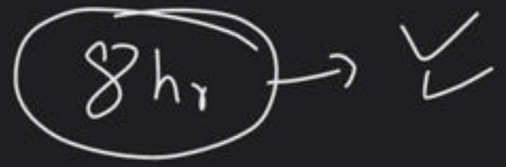


LinkedList Doubt Class with Lakshay Bhaiya

Special class

Love Babbar • Nov 17, 2023

Linked List - Class 1

Special class

2 min
Break dedo

Love Babbar • Nov 3, 2023

Linked List

non-continous memory allocation

wastage

Linear D.S

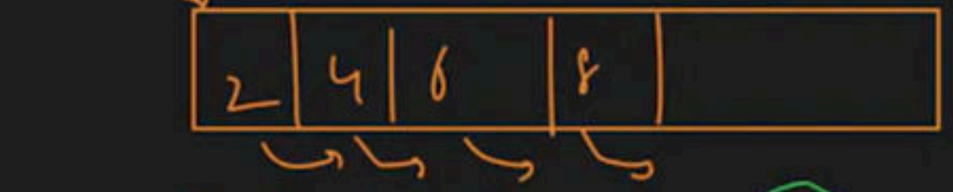
Array / Vector

waste

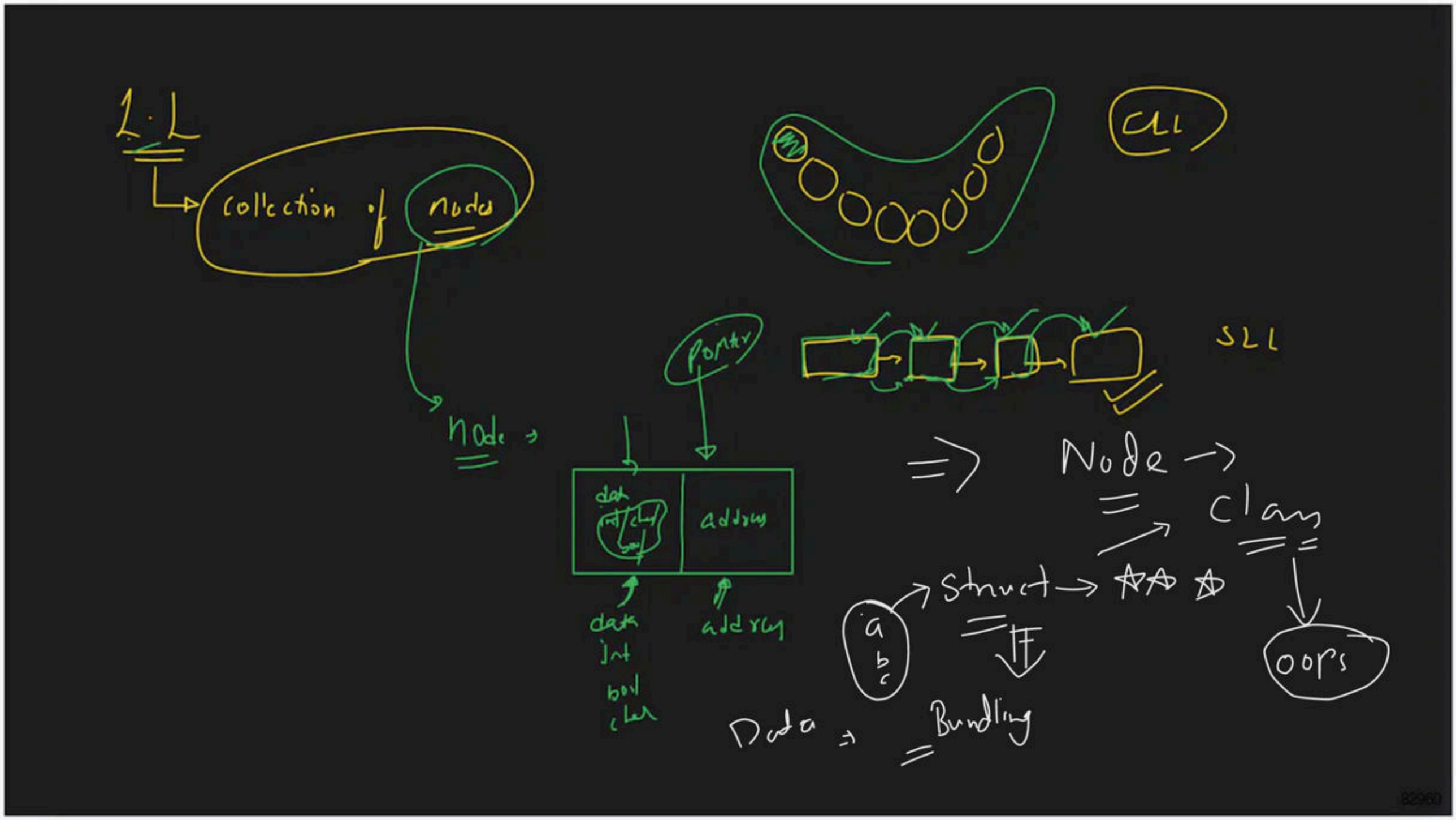
continous memory

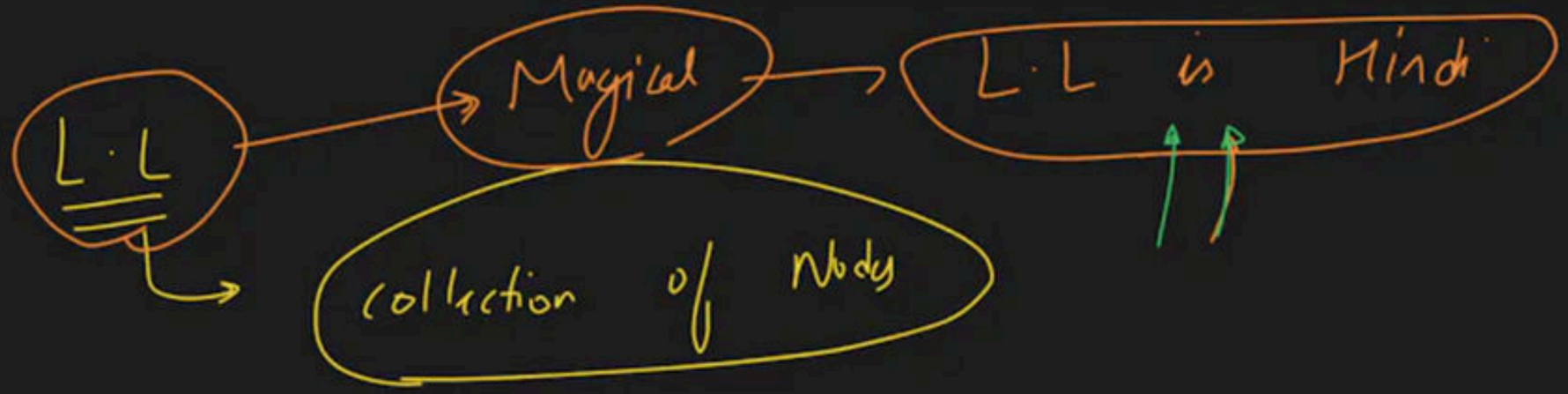
insert/shift → T.C → O(n)

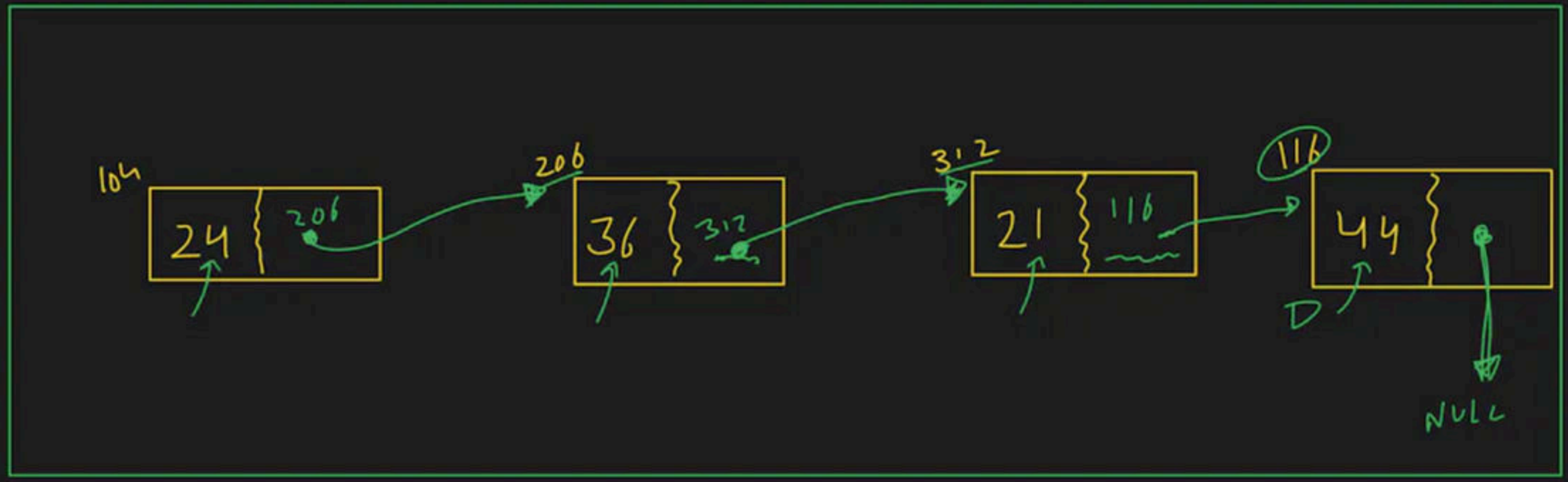
ID

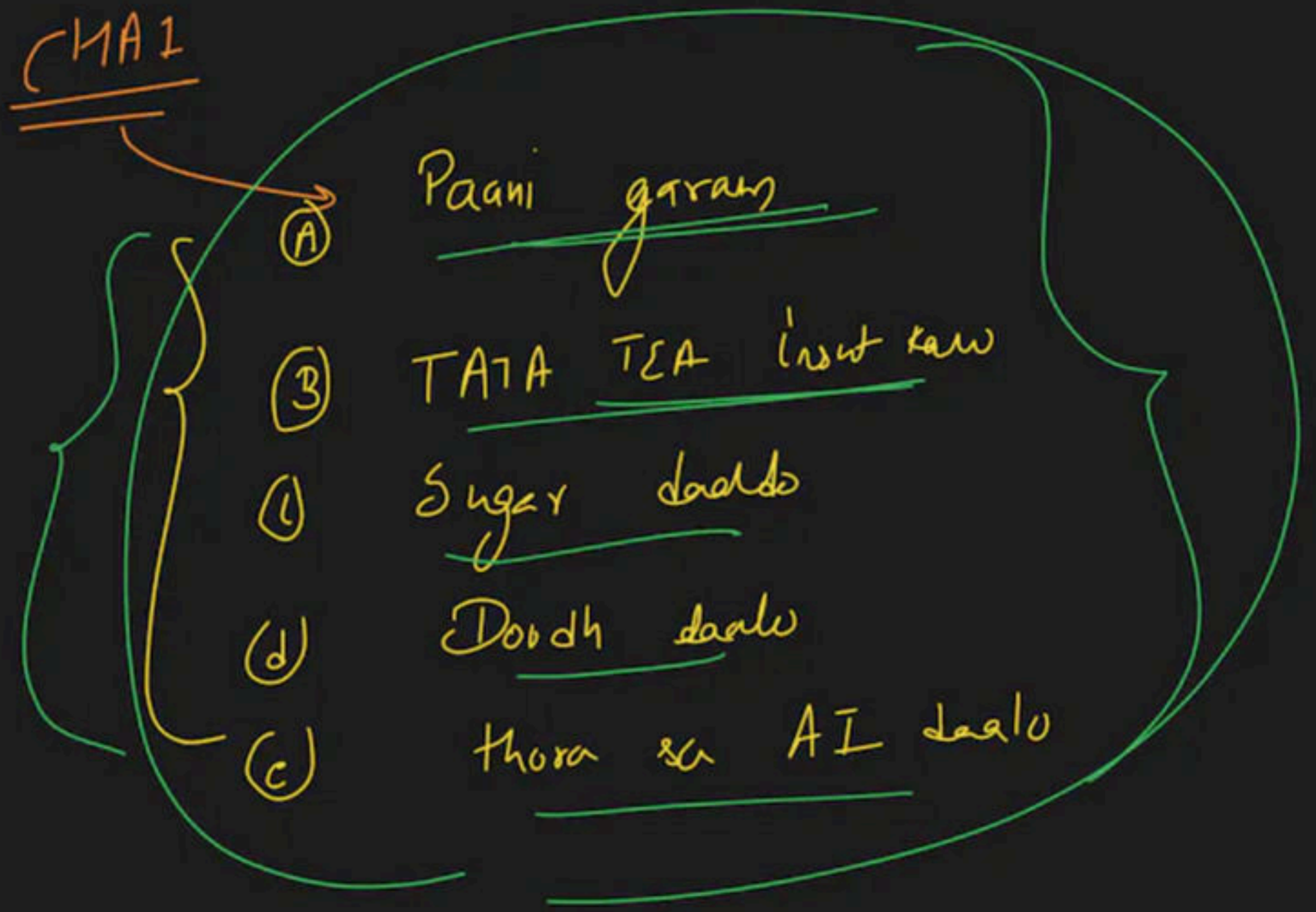


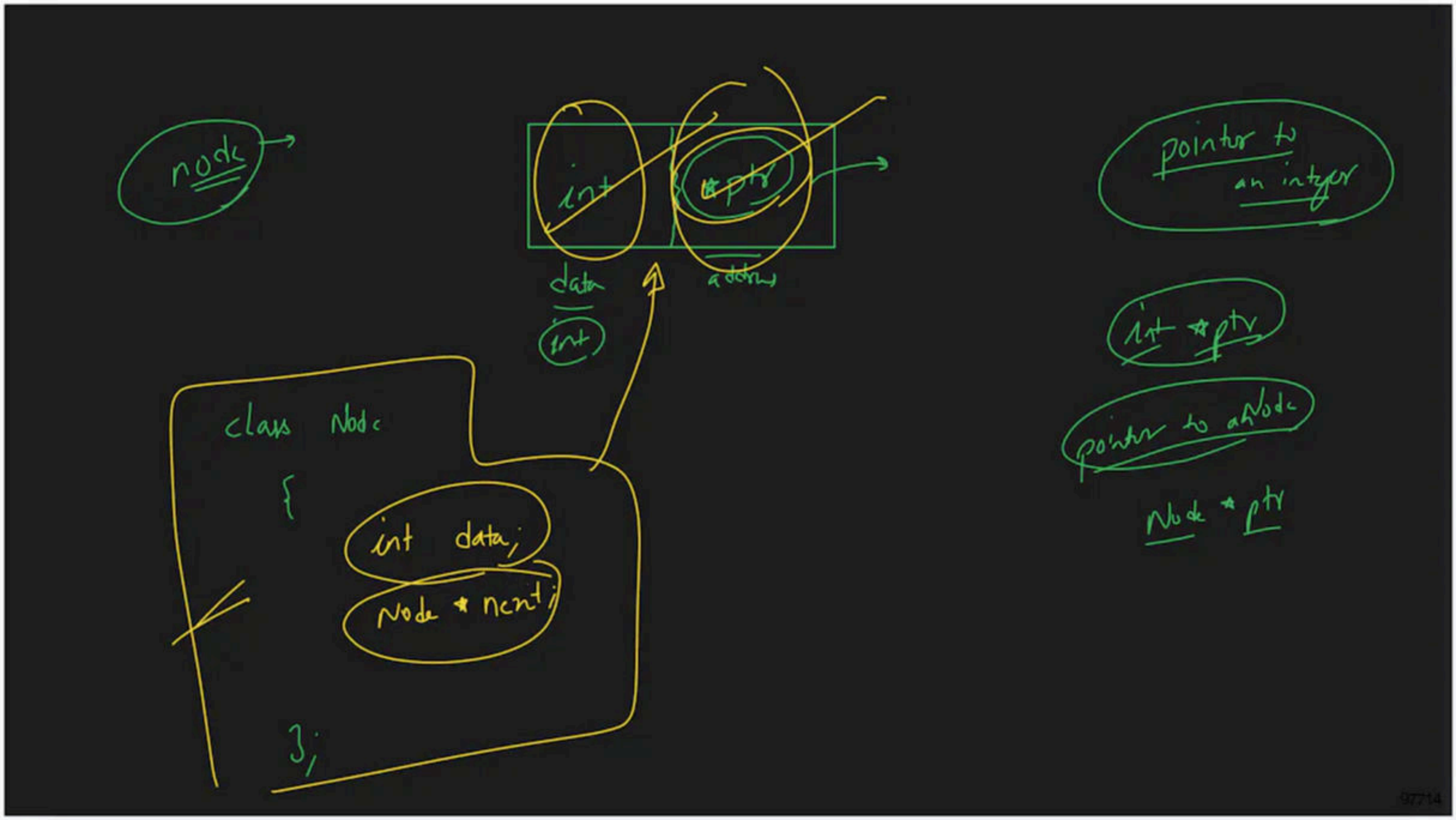
42204

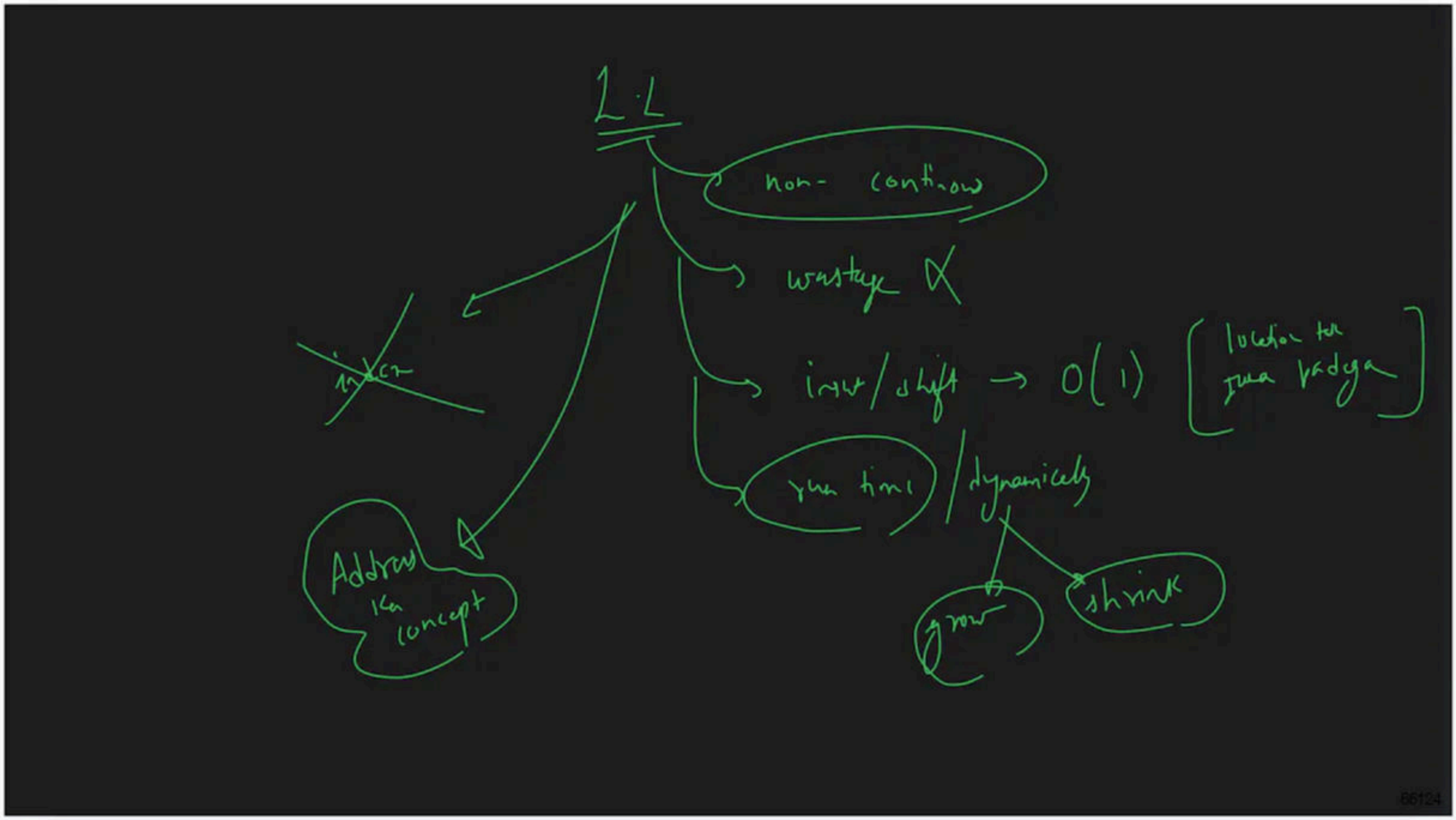

L.L
collection of Node
node =
CLL
Pointer
SLL
data
address
data
int
bool
char
address
=>
Node ->
class
Struct ->
a
b
c
oops
Data =
Bundling


L·L
Magical
L·L is Hindi
collection of Nody


104
24
206
206
36
312
312
21
116
116
44
D
NULL


CHAI
A Paani garam
B TATA TEA instant
C thora sa AI daalo
D Doodh daalo
C Sugar daalo


node
int
*ptr
data
int
address
Pointer to an integer
int *ptr
pointer to a Node
Node *ptr
class Node
{
int data;
Node * next;
};

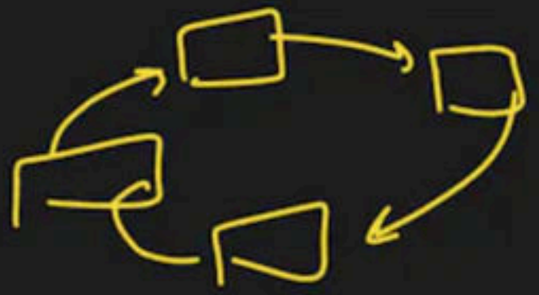

L.L
non- continuous
wastage ✗
index
insert/shift → O(1)
location ka
Jaisa padega
run time
/dynamically
Address
ka
concept
grow
shrink

Types

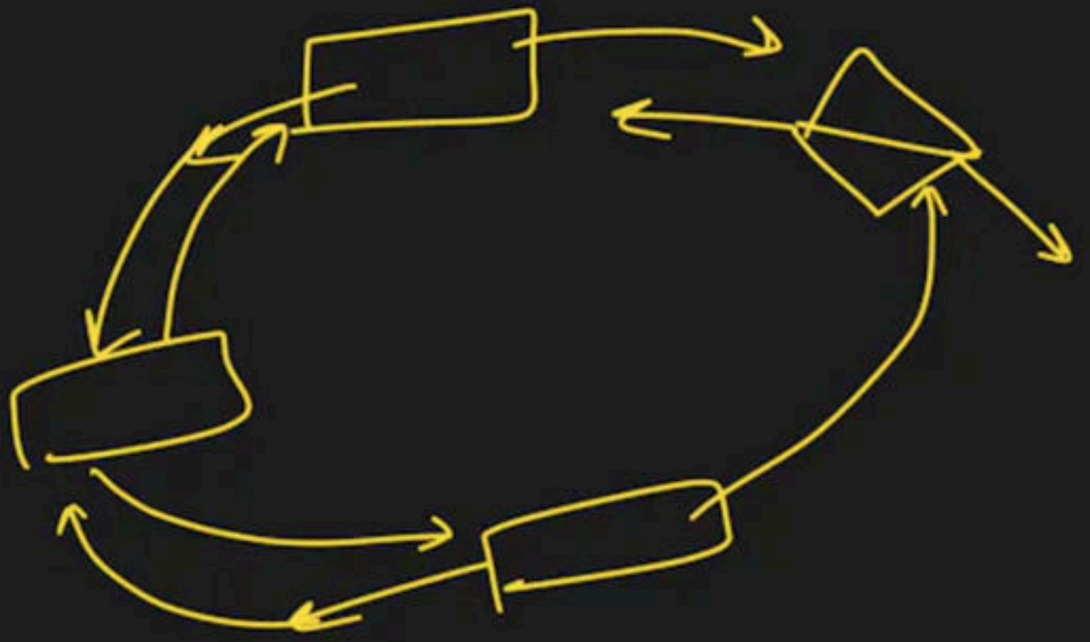
- Singly L.L

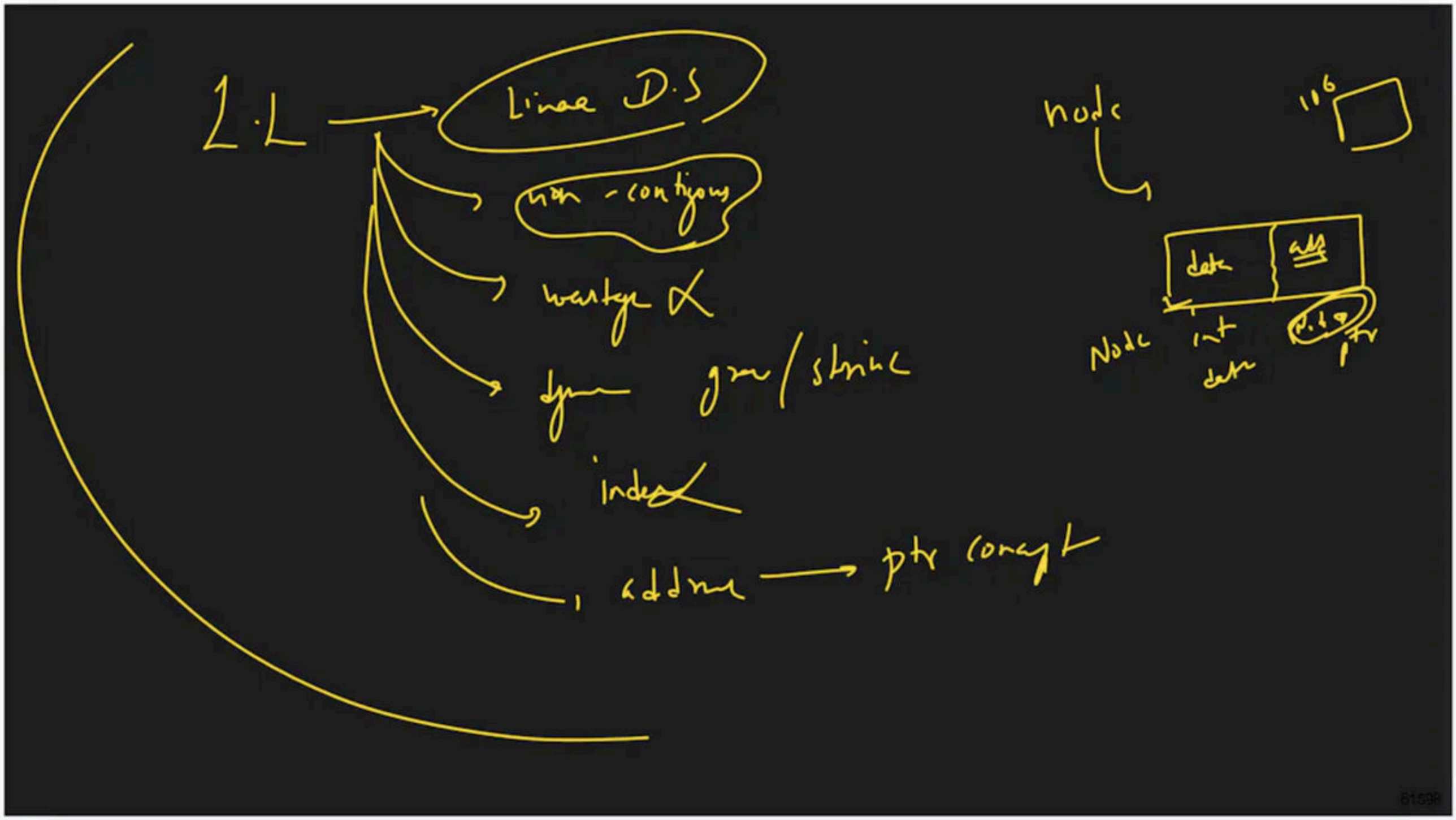
- Doubly L.L

- Circular Singly L.L

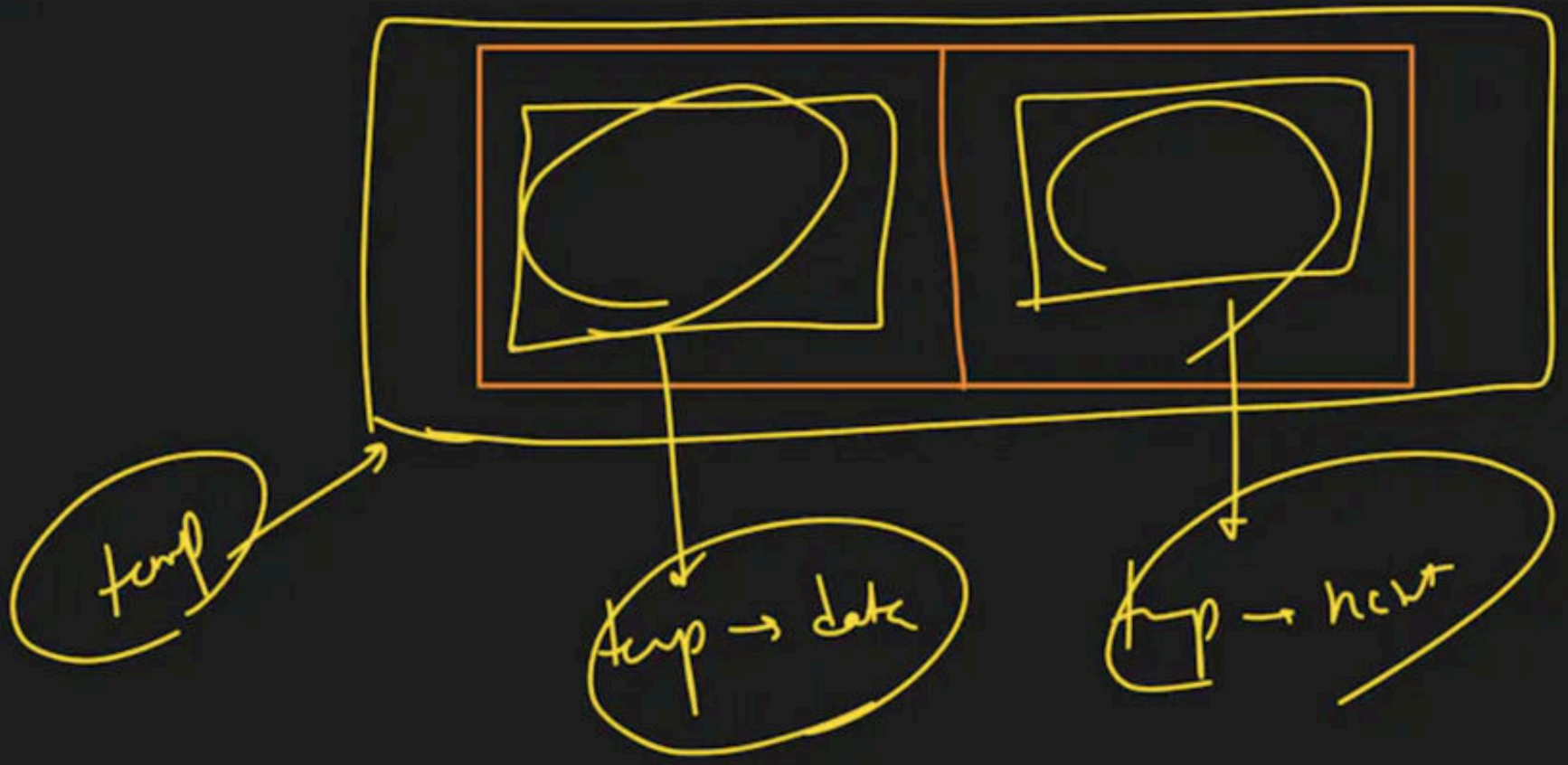


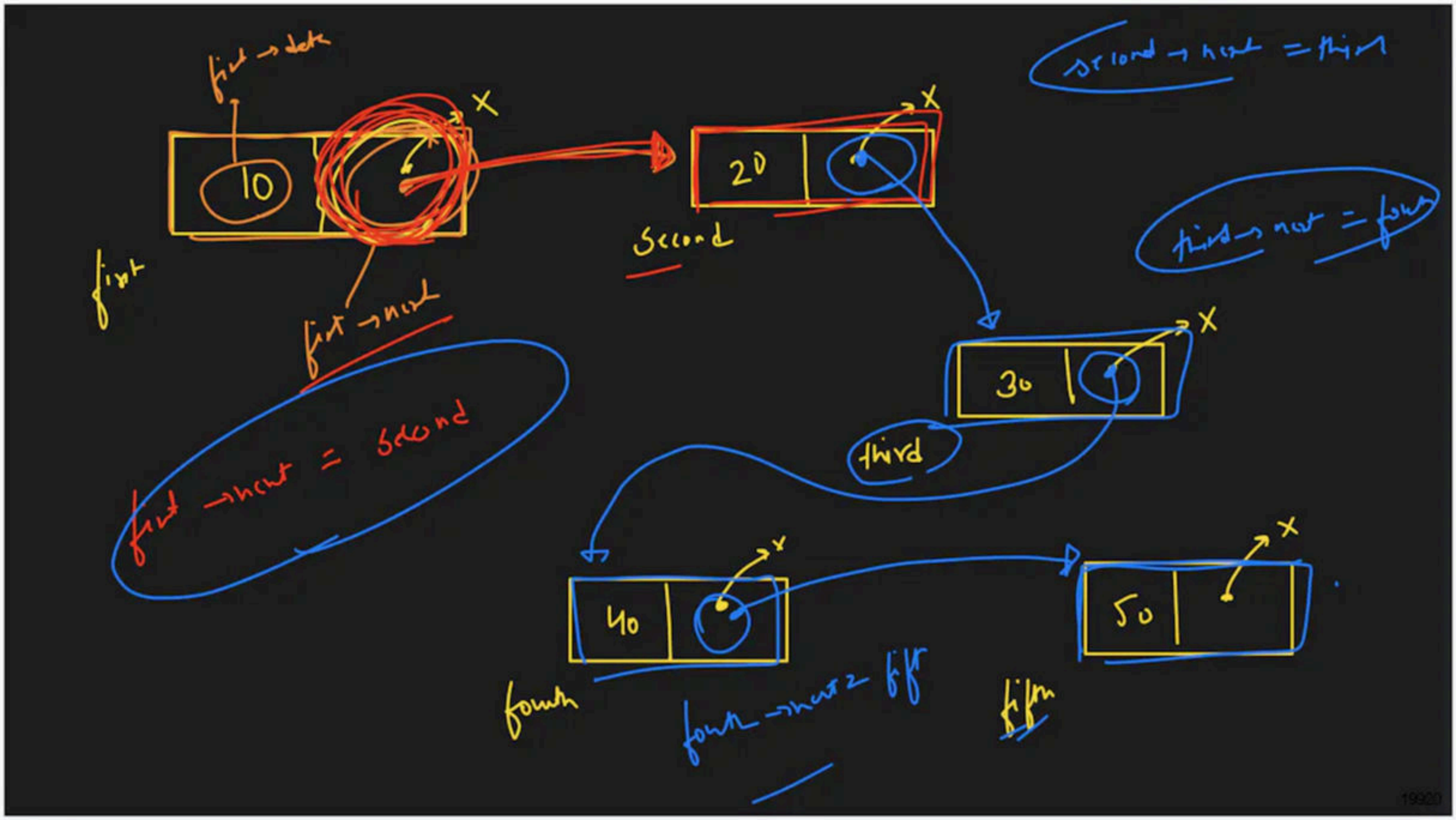
Circular Doubly L.L

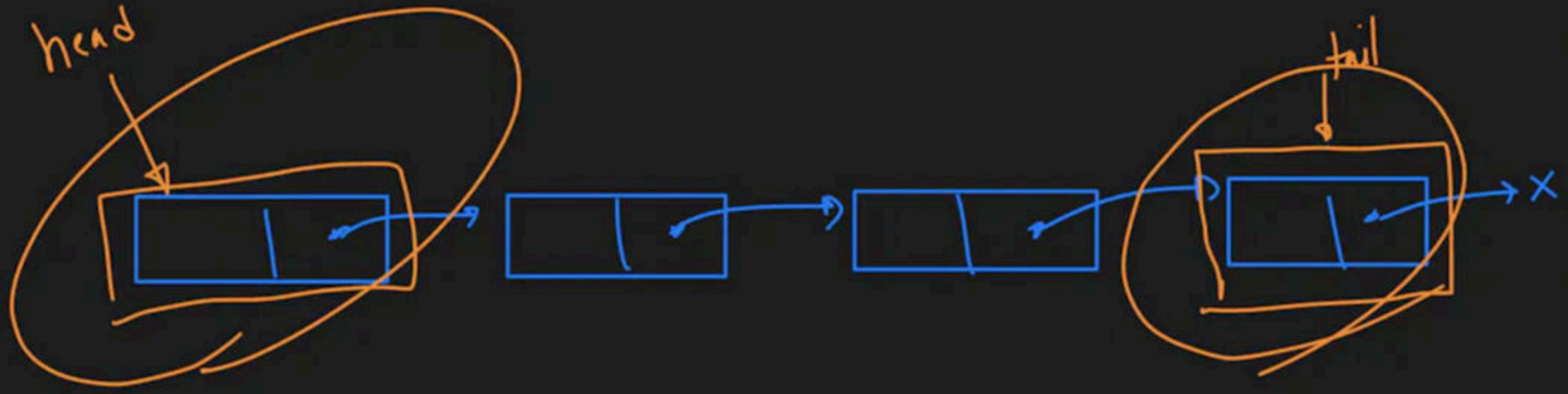


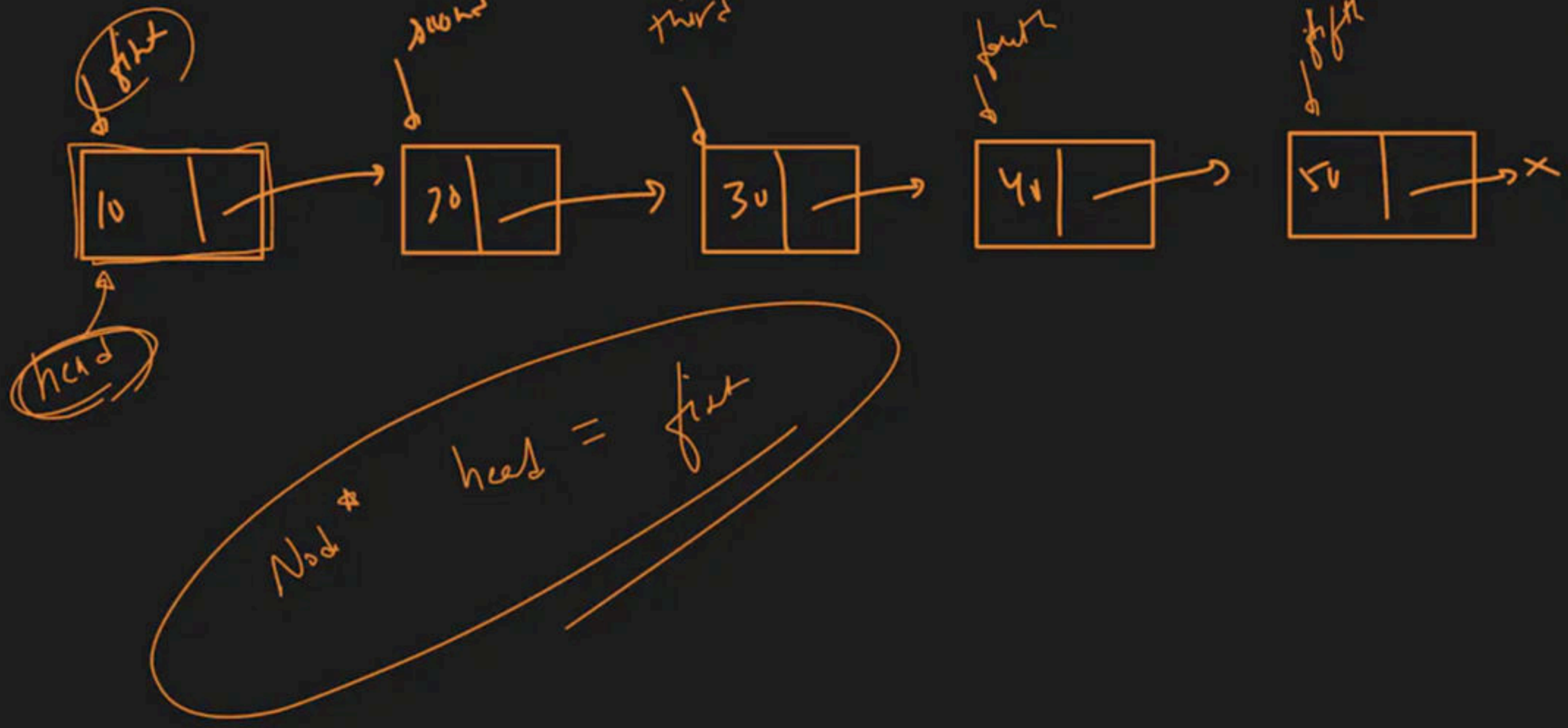

L.L
Linear D.S
non-contiguous
node
data
Node
int
data

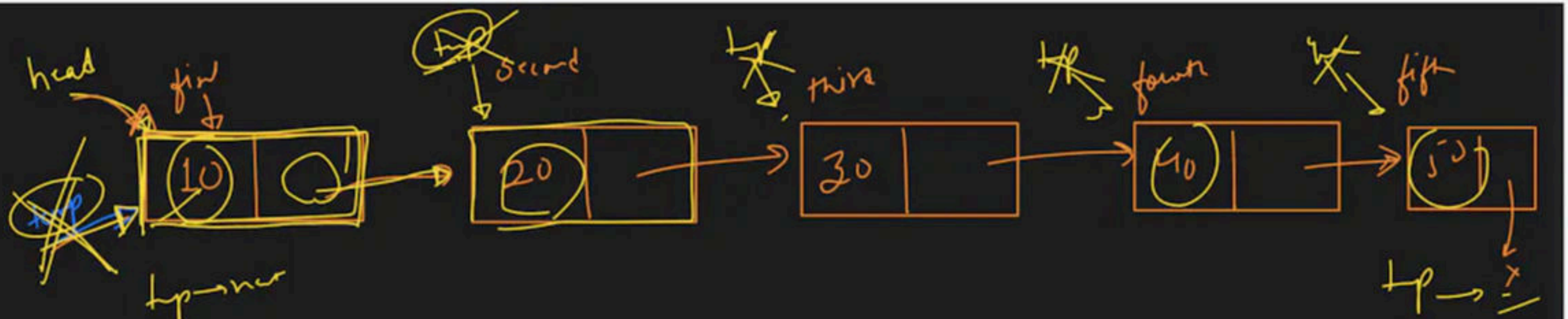
Node *temp = new Node()




first -> data
10
X
first
first -> next
second -> next = third
20
X
second
third -> next = fourth
first -> next = second
30
X
third
40
X
fourth
fourth -> next = fifth
50
X
fifth

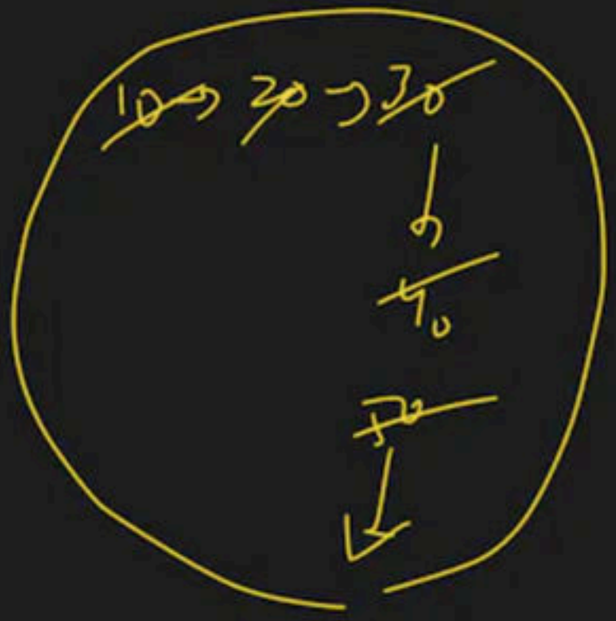

head
tail
X

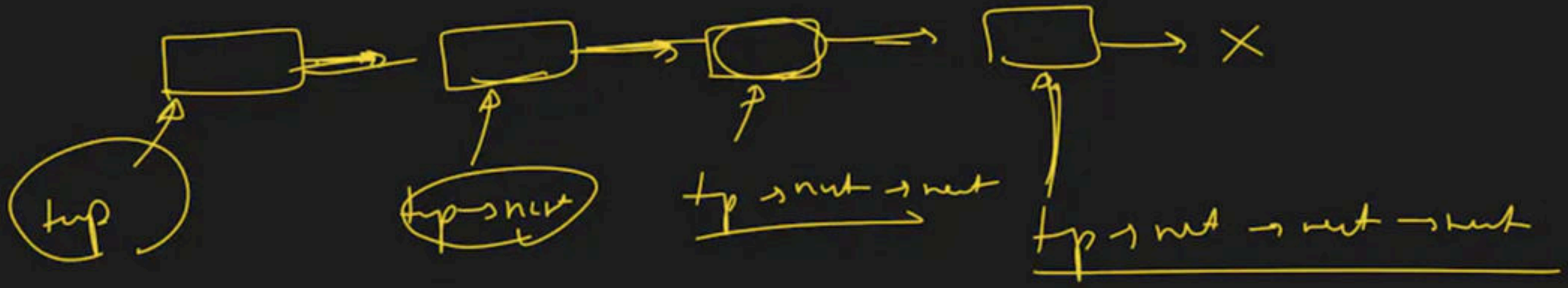

first
second
third
fourth
fifth
10
20
30
40
50
head
Node * head = first

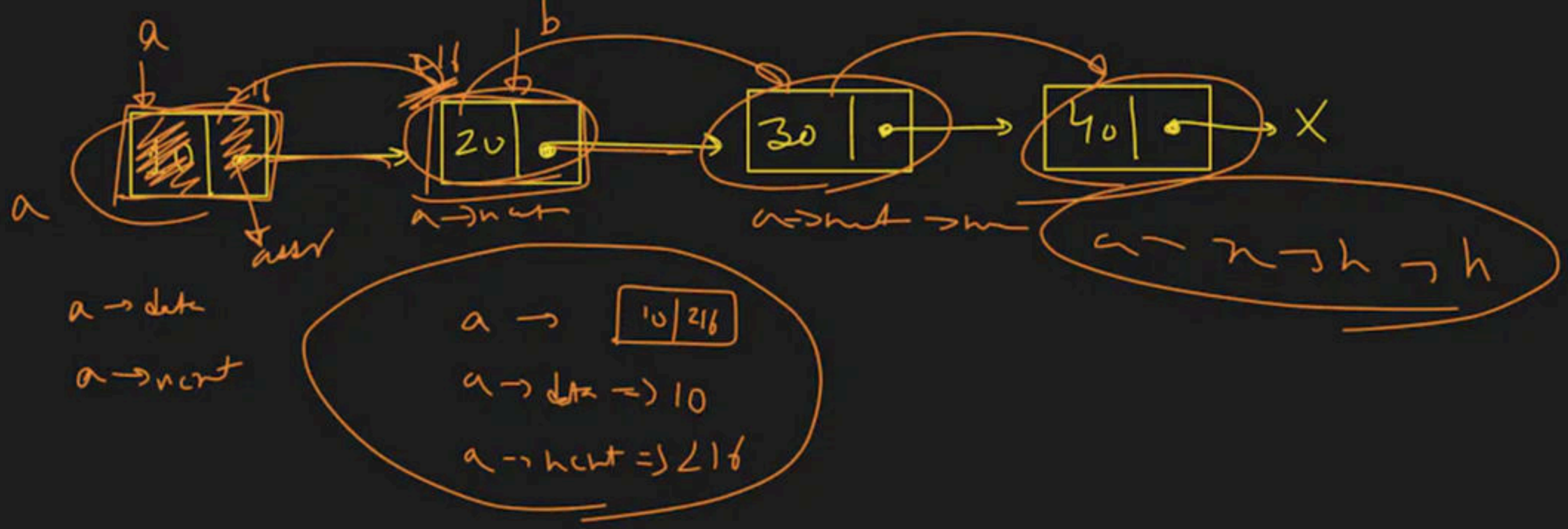


print (head)

```
{
    Node *tmp = head;
    while (tmp != NULL)
    {
        cout << tmp->data;
        tmp = tmp->next;
    }
}
```



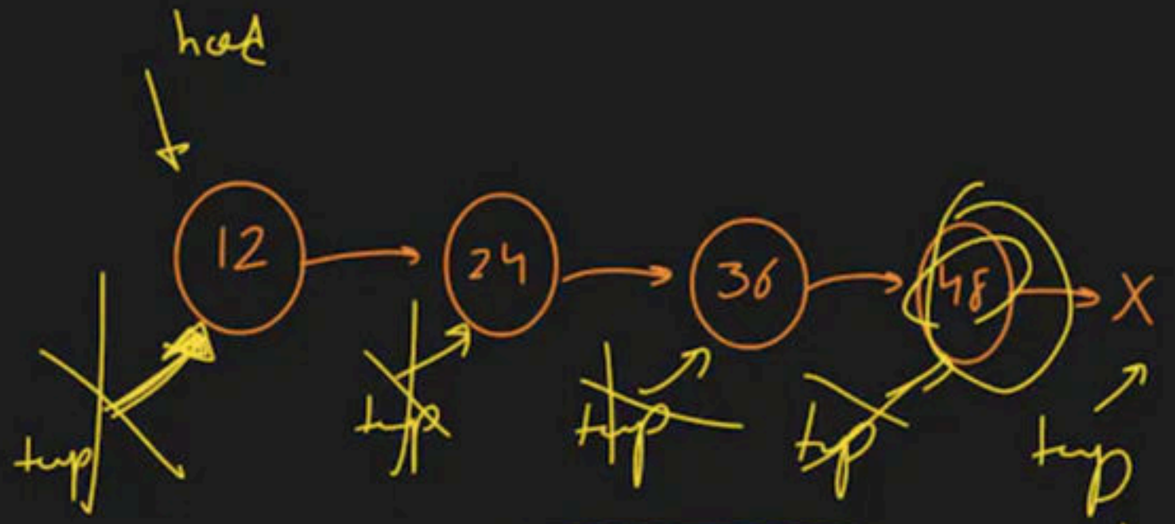

X
tmp
tmp->next
tmp->next->next
tmp->next->next->next


a
b
10
20
30
40
X
a
a->next
a->next->next
a->next->next->next
a->data
a->next
a -> 10 216
a->data => 10
a->next => 216

b = a->next

b = 216

```
print ( head )
{
    Node * temp = head;
    while ( temp != NULL )
    {
        cout << temp -> data;
        temp = temp -> next;
    }
}
```

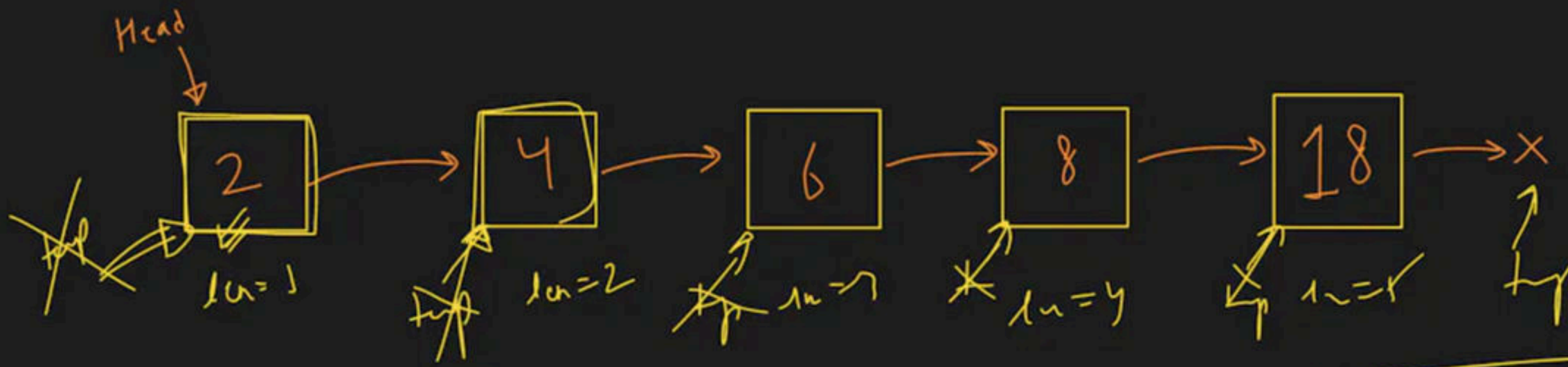



```
while ( temp -> next != NULL )
```

Point Len = No. of Words

in a L.L

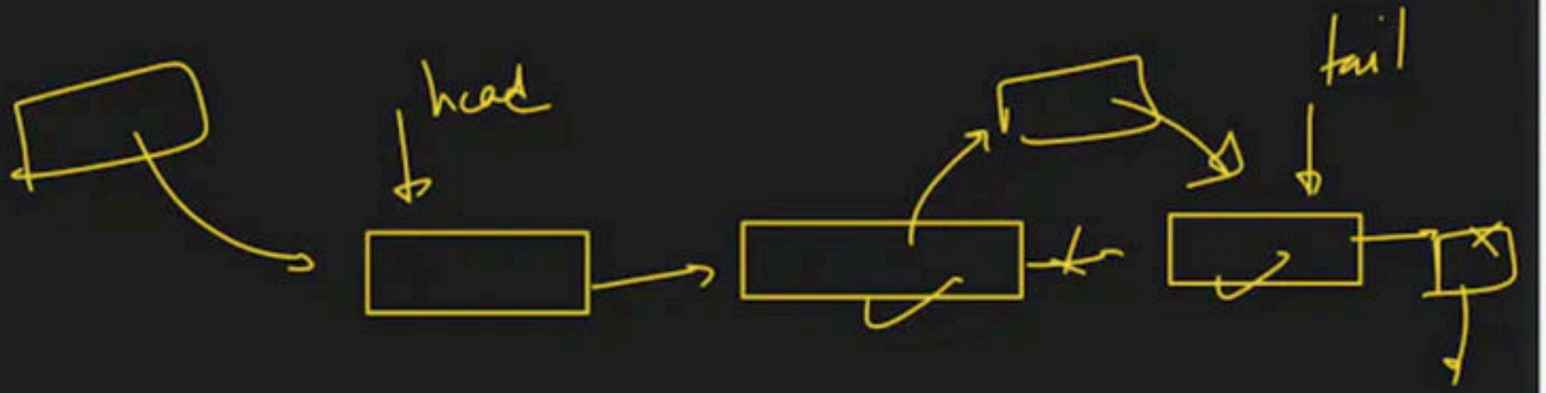
2 min



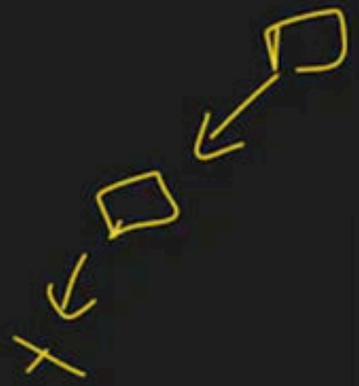
Len = 0

```
while (typ != NULL)
{
    length++;
    typ = typ -> next;
}
```

Insertion:-

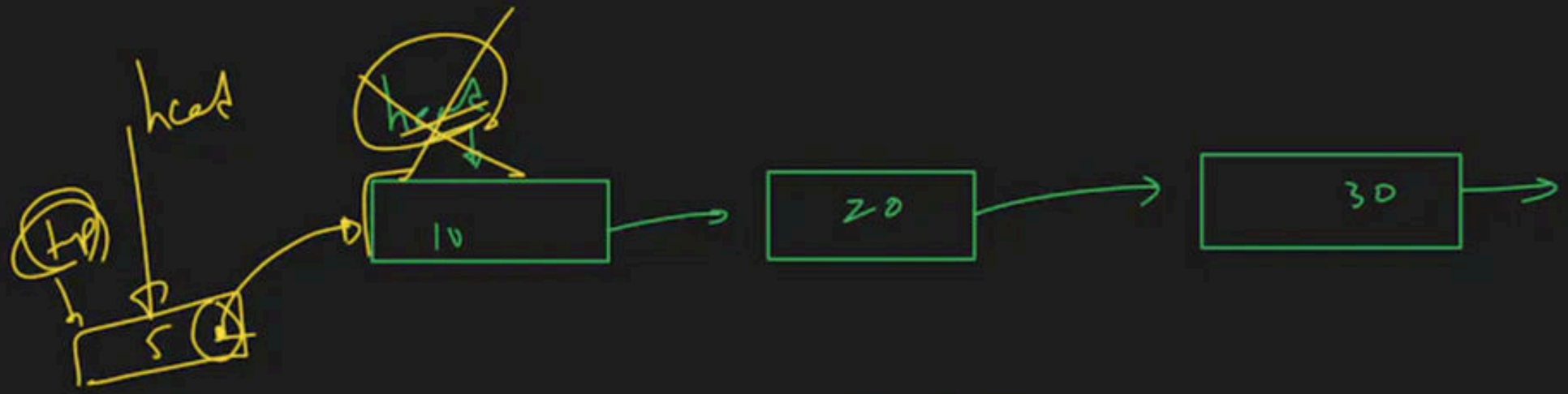


- head – beginning
- tail
- insert at any point



Insertion

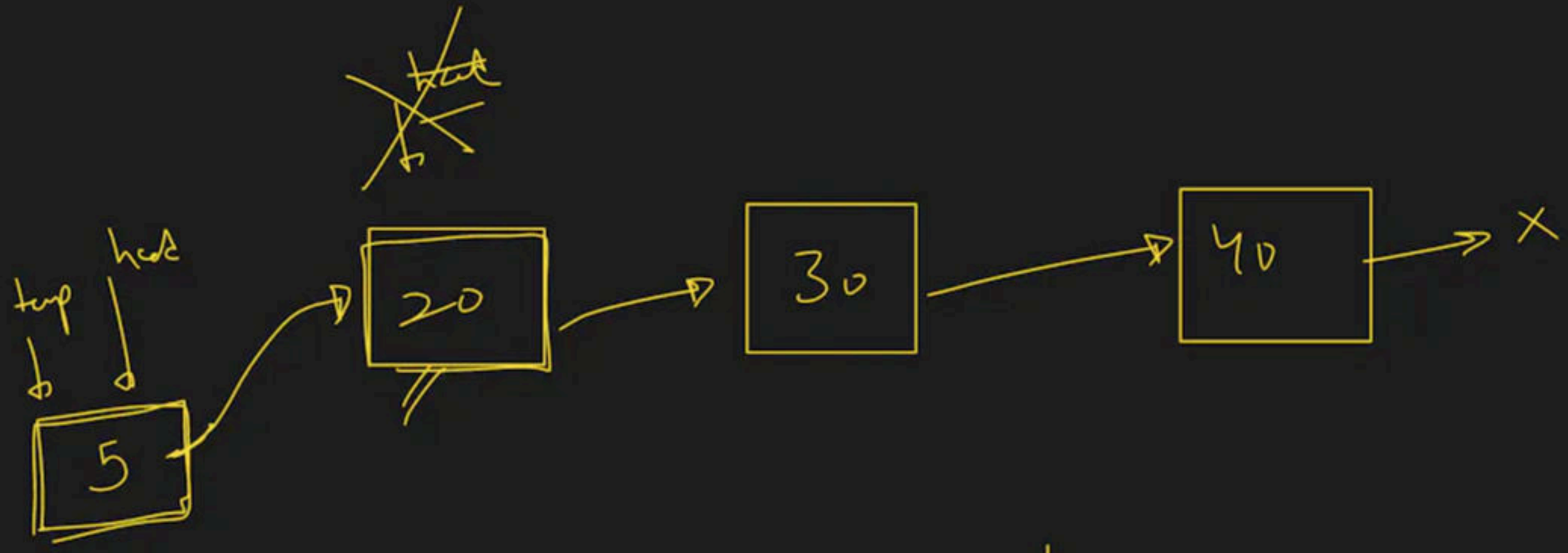
→ insertAtHead



(A) Create new Node

(B) temp → next = head

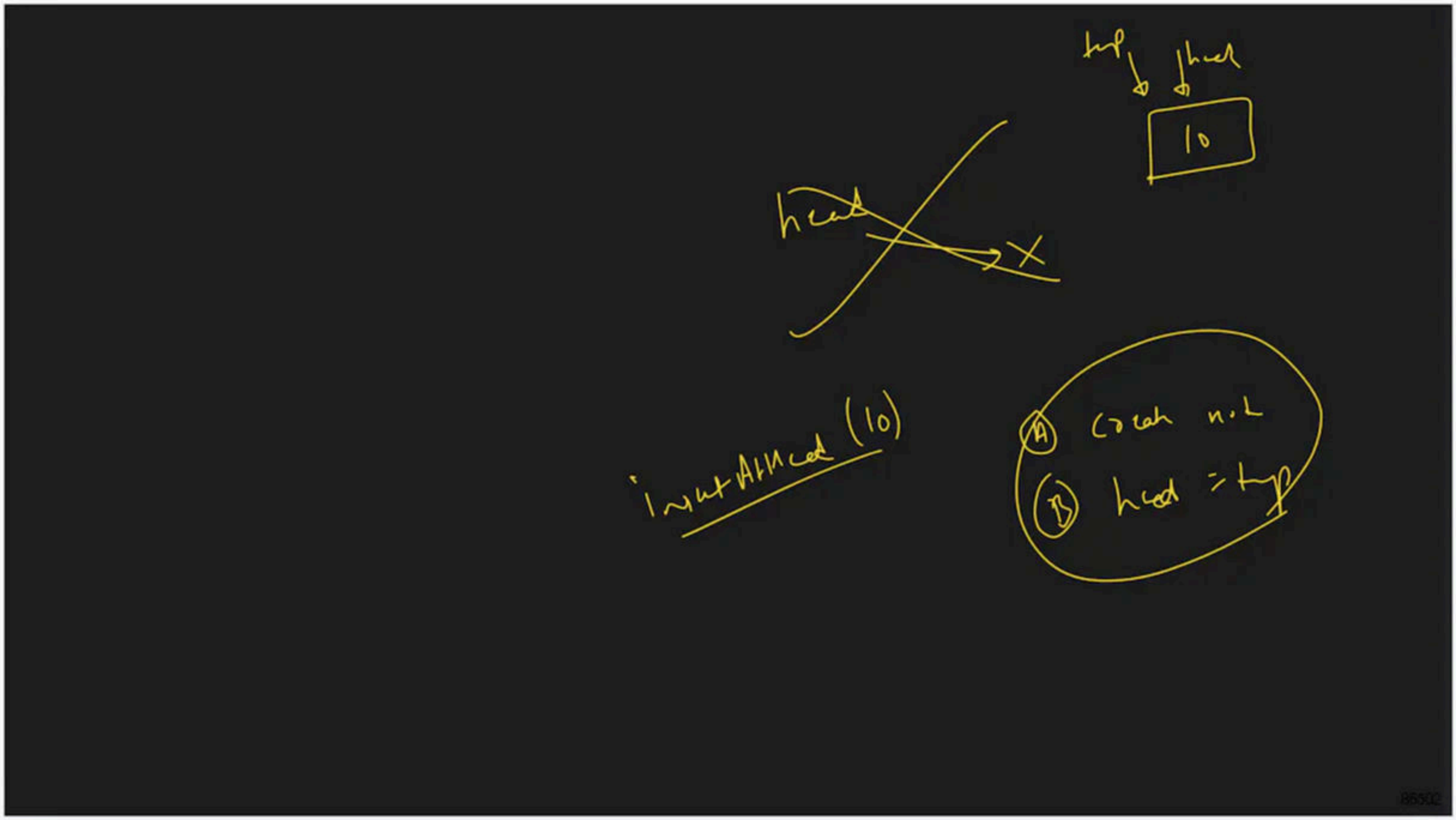
(C) head = temp



① Create new Node

② temp → next = head

③ head = temp


temp
head
10
head
X
insertAtHead(10)
A
Create n.L
B
head = temp

Insert At tail

insertATail (560)

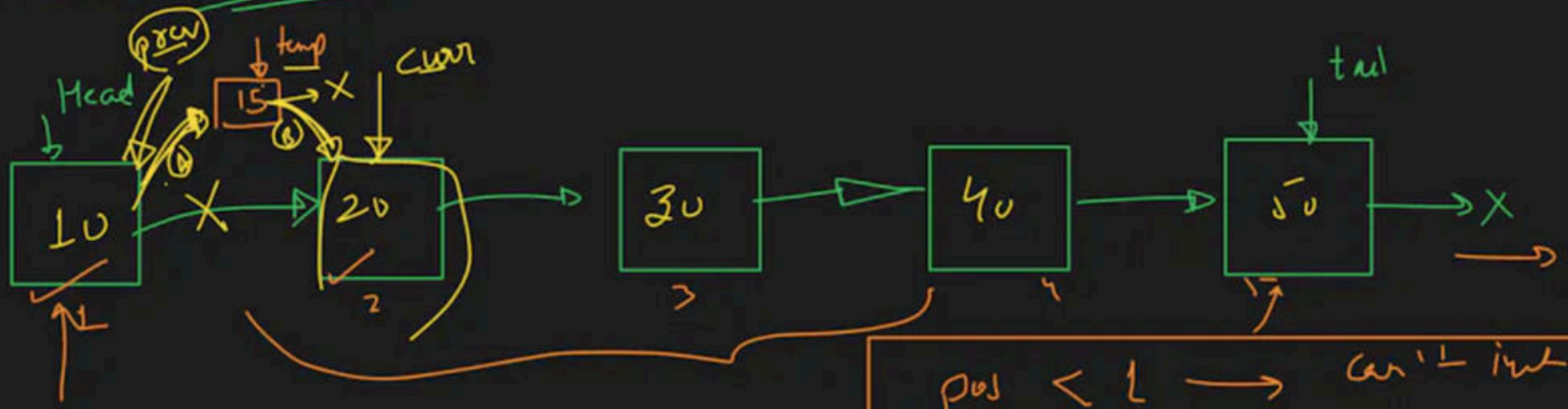
(A) Create new Node

(B) tail->next = temp

(C) tail = temp

Insert At Position

i/p → "n" position



position = 2

(A) create a node

(B) traverse curr/prev to position

(C) prev → next = temp

temp → next = curr

pos < 1 → can't insert

pos = 1 → insertAtHead

pos = 5 → insertAtTail

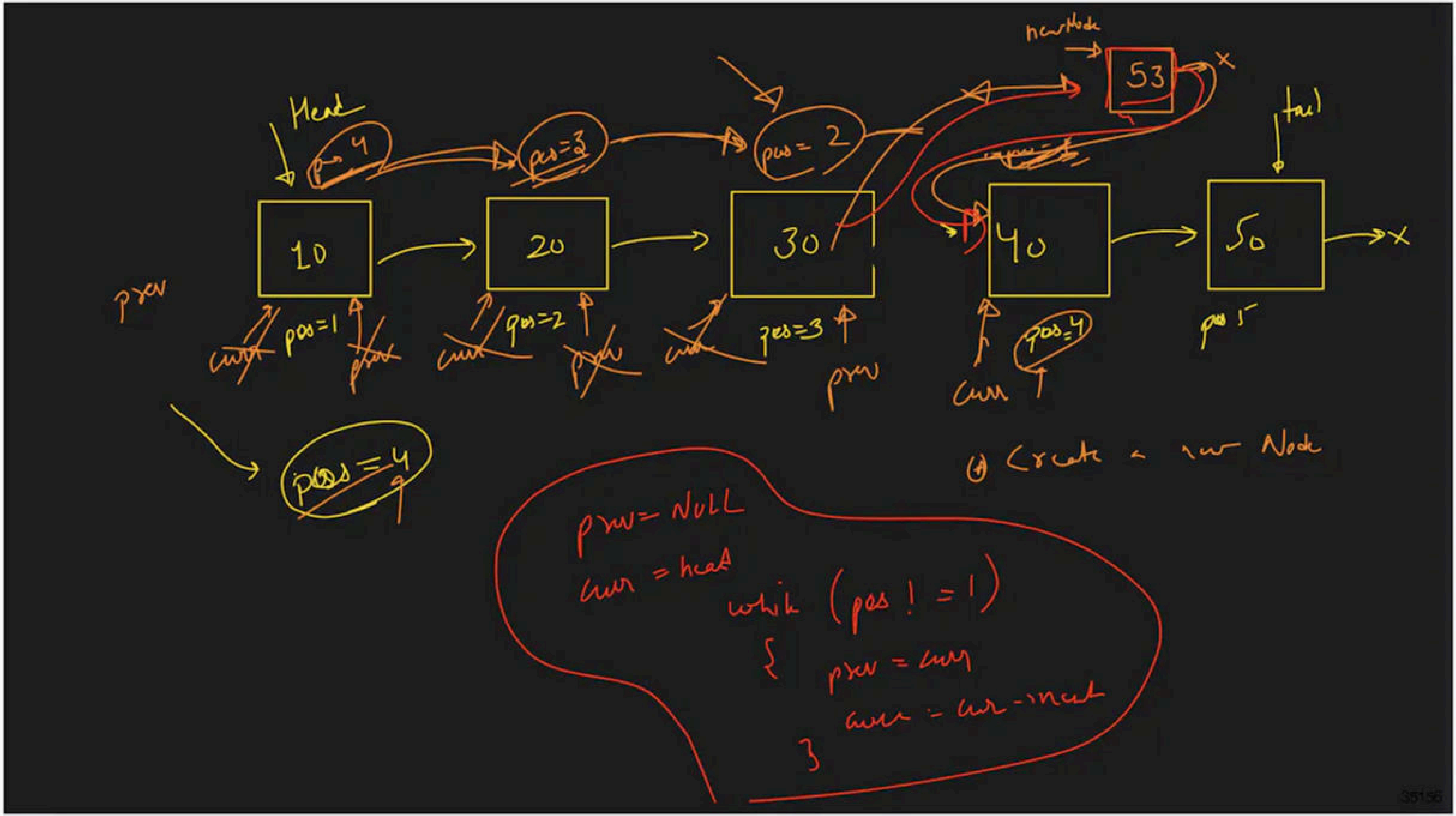
pos > 1 & pos < 5 → Logic fine

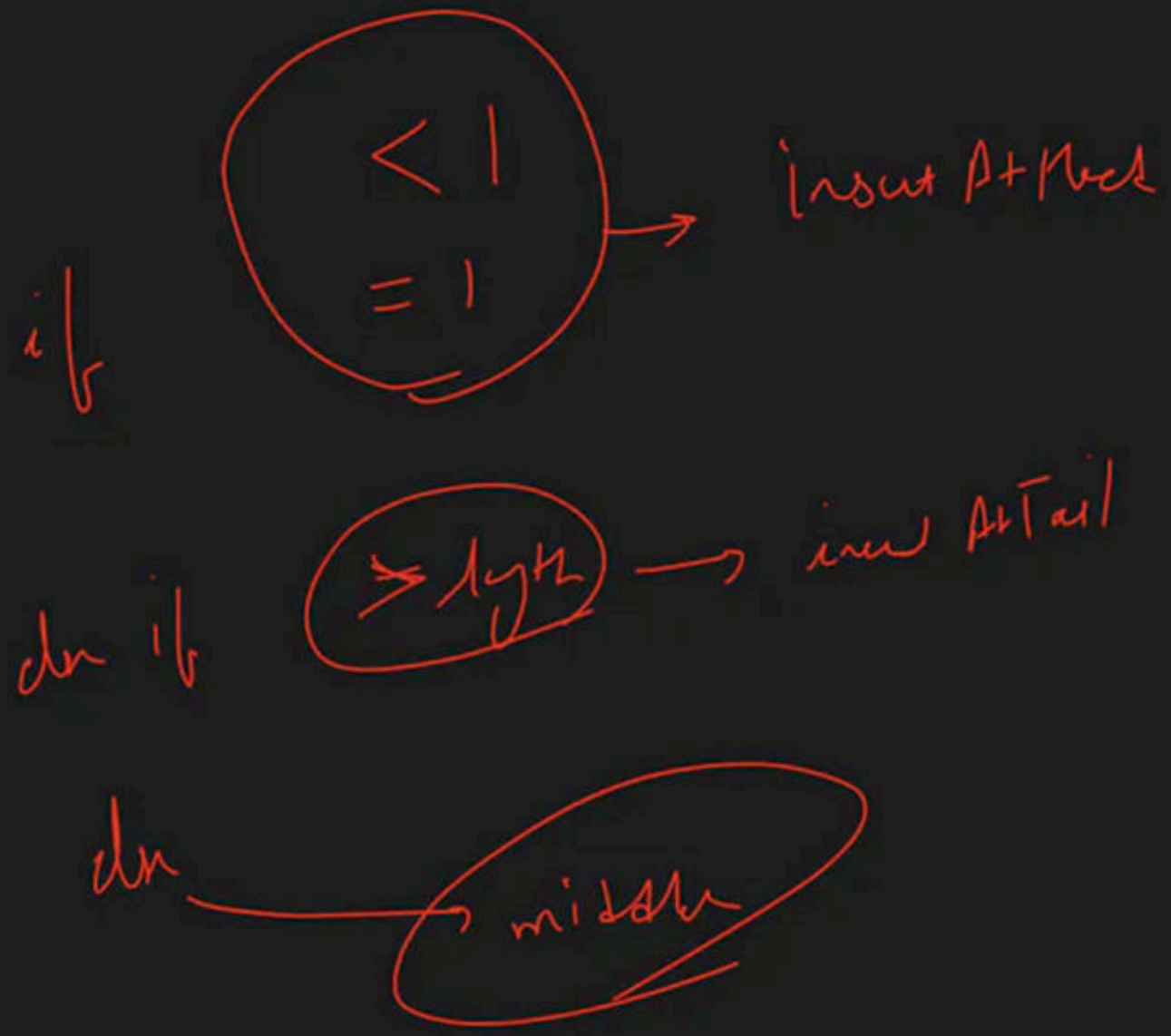
pos > 5 → can't insert

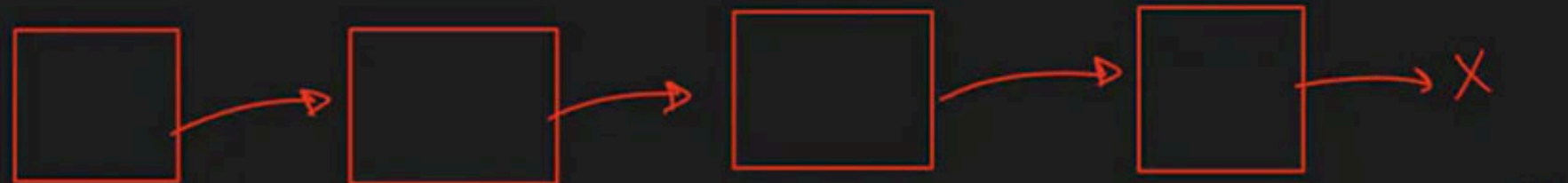
pos < 1 → cannot insert

post < 1
pos = 1 → insert At Head

pos = lyn + 1
pos > lyth → insert At Tail

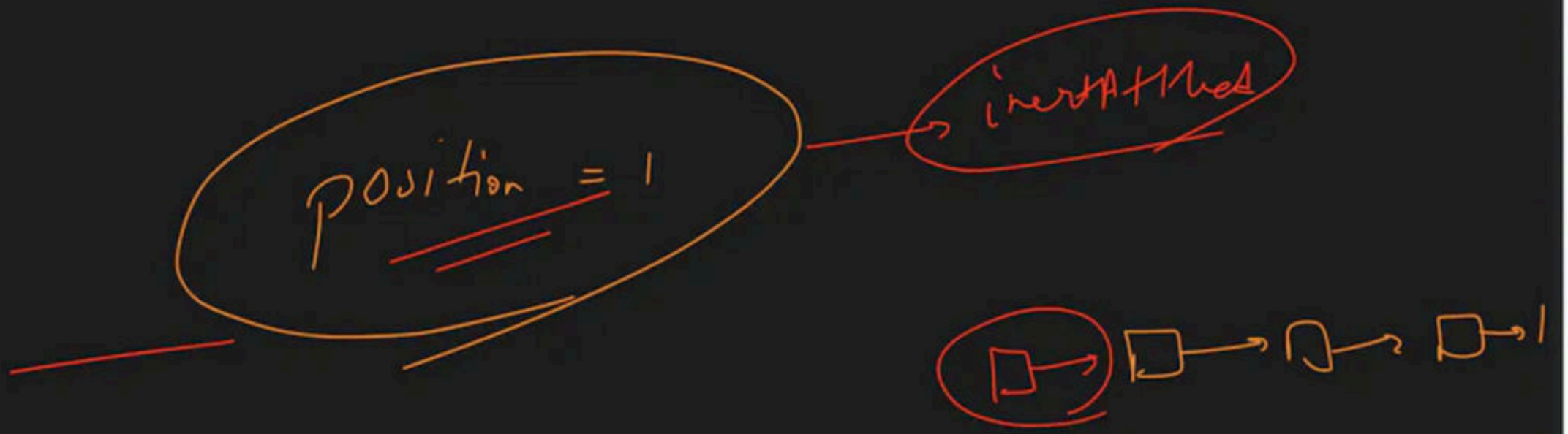

Head
10
20
30
40
50
newNode
53
tail
pos=1
pos=2
pos=3
pos=4
pos=2
pos=3
pos=4
pos 5
prev
curr
pos=4
(1) Create a new Node
prev = NULL
curr = head
while (pos != 1)
{
prev = curr
curr = curr->next
}


< 1
= 1
insert AtHead
>= 1
insert AtTail
middle

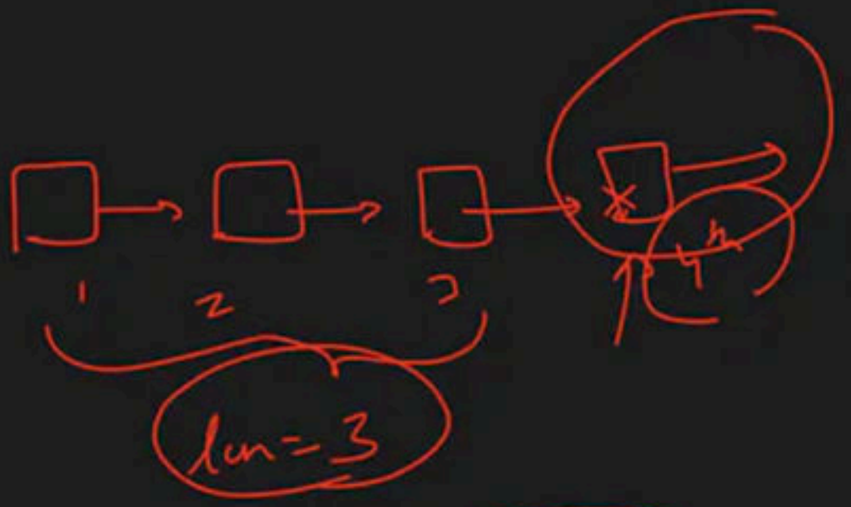


len = 4

pos → [1 ⟷ len + 1]

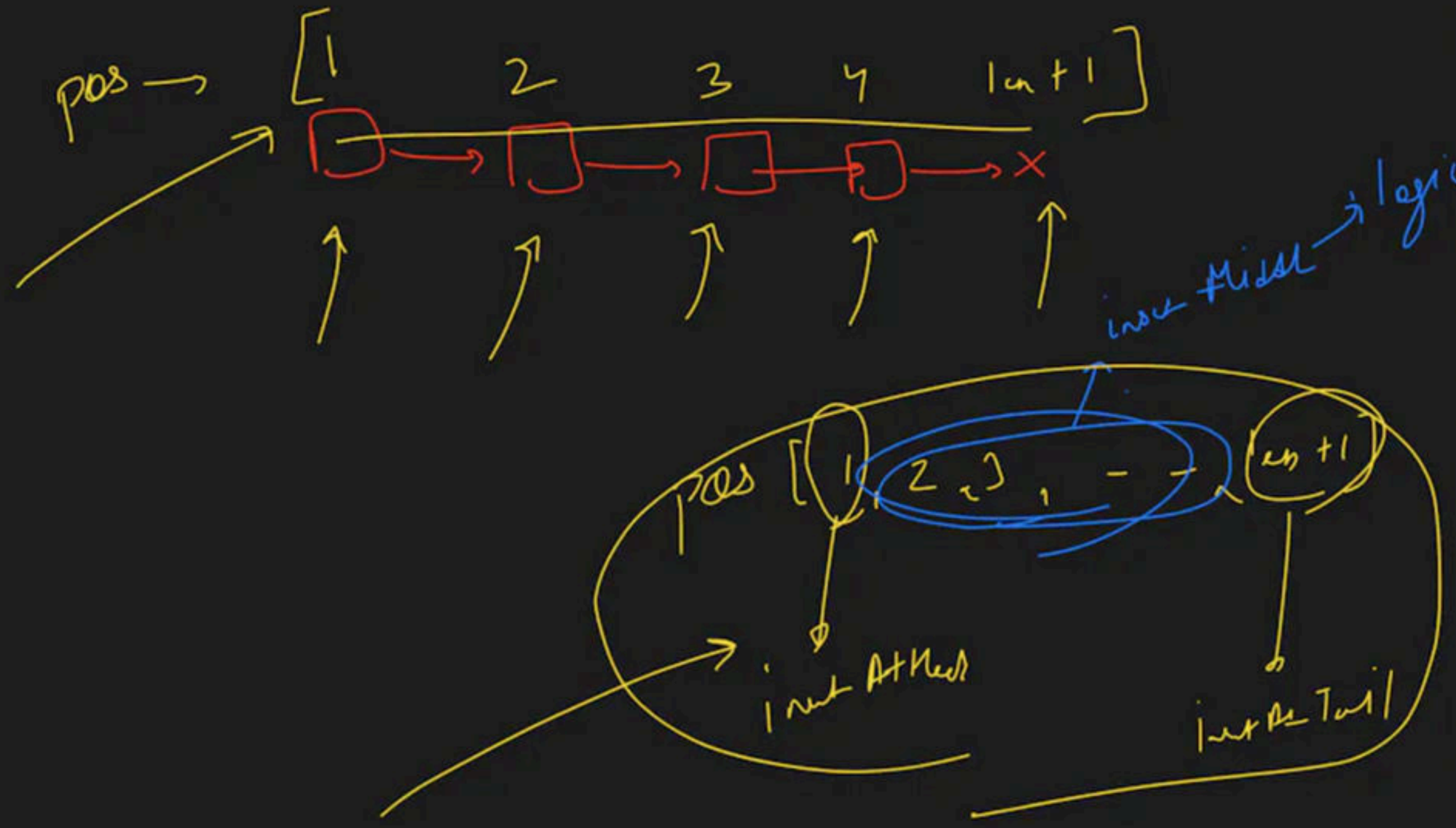

position = 1

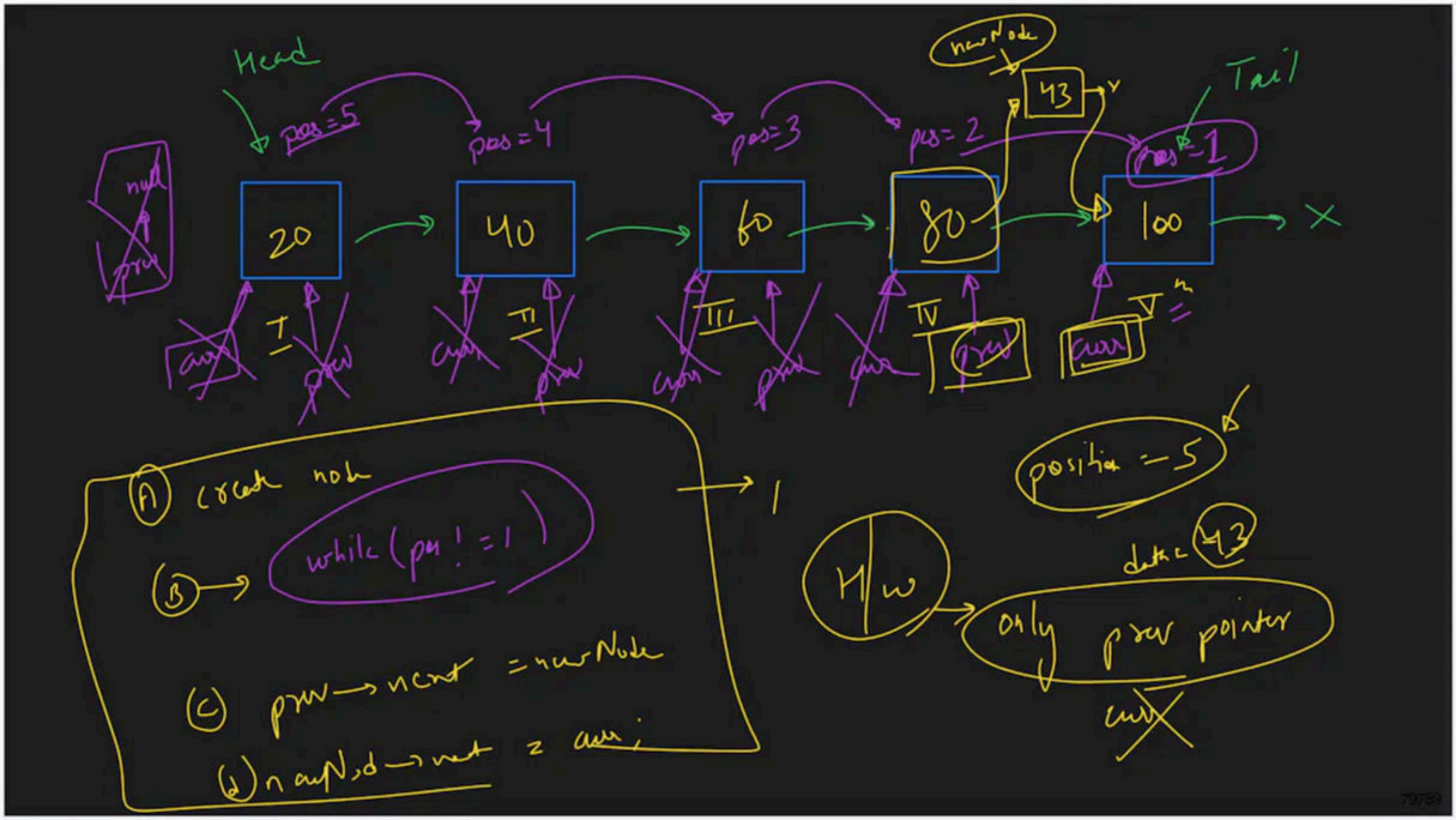
if (pos == len + 1) → insertAtTail()

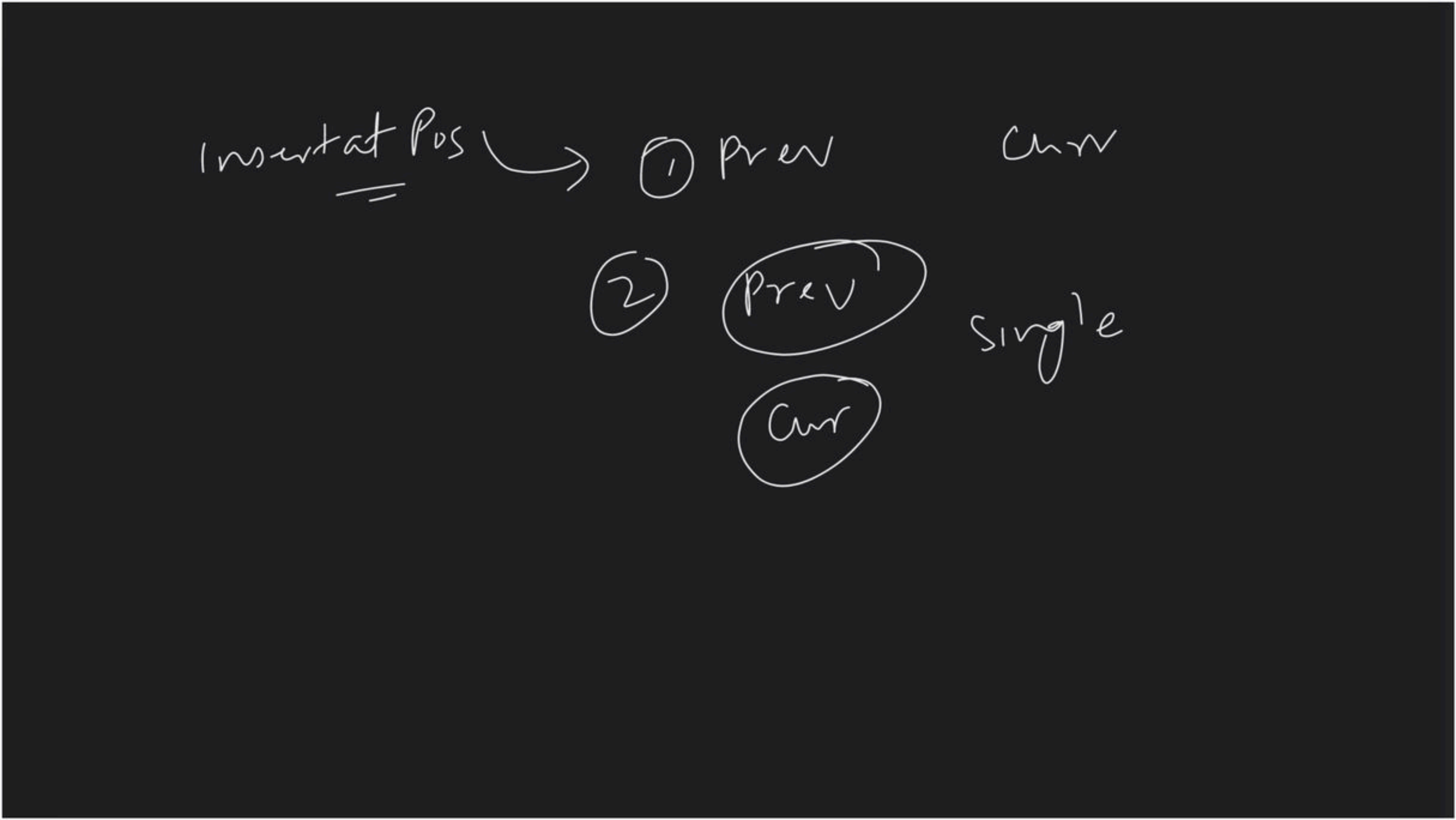


pos = 4th

pos = len + 1


pos →
1
2
3
4
len+1
x
insert Middle → logic
pos
1
2
3
4
len+1
insert AtHead
insert AtTail


Head
pos=5
pos=4
pos=3
pos=2
pos=1
newNode
43
Tail
null
prev
20
40
60
80
100
X
I
II
III
IV
V
curr
prev
A) create node
B)
while (prev != 1)
1
C) prev → next = newNode
D) newNode → next = curr
position = 5
data = 43
H/W
only prev pointer
curr


insertatPos
1
Prev
Curr
2
Prev
Curr
single

=> insertAtPos ==> L

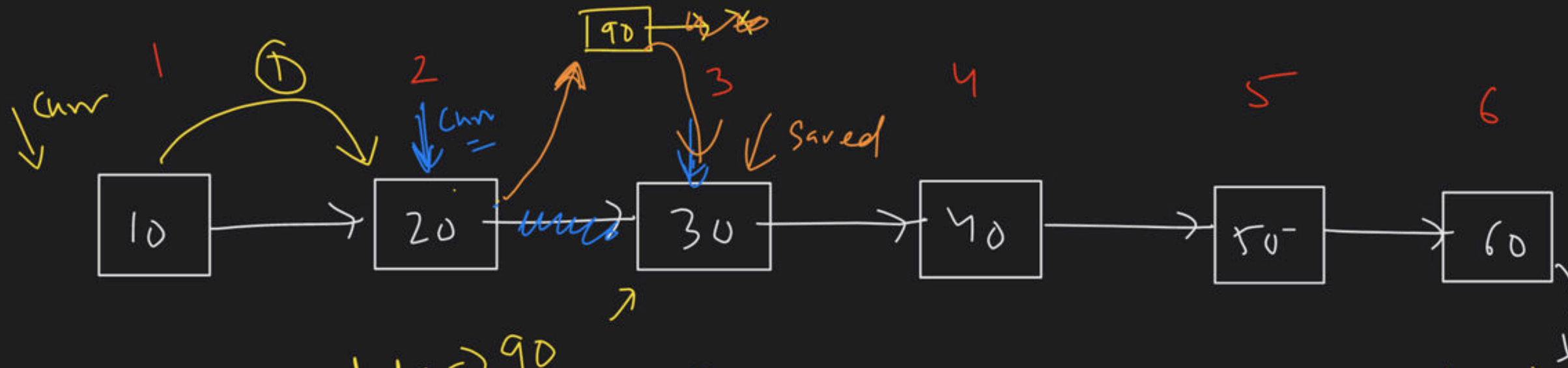
=>

① len = find(length)

② pos == 1
↳ insert At head

③ pos > l ⟶ insert At Tail

④ Else insert at middle



data => 90

=> Pos = 3 =>

① newNode = new Node(90)

② Curr = head;

for(i=0; i<Pos-1; i++)
{
Curr = curr->next
}

Node *Nextnode = Curr->next;

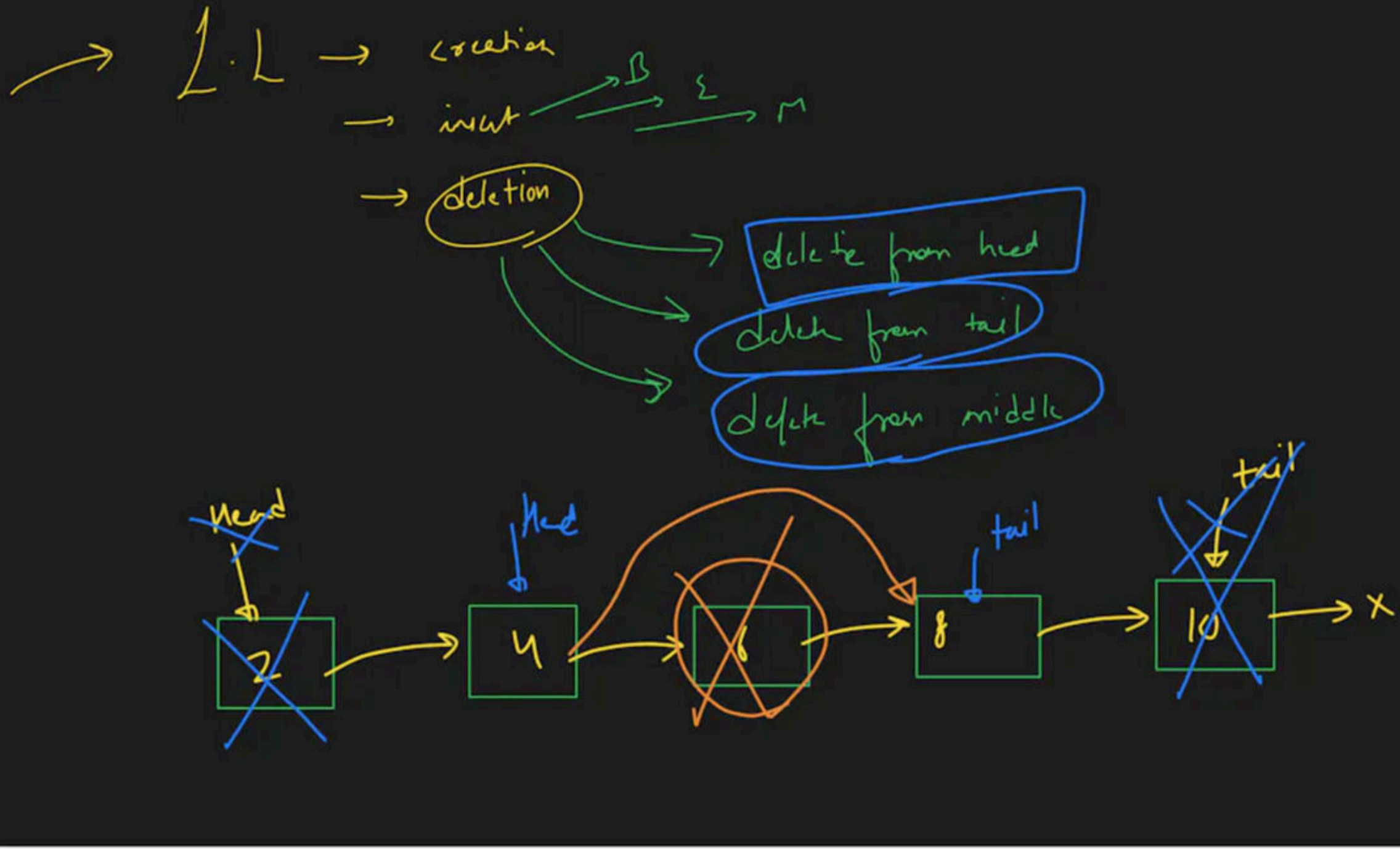
Curr->next = newNode;

newNode->next = nextNode;

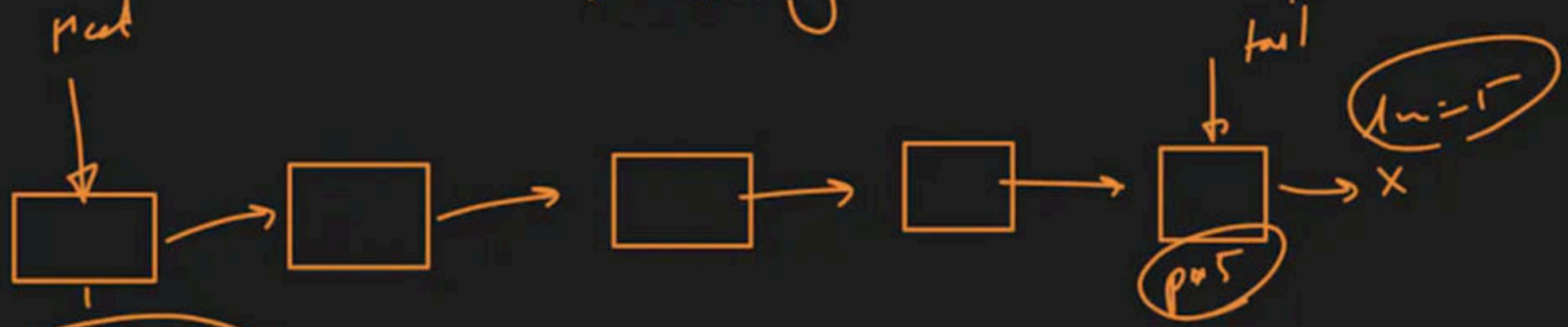
Linked List - Class 2

Special class

Love Babbar • Nov 4, 2023


L.L
creation
insert
B
E
M
deletion
delete from head
delete from tail
delete from middle
head
head
tail
tail
2
4
6
8
10
X

L.L → deletion → head
→ tail
→ middle
→ deleteNode(position)

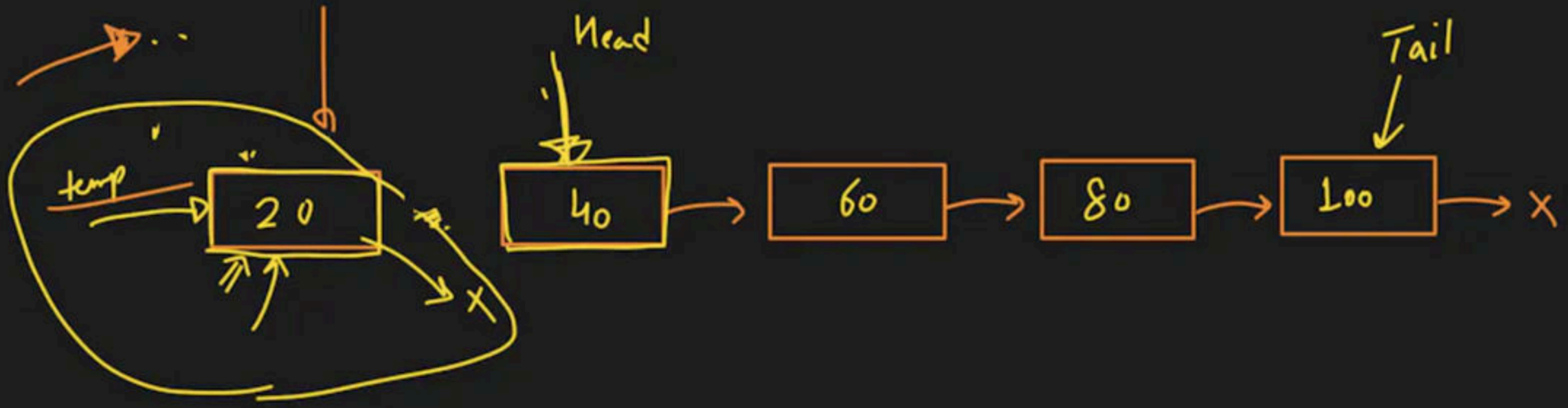


① Empty list → head == NULL

② if (pos == 1) → delete from head

③ if (pos == len) → delete from tail

else → middle

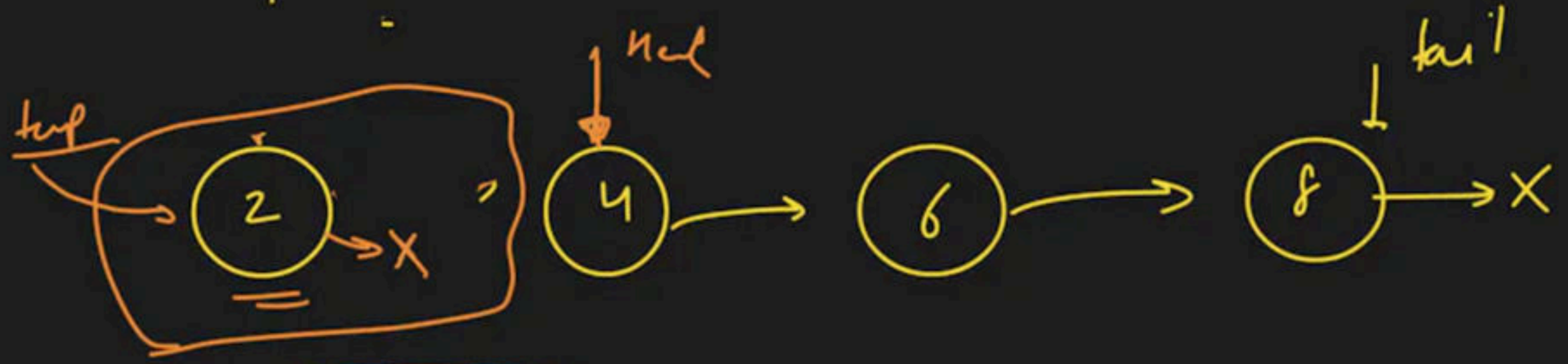


(A) Node * temp = head

(B) head = temp → next

(C) temp → next = NULL

(D) delete temp

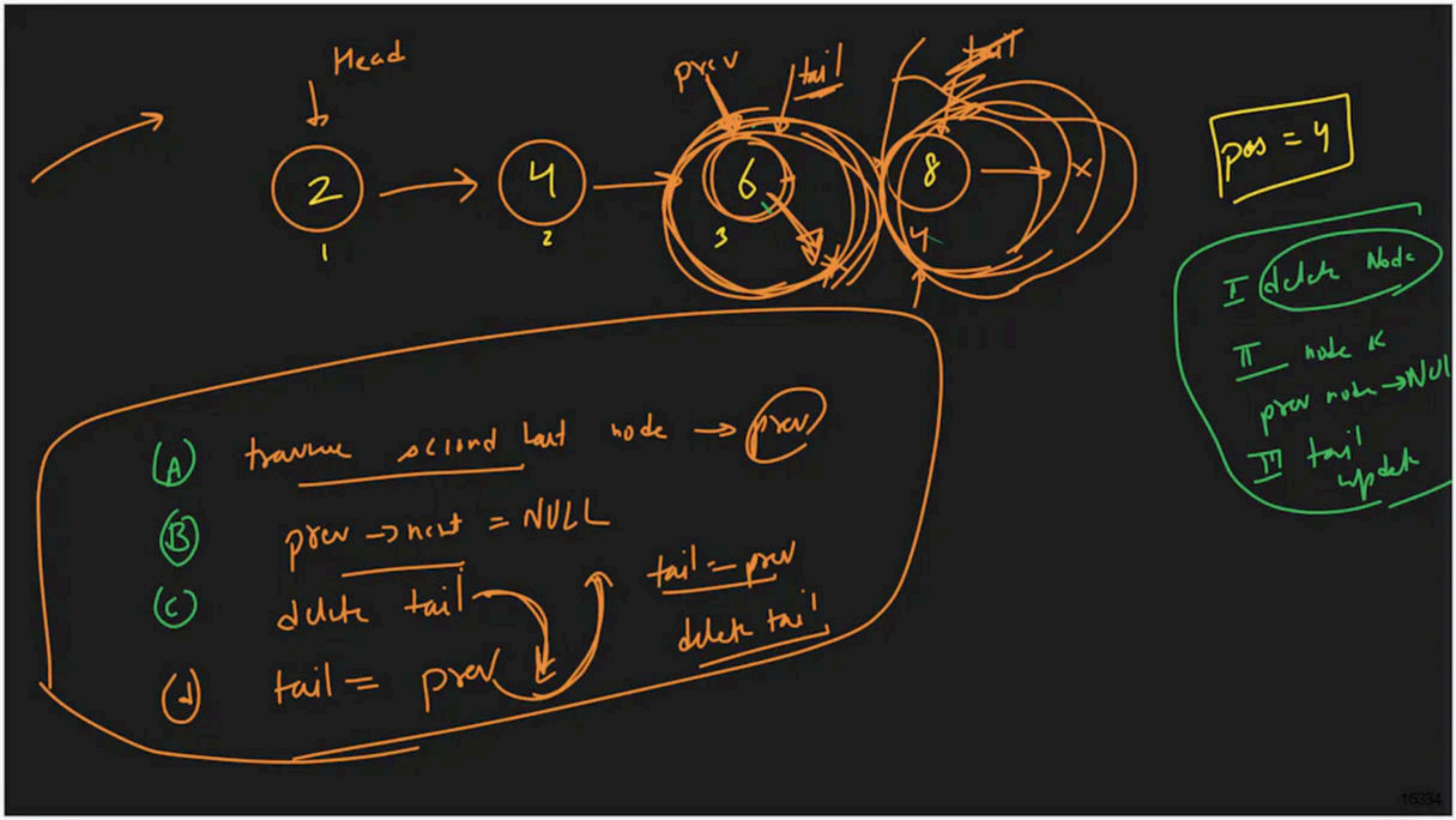


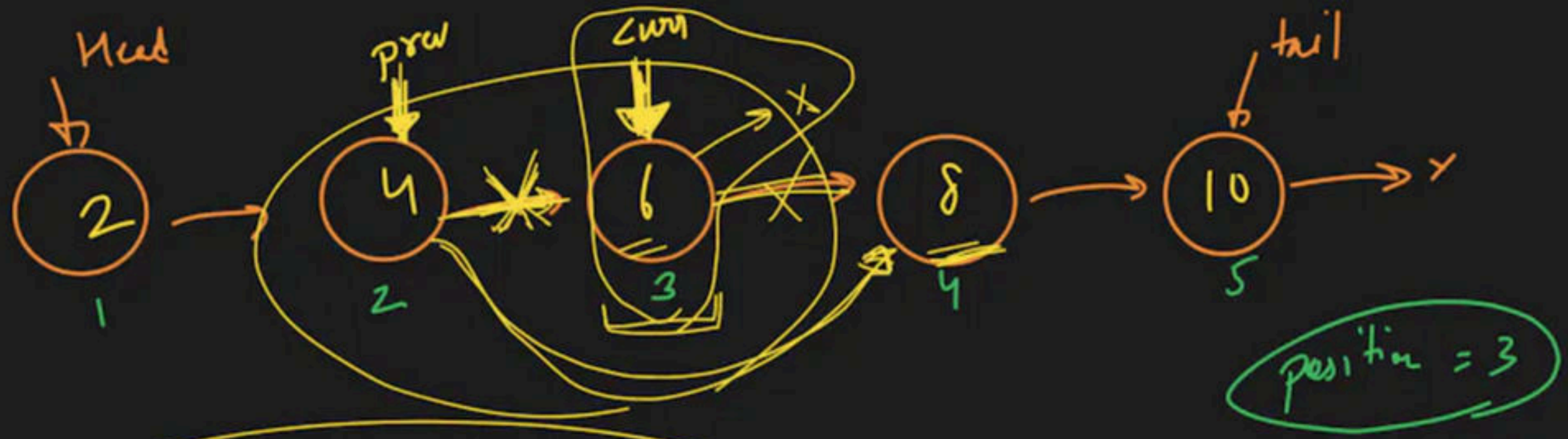
(A) Node * temp = head

(B) Head = Head -> next;

(C) temp -> next = NULL;

(D) delete temp


Head
prev
tail
temp
2
4
6
8
1
2
3
4
pos = 4
I delete Node
II node ka prev node → NULL
III tail update
(A) traverse second last node → prev
(B) prev → next = NULL
(C) delete tail
(D) tail = prev
tail = prev
delete tail

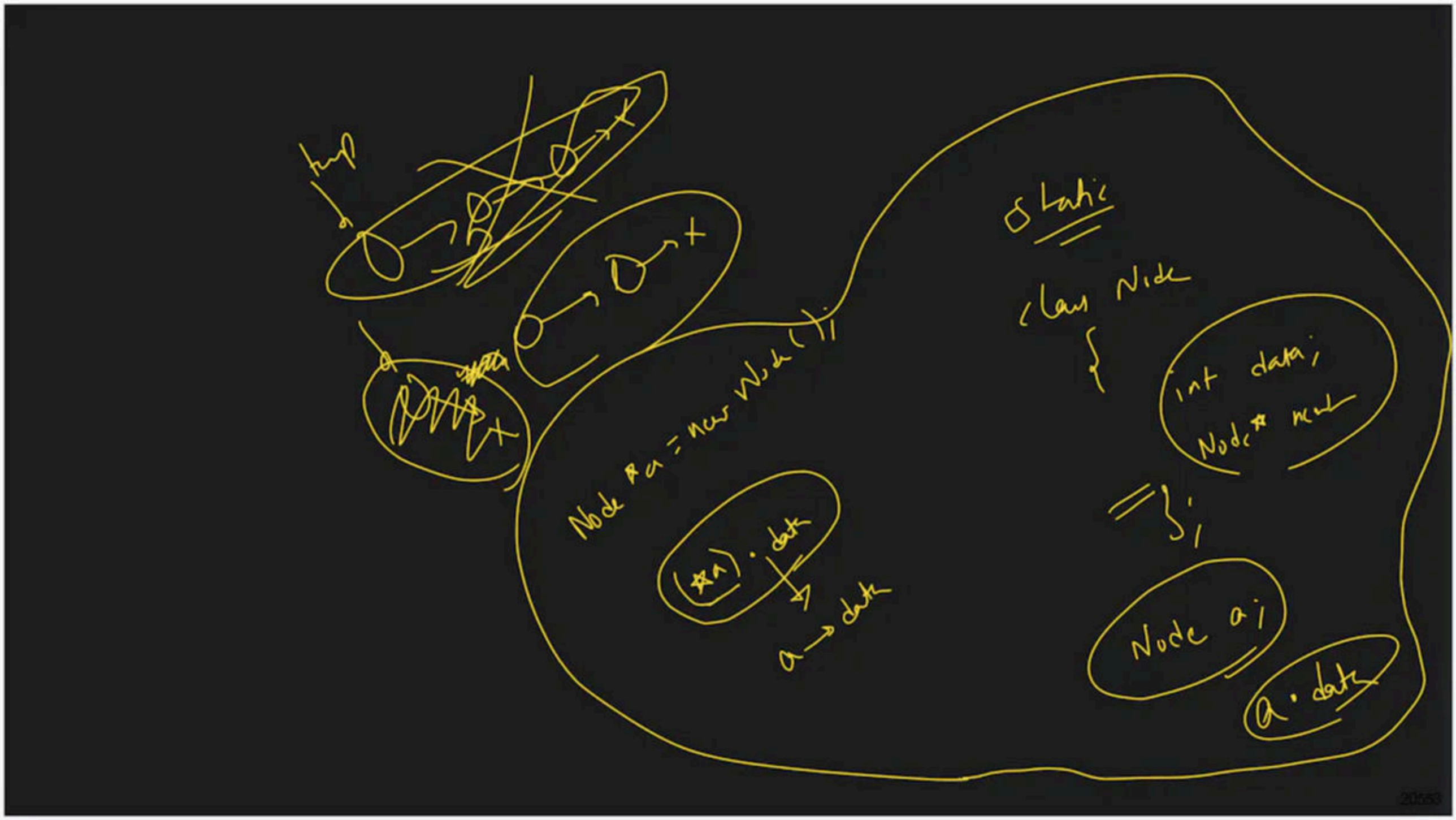


(1) traverse LL for prev / curr

(2) prev -> next = curr -> next

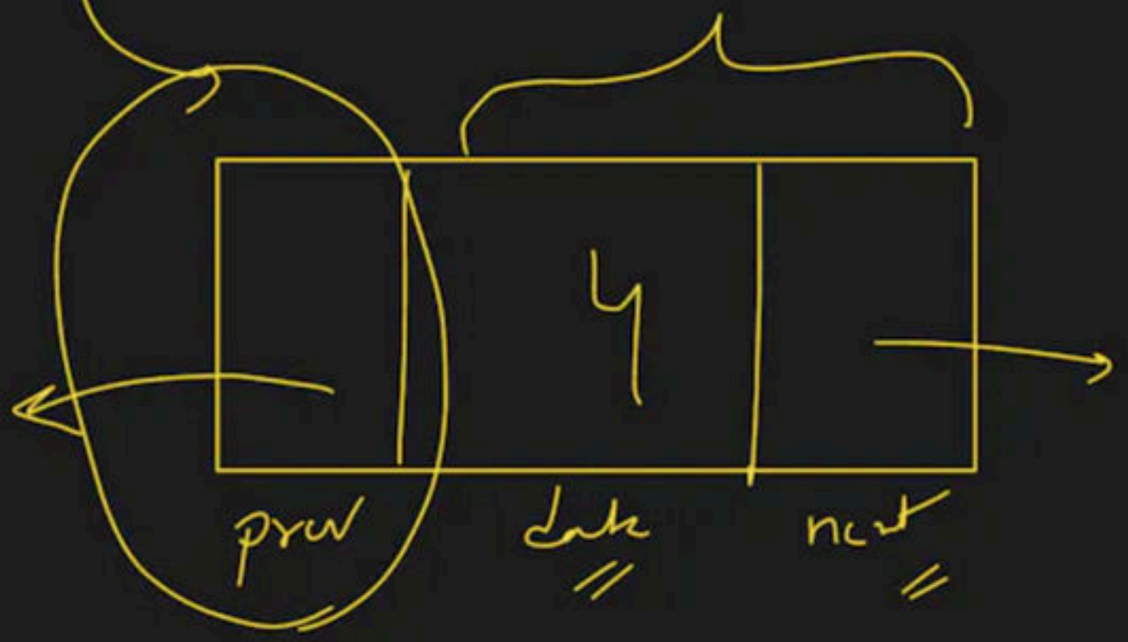
(3) curr -> next = null

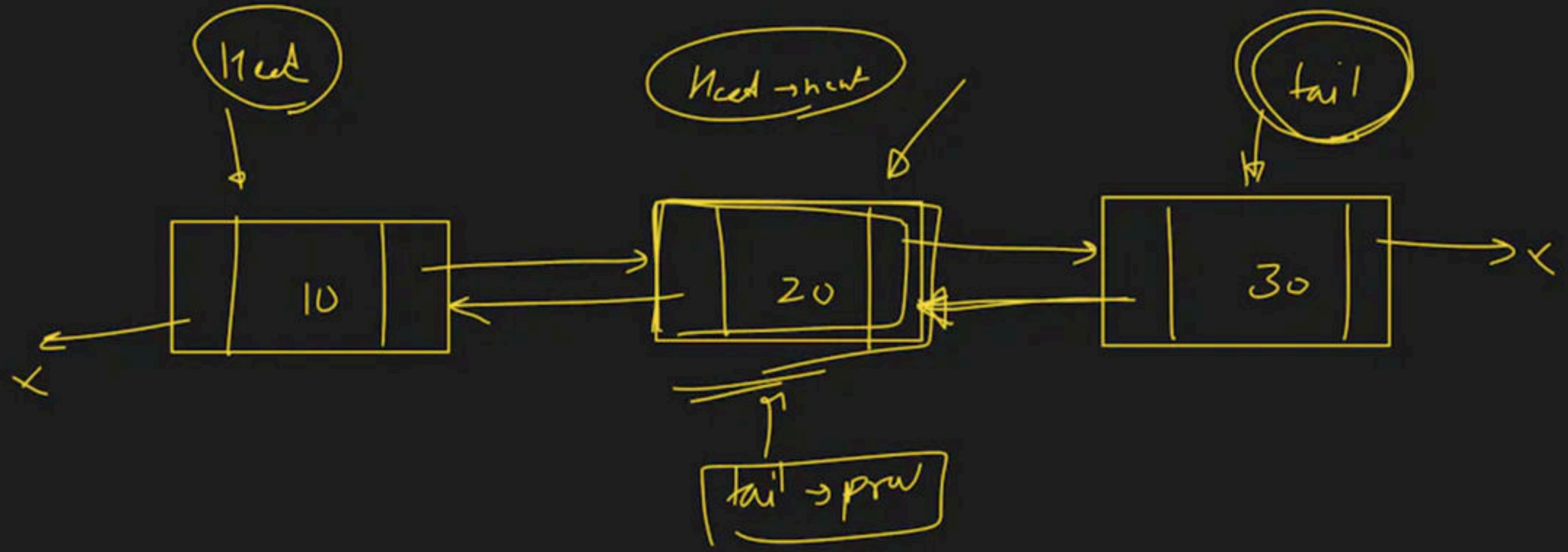
(4) delete curr

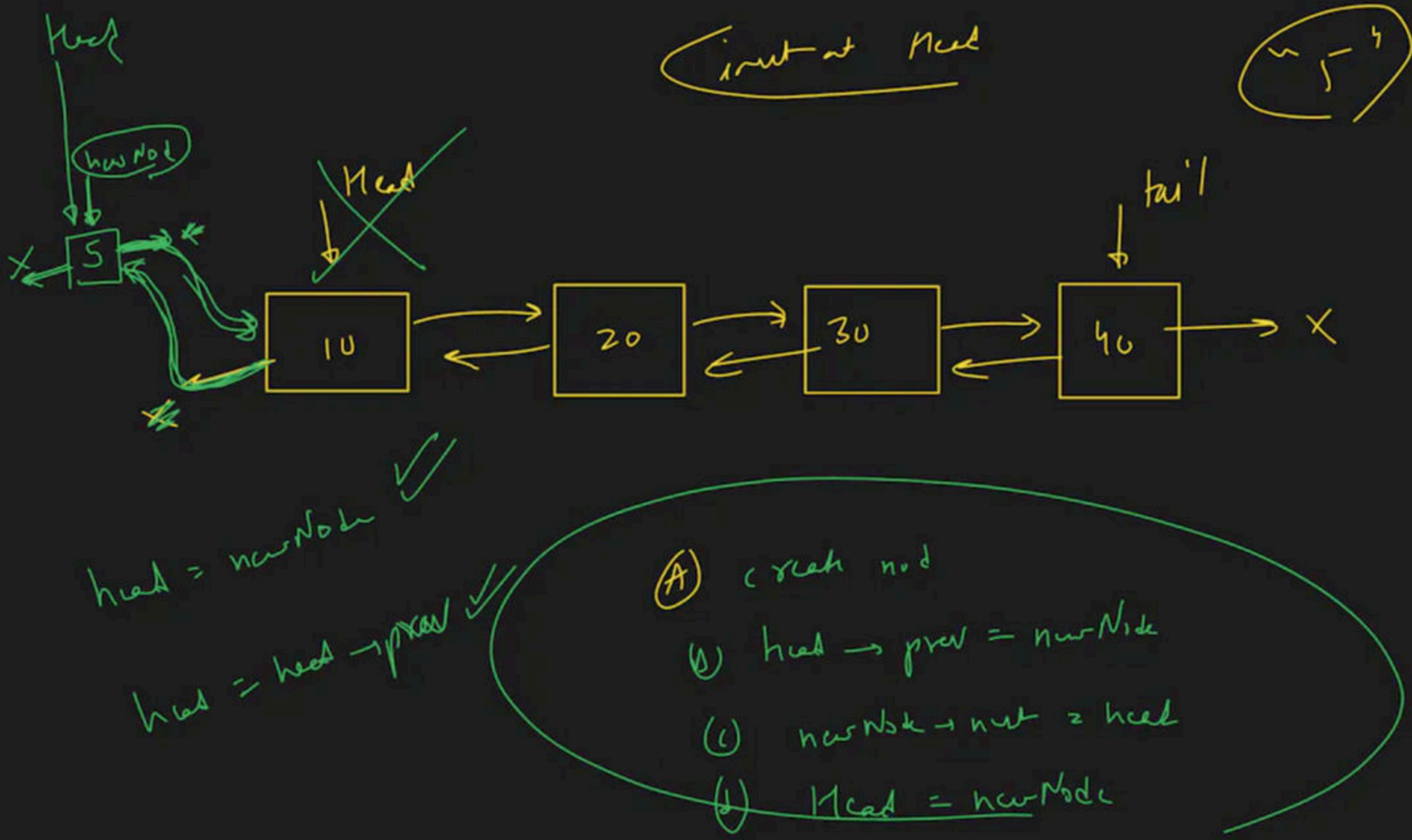

Static
class Node
{
int data;
Node* next
};
Node a;
(*a).data
a->data
Node ai
a.data

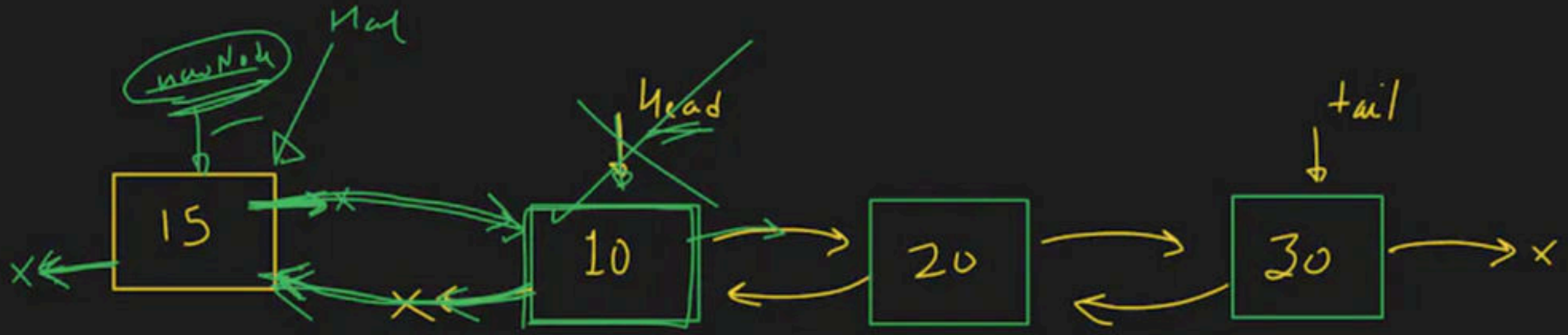
Doubly Linked List

Node




head
head->next
tail
10
20
30
X
X
tail->prev


Head
newNode
5
Head
10
20
30
40
tail
X
insert at Head
head = newNode
head = head → prev
A create node
B head → prev = newNode
C newNode → next = head
D Head = newNode

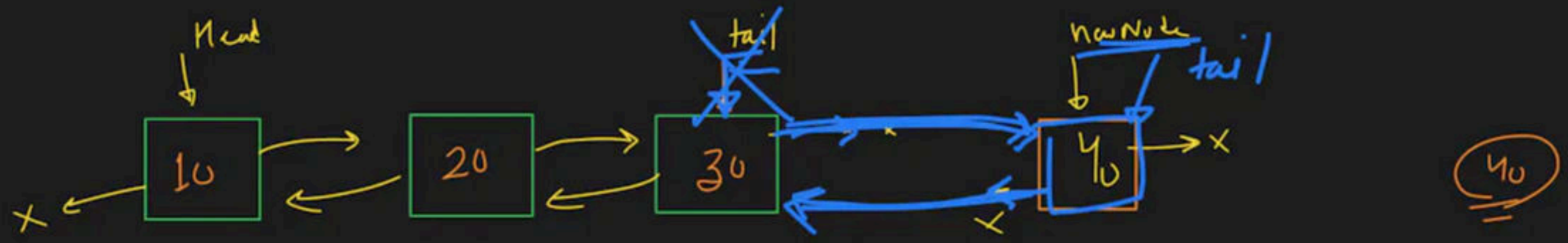


(A) Create Node → Node* newNode = new Node(15);

(B) newNode → next = head;

head → prev = newNode

Head = newNode

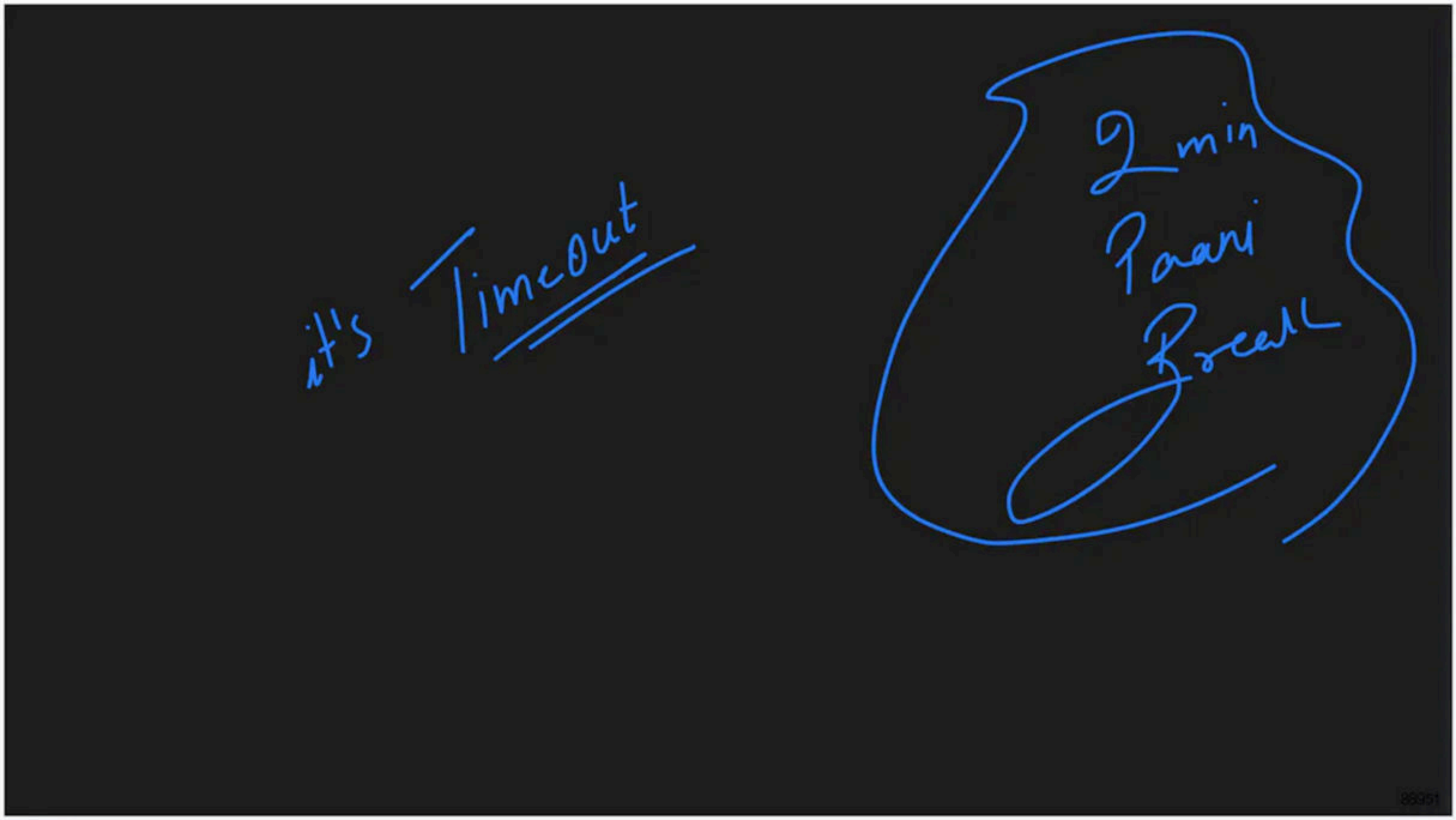


(A) create Node

tail->next = newNode

newNode->prev = tail

tail = newNode


It's Timeout
2 min
Paani
Break

(A) Create a Node

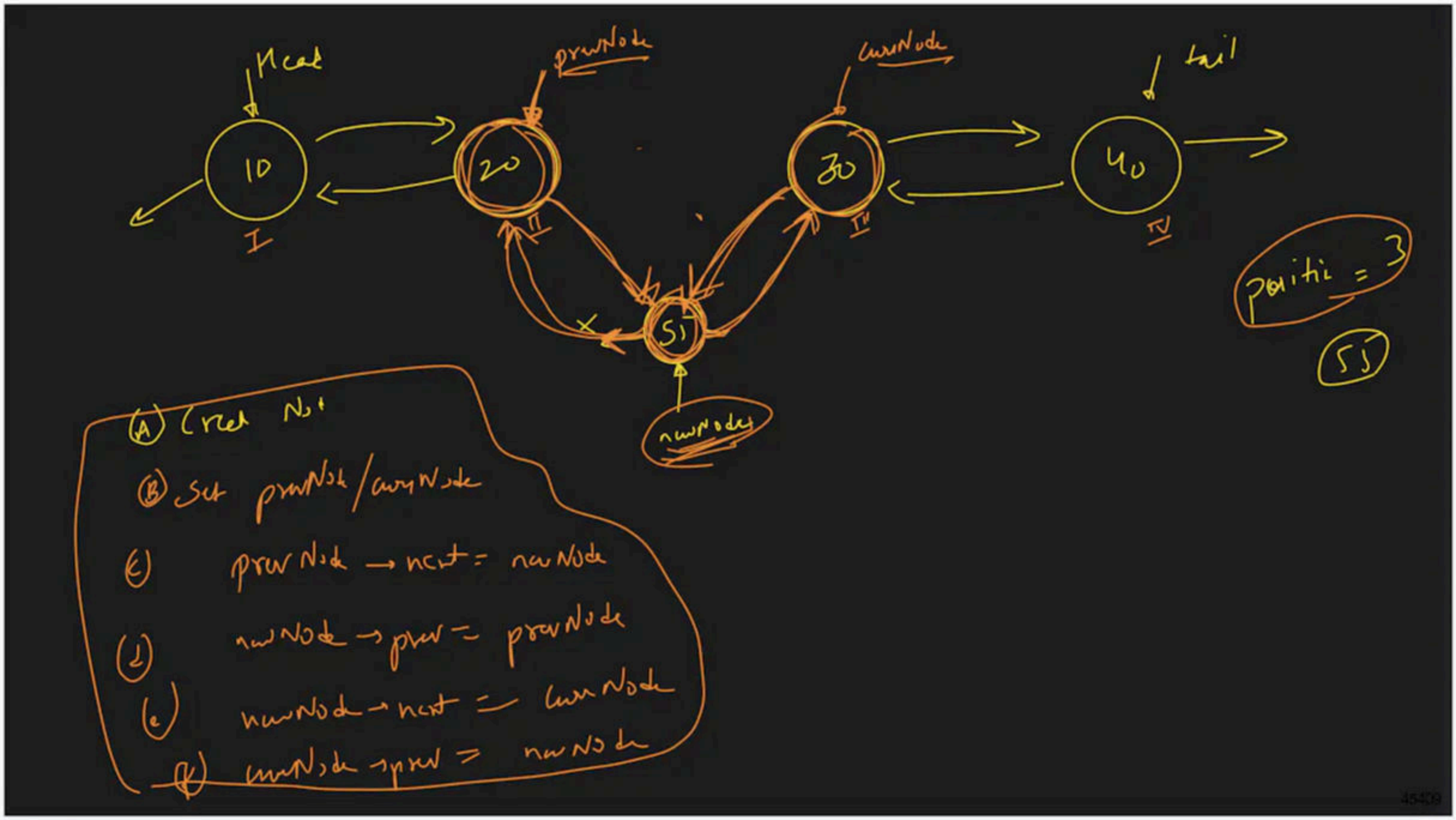
(B) Set prev/curr

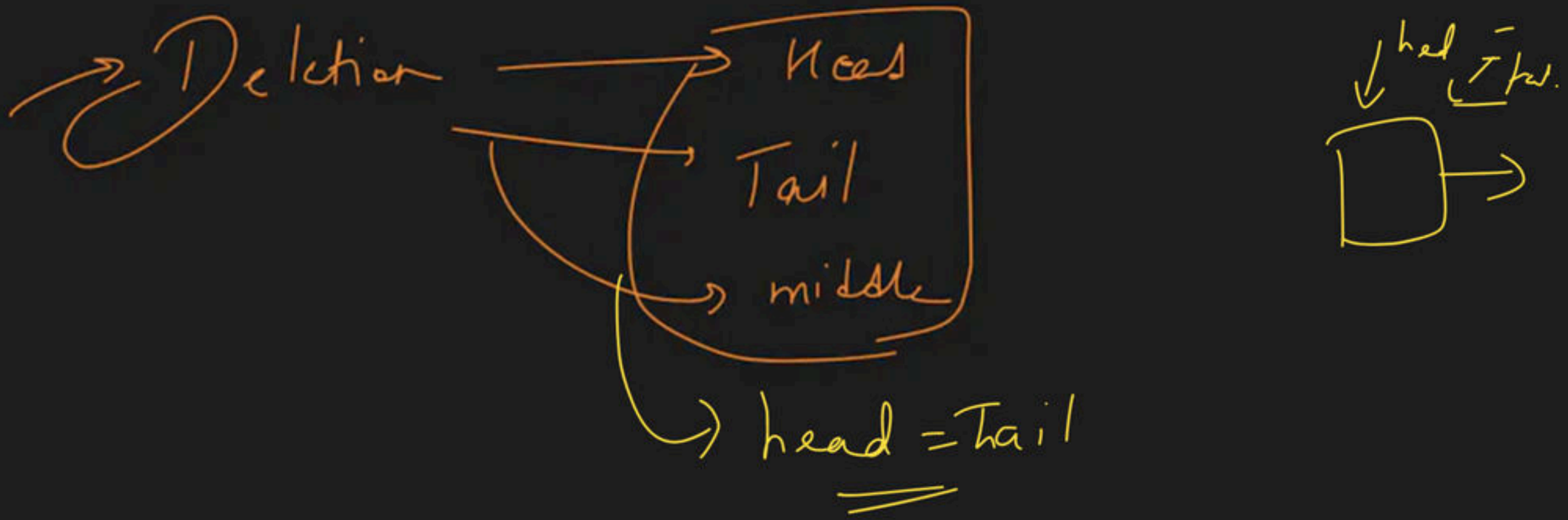
(C) prevNode → next = newNode

newNode → prev = prevNode

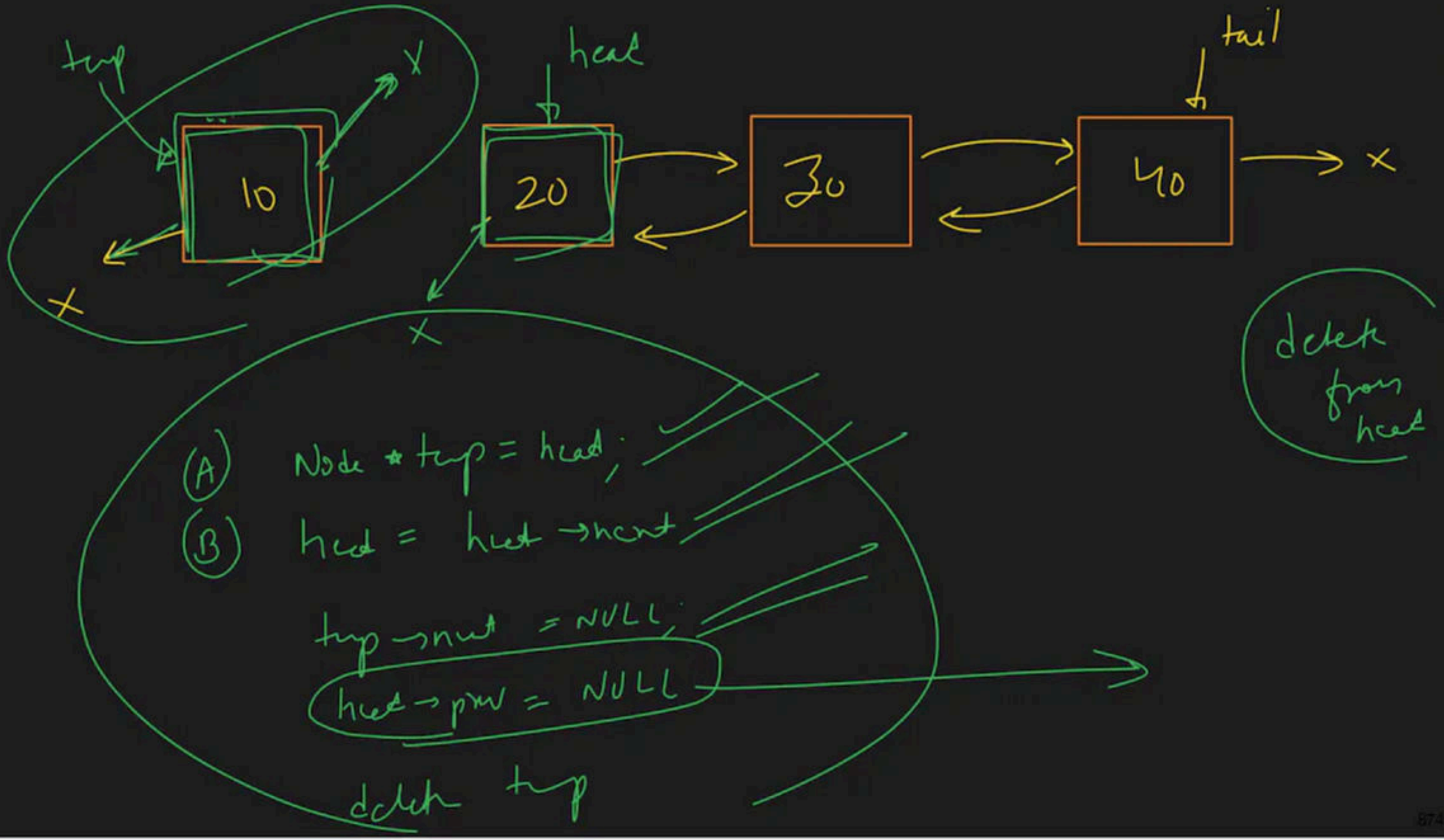
newNode → next = currNode

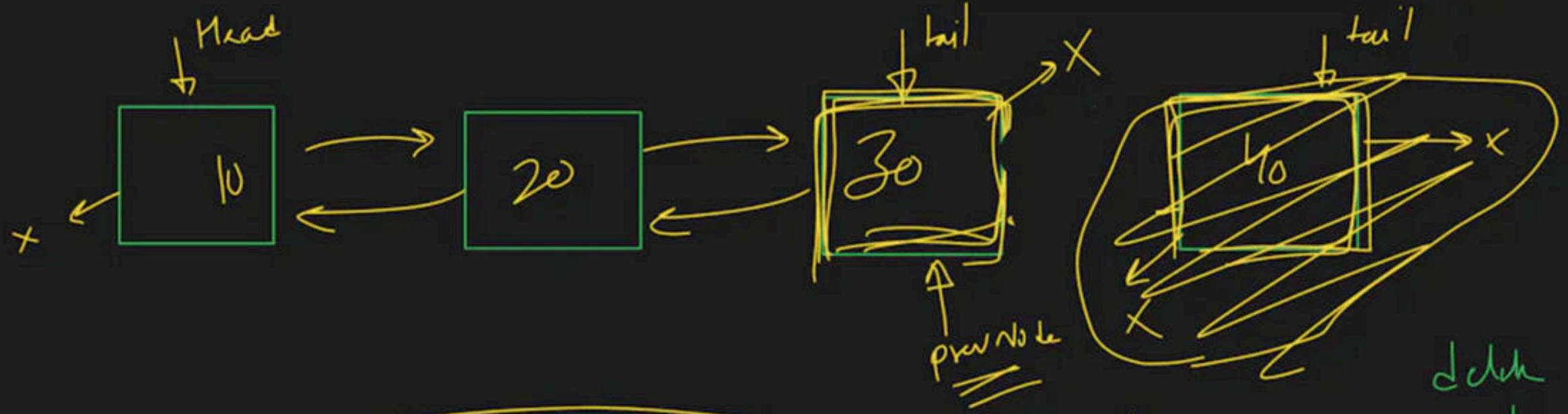
currNode → prev = newNode


Head
prevNode
currNode
tail
10
20
30
40
51
I
II
III
IV
newNode
position = 3
(3)
(A) Create Node
(B) Set prevNode / currNode
(C) prevNode → next = newNode
(D) newNode → prev = prevNode
(E) newNode → next = currNode
(F) currNode → prev = newNode


Deletion
Head
Tail
middle
head = Tail

head = Tail

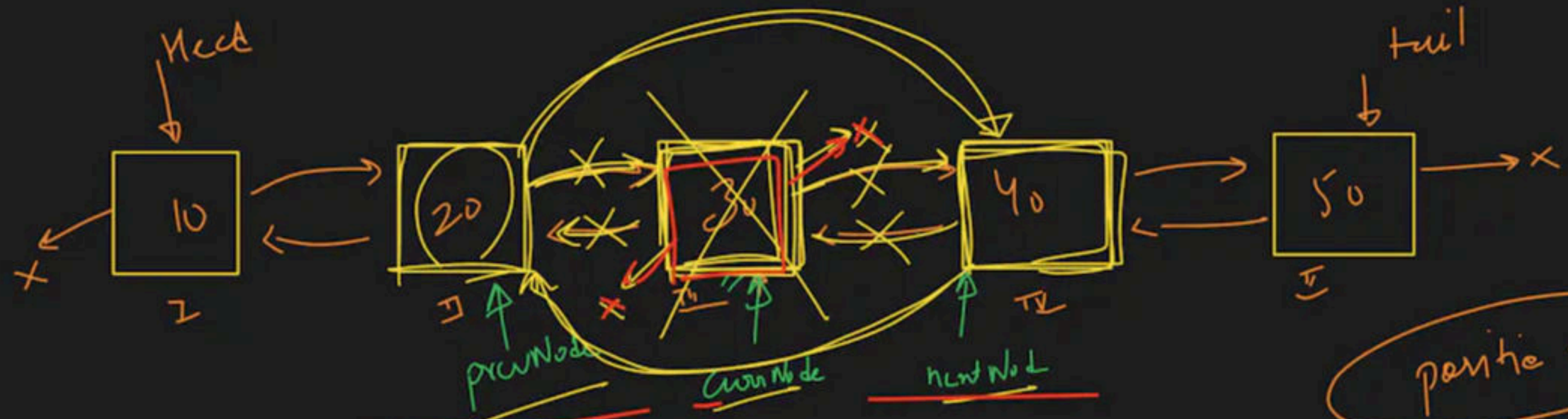

temp
head
tail
10
20
30
40
X
delete from head
(A) Node * temp = head;
(B) head = head->next
temp->next = NULL;
head->prev = NULL
delete temp



delete from tail (TOX)

(A)

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



position = 3

2 min Break

(A) Set prevNode / currNode / nextNode

(B) prevNode → next = nextNode

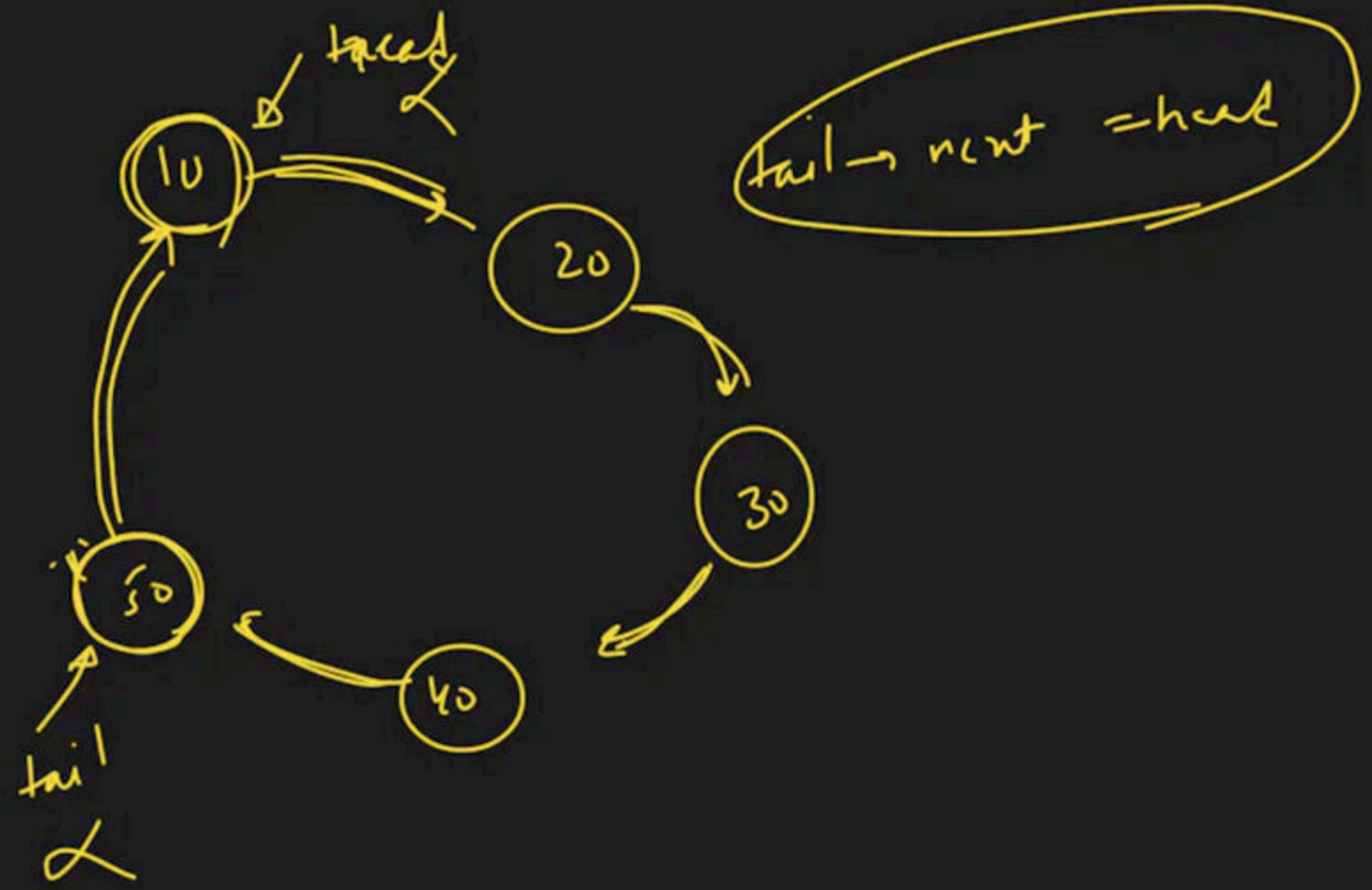
currNode → prev = NULL

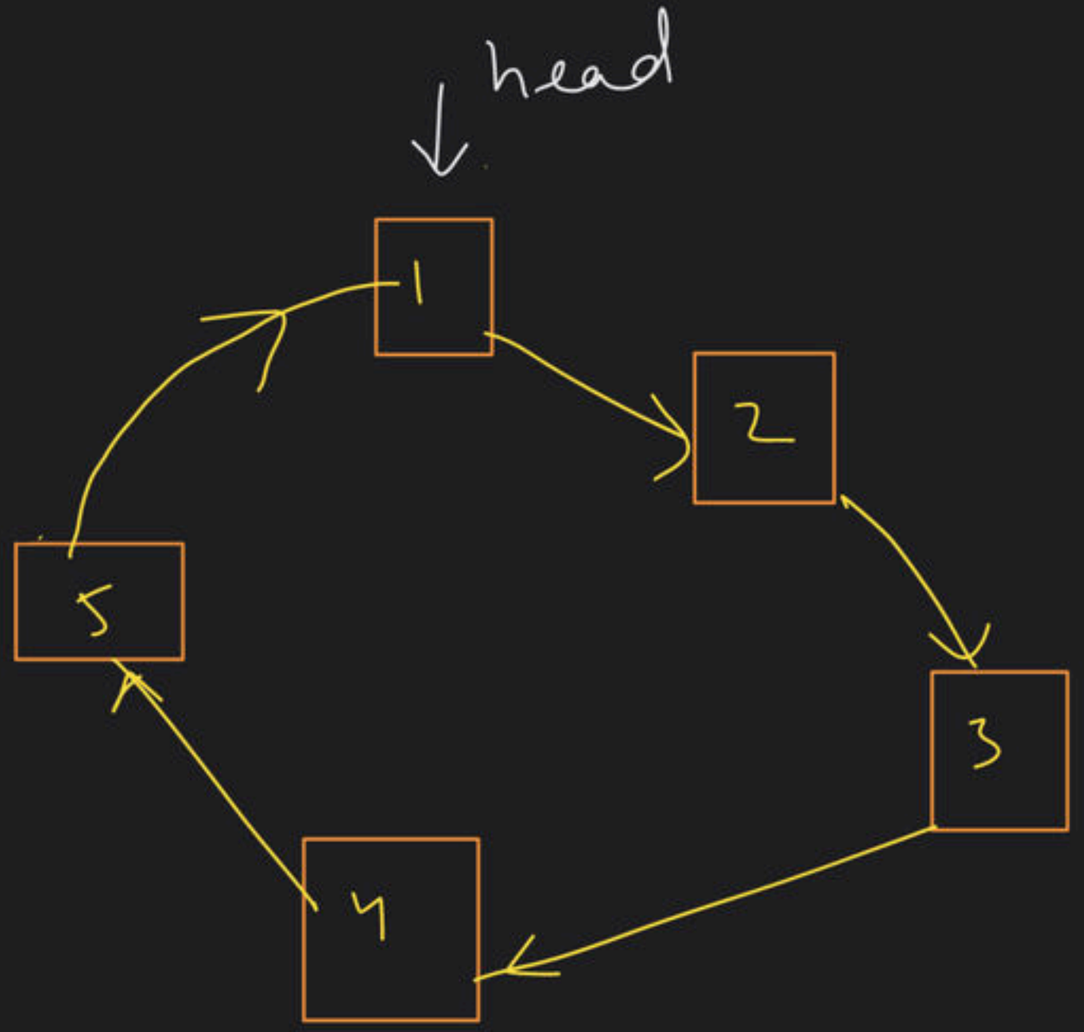
currNode → next = NULL

nextNode → prev = prevNode

delete currNode

Circular → Singly Linked List

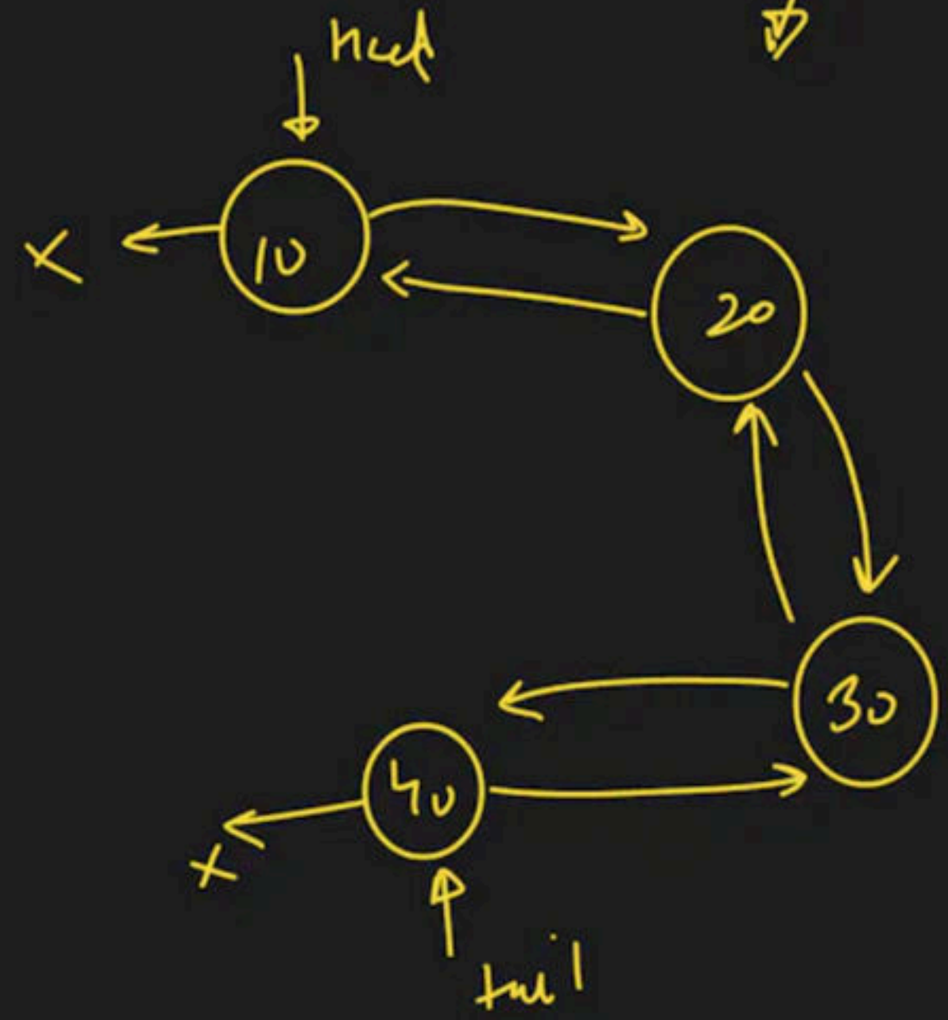





```
*it = head.

while ( 1 )
{
  cout << it -> data.
  =
  if(it -> next == head)
  {
    Break;
  }
  it = it -> next;
  =
}
```

Circular → Doubly L.L

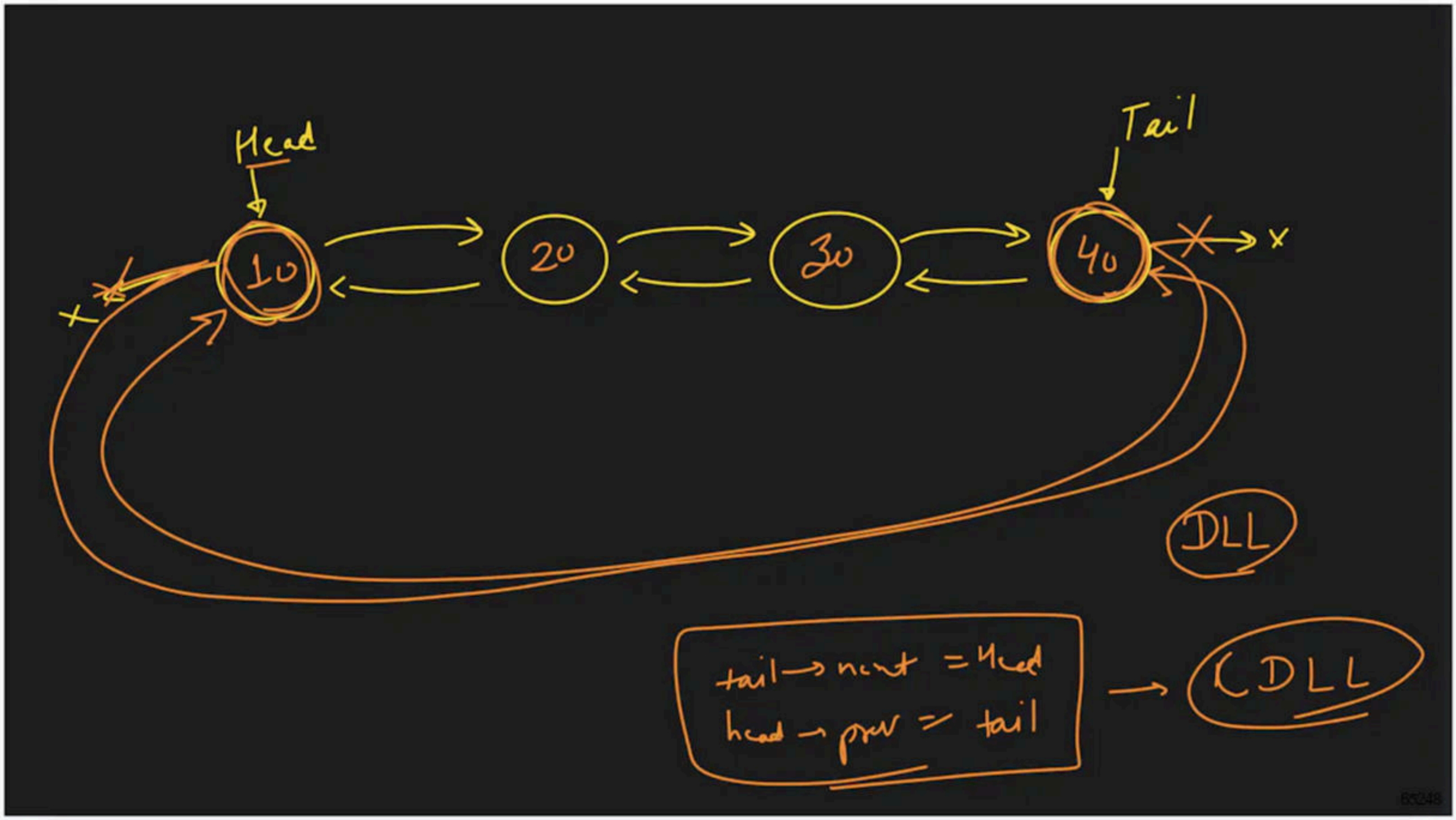


Marathon

→ L.B

↓

L.L → Queue


Head
Tail
10
20
30
40
x
x
DLL
tail → next = Head
head → prev = tail
CDLL


detect & delete
loop

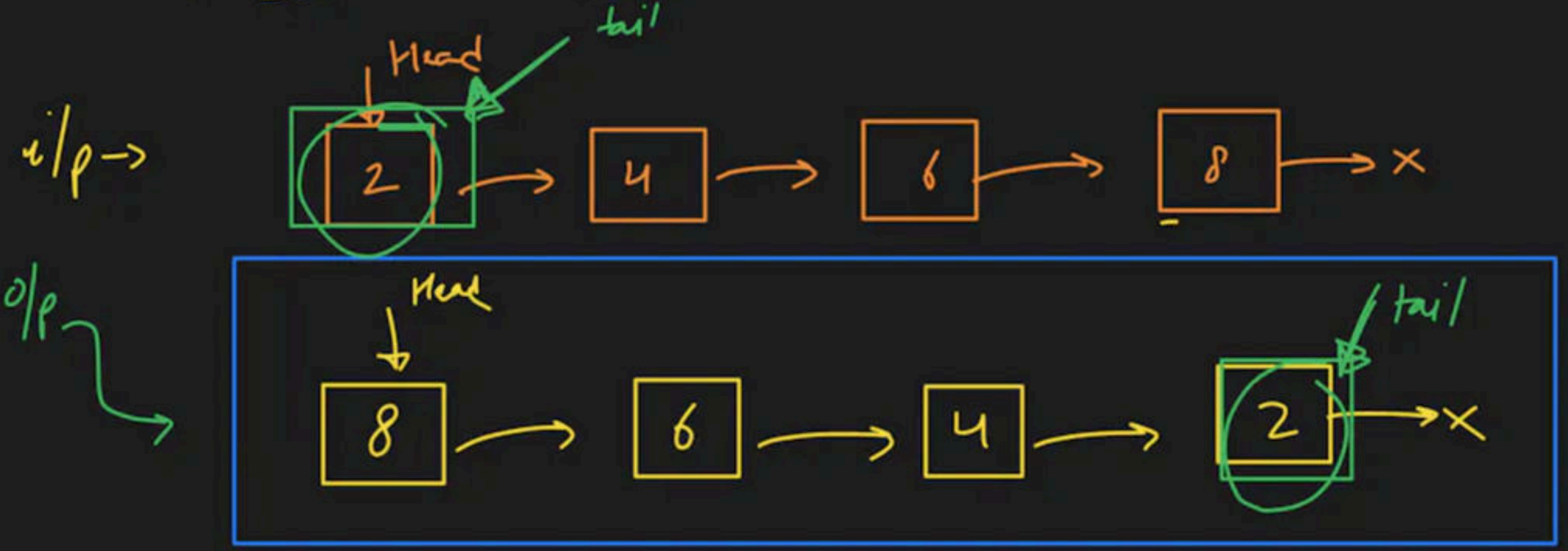

$\mathcal{G}_{pm}$
→ L L → III

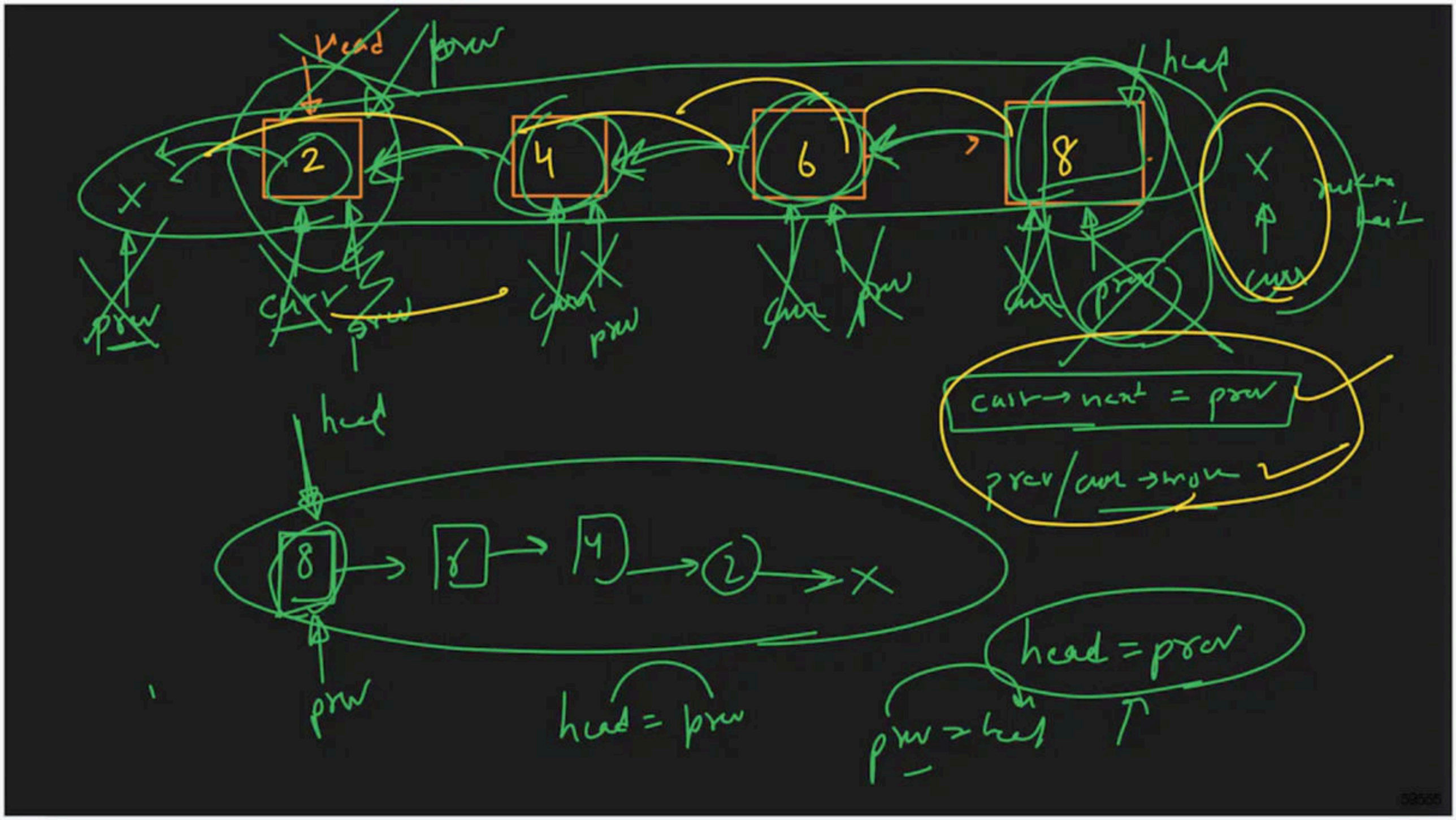
Linked List - Class 3

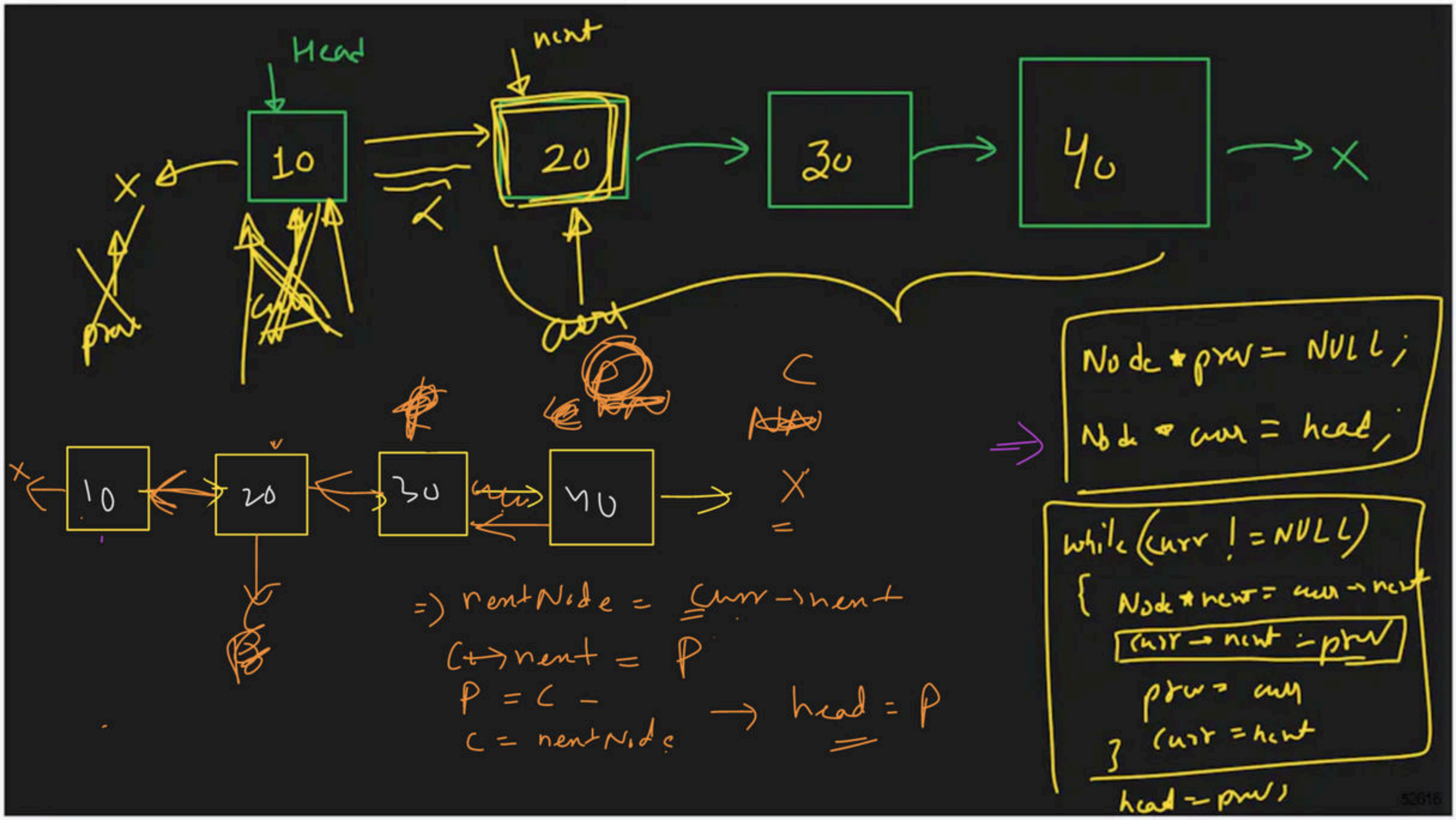
Special class

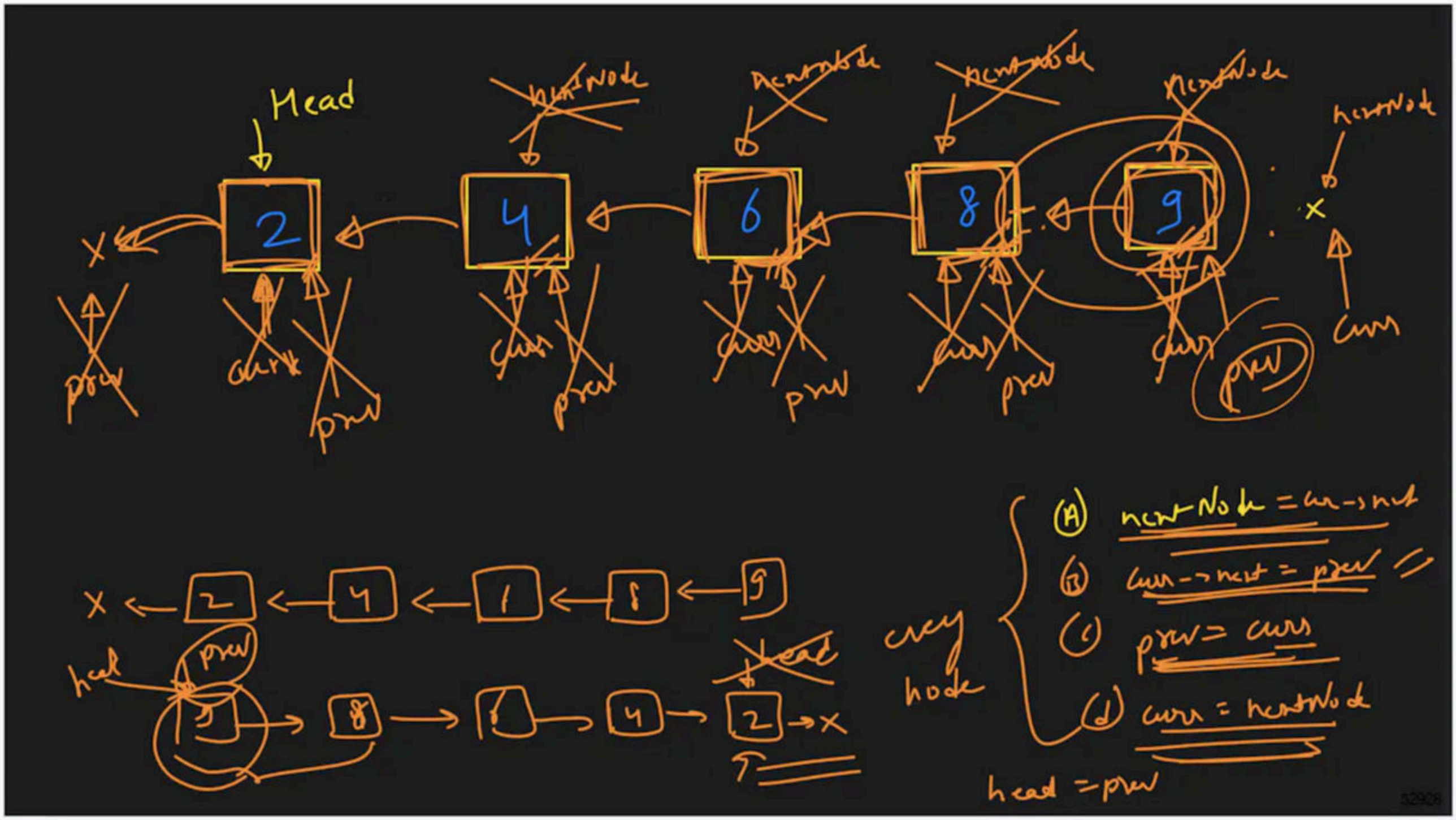
Love Babbar • Nov 5, 2023

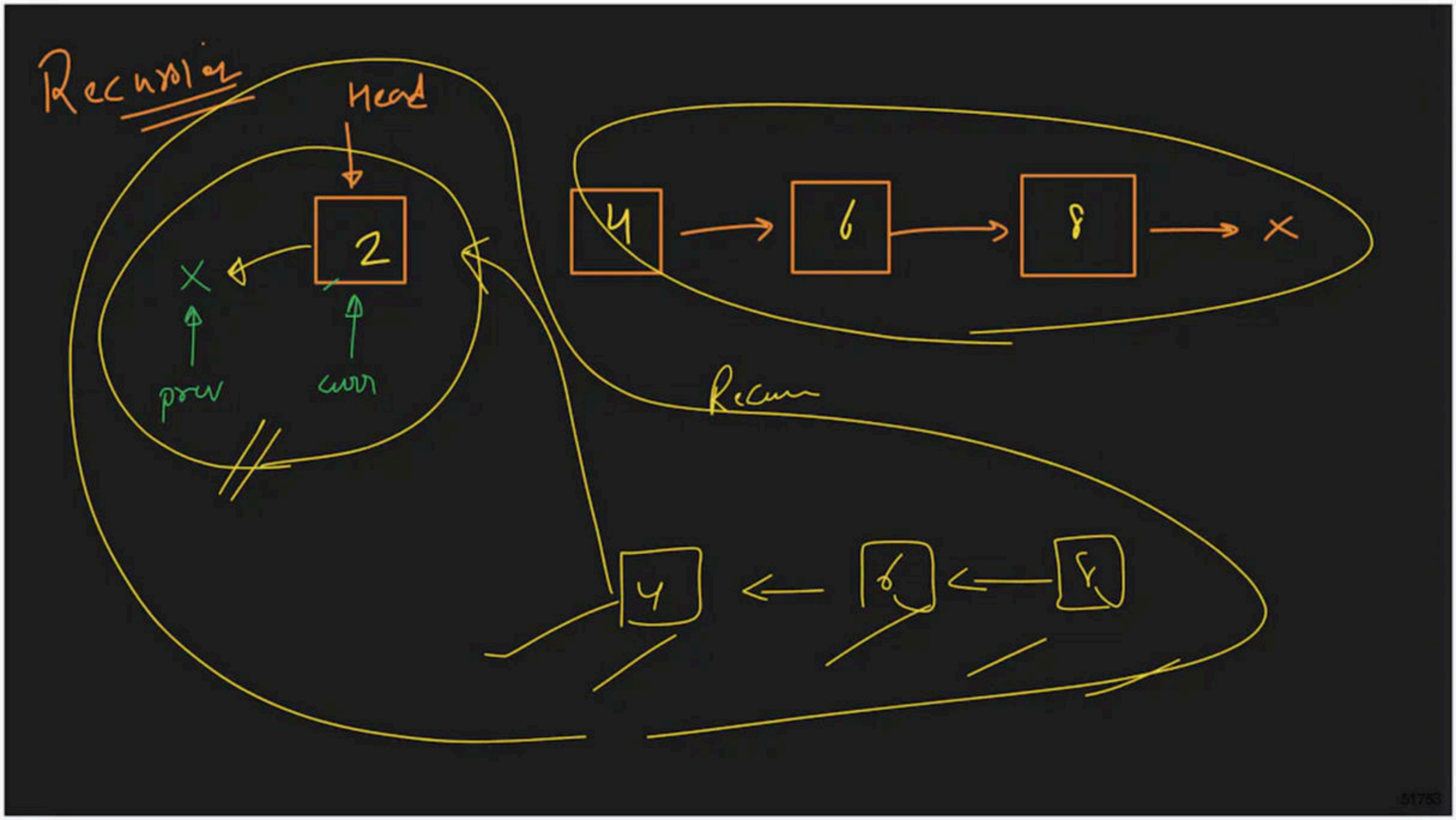
Reverse a Linked List

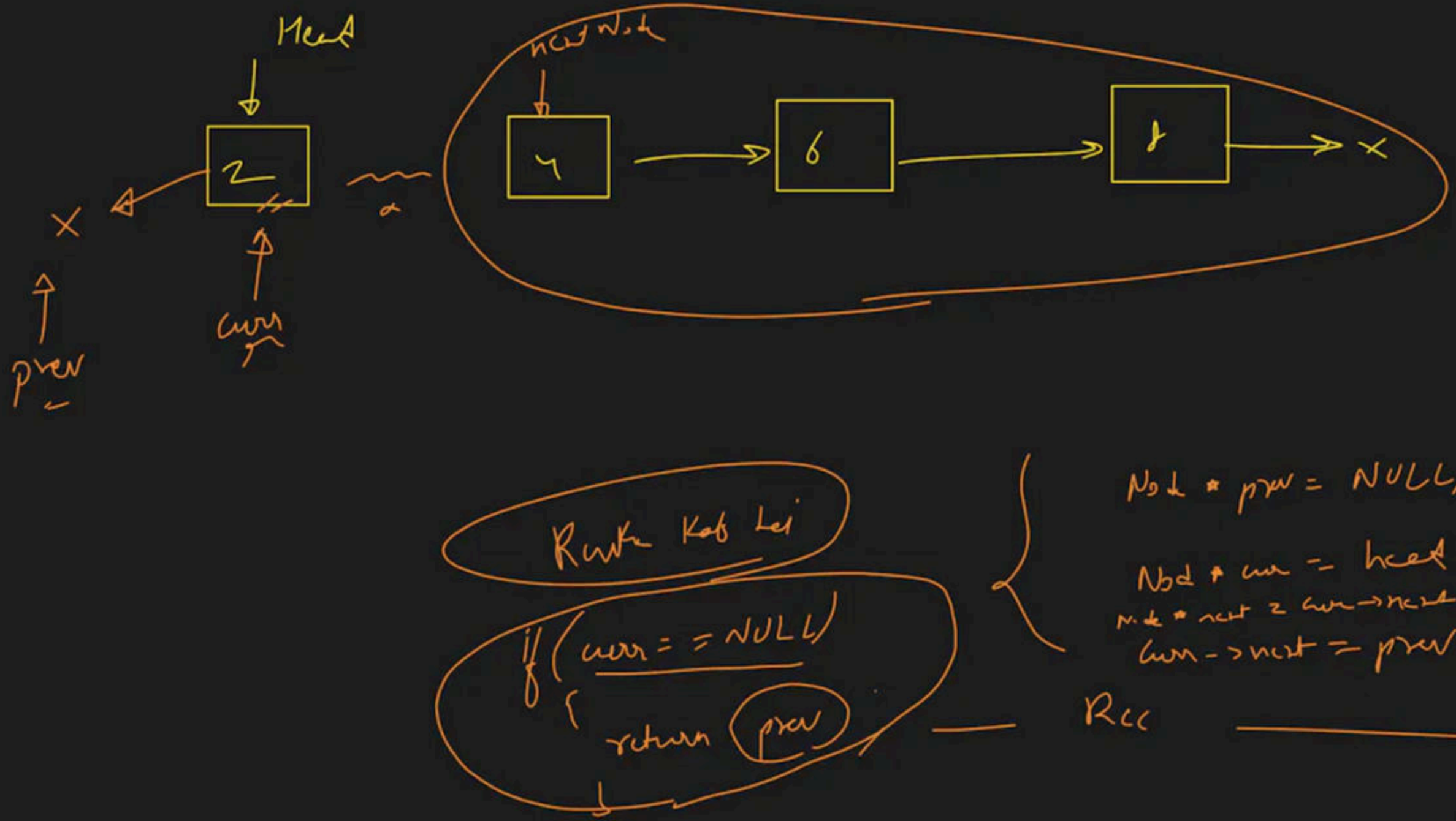


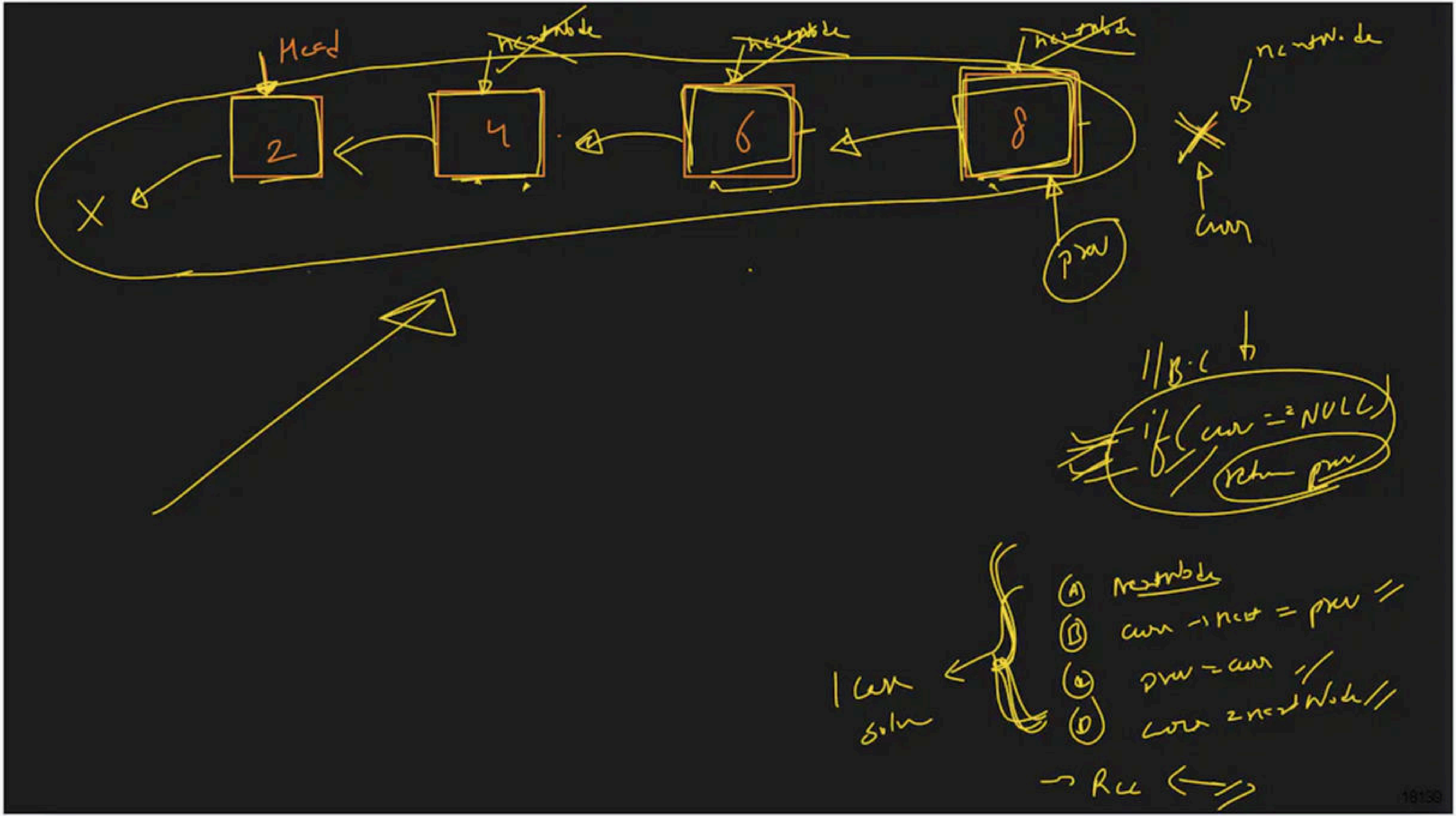

head
prev
2
4
6
8
X
curr
prev
curr->next = prev
prev/curr->next = 2
head
8
4
2
X
prev
head = prev
head = prev
prev = head


Head
next
10
20
30
40
X
prev
curr
C
P
10
20
30
40
X
=) nextNode = curr->next
C->next = P
P = C
C = nextNode
→ head = P
Node *prev = NULL;
Node *curr = head;
while (curr != NULL)
{ Node *next = curr->next
curr->next = prev
prev = curr
curr = next
}
head = prev;

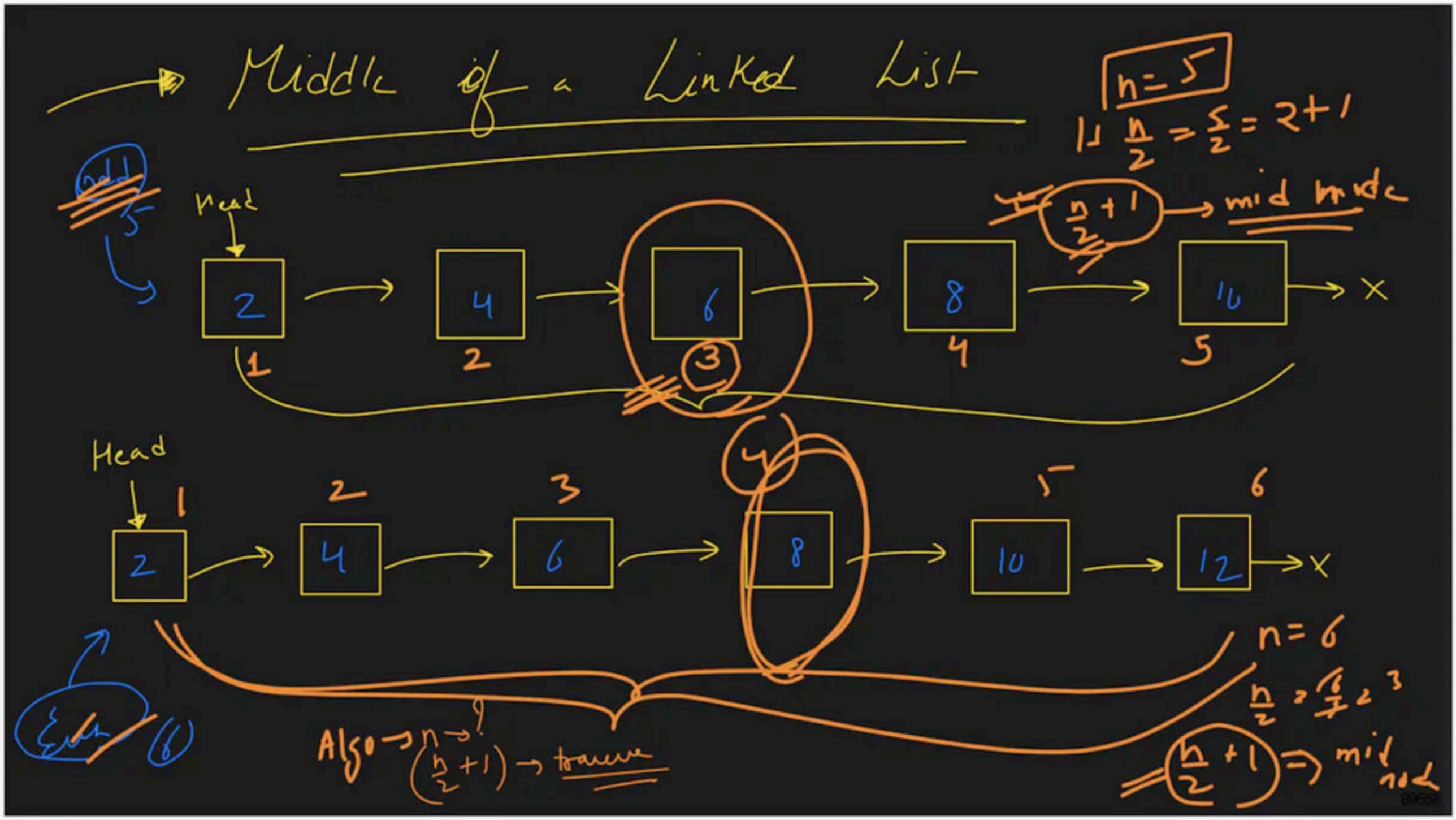

Head
2
4
6
8
9
curr
prev
nextNode
x
every node
(A) nextNode = curr->next
(B) curr->next = prev
(C) prev = curr
(D) curr = nextNode
head = prev

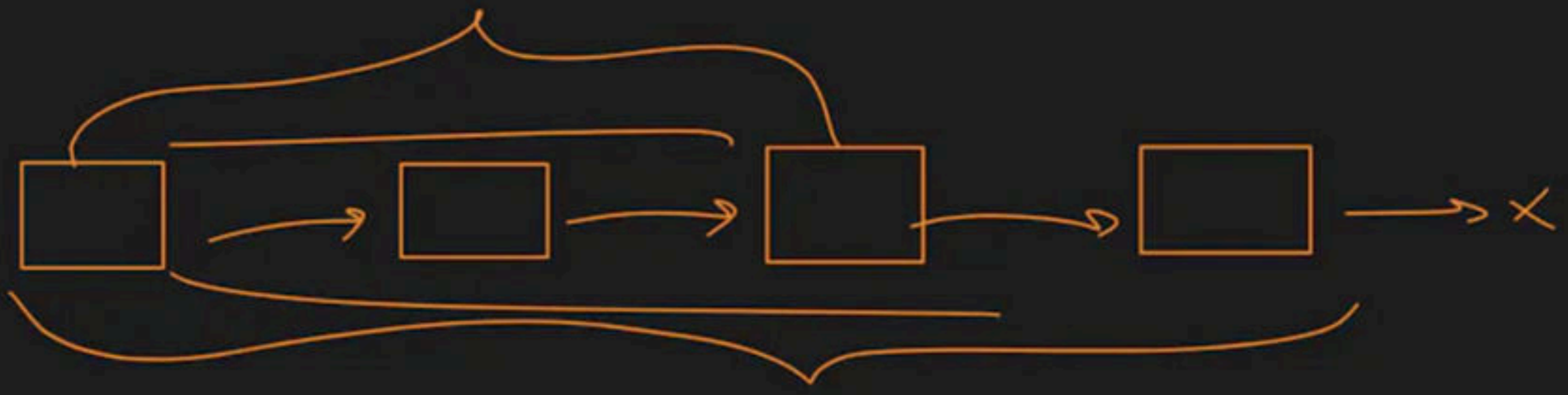

Recursion
Head
2
X
prev
curr
4
6
8
X
Recurr
4
6
8

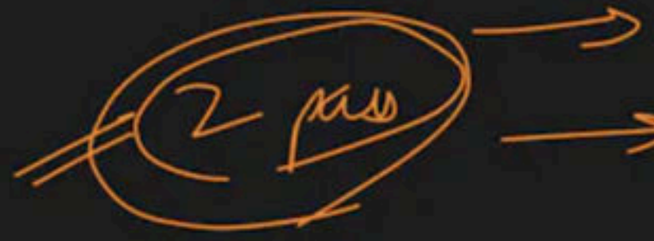

Head
next Node
4
6
1
2
x
x
prev
curr
Node * prev = NULL;
Node * curr = head
Node * next = curr->next
curr->next = prev
Rule Kaf Lei
if (curr == NULL)
{
return prev
}
Rcc


Head
nextNode
2
4
6
8
prev
curr
//B.C
if(curr == NULL)
return prev
A nextNode
B curr->next = prev
C prev = curr
D curr = nextNode
1 call solu
→ Rec

Middle of a Linked List







1 cn → $O(N)$

mid → $O(\frac{N}{2})$

$O(N) + O(\frac{N}{2}) \rightarrow O(N)$

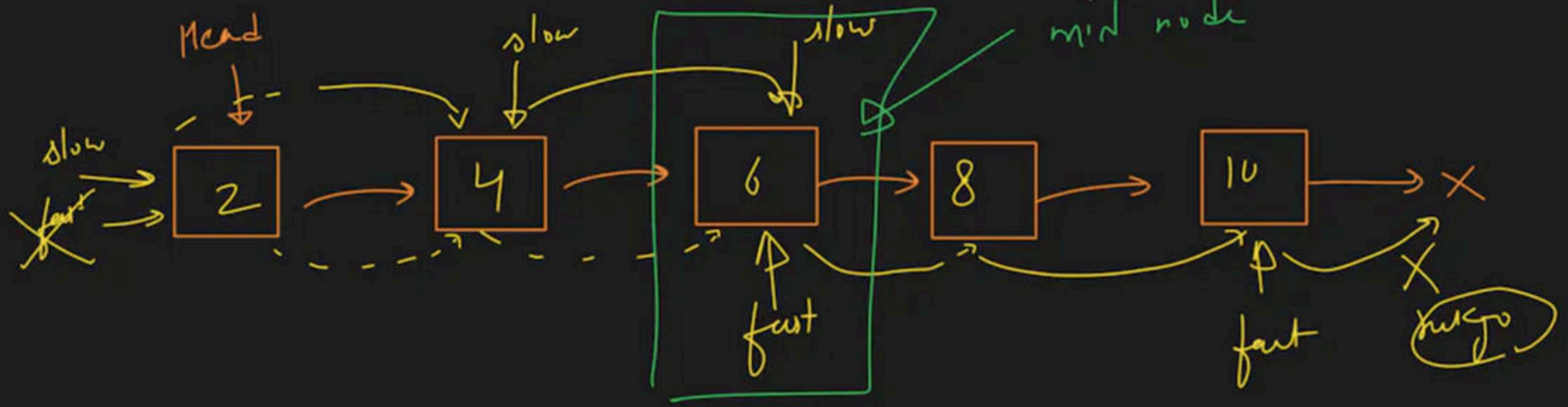
Single Pass

$\Rightarrow$

Slow & fast

Slow → 1 step

fast → 2 step

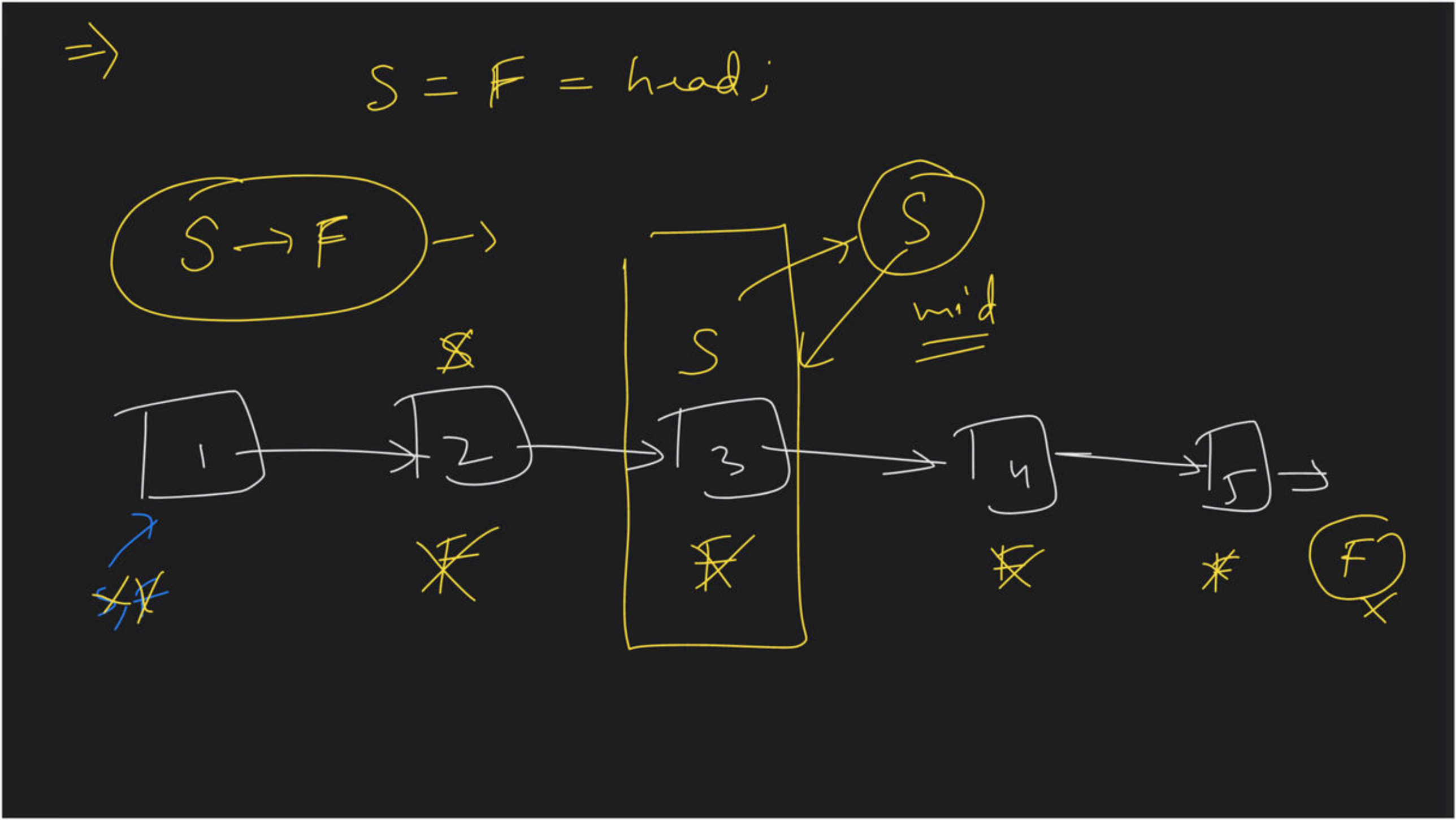


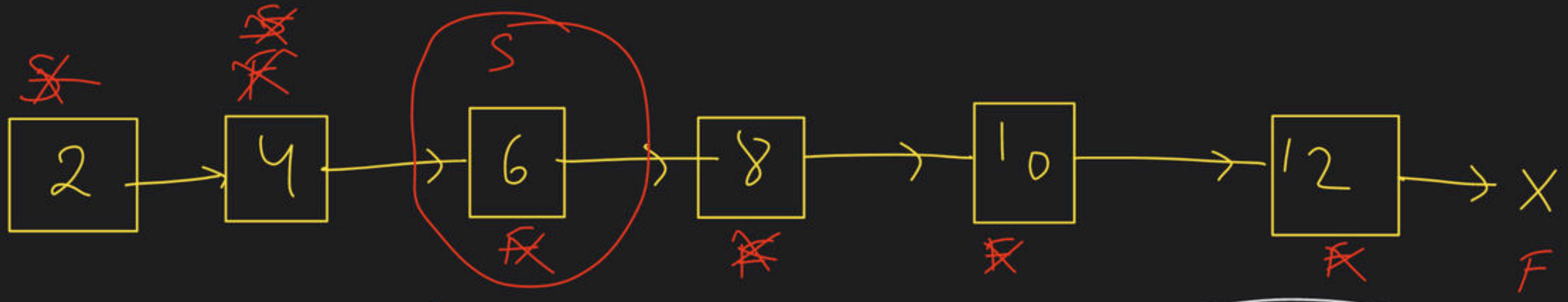
① slow/fast → init

Tortoise

slow-fast (SF)

2 ptr approach


S = F = head;
S → F
S
mid
S
1
2
3
4
5
F



1

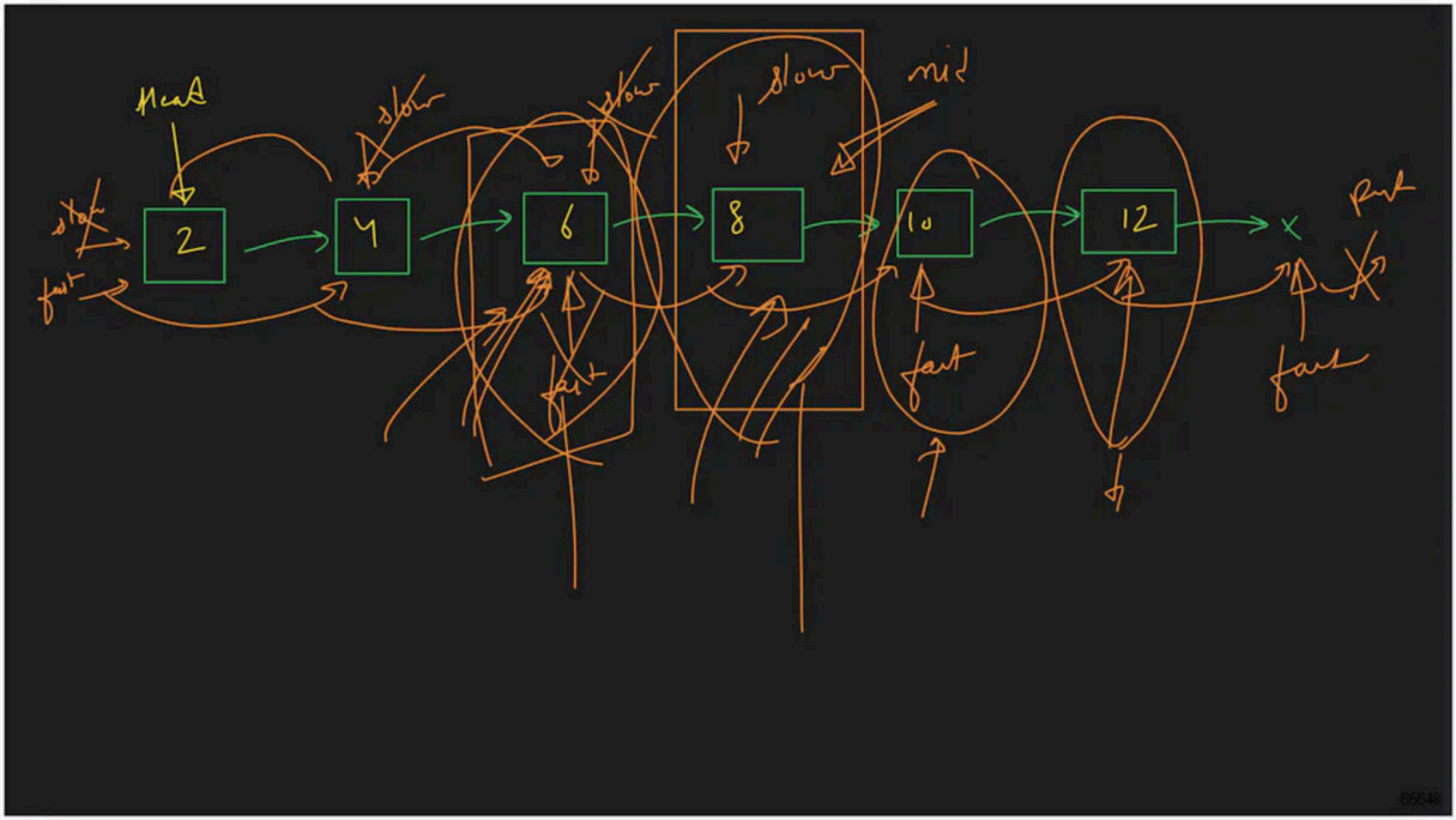
```
while ( fast )
{  fast = fast -> next;
   if ( fast )
     {  fa = fast -> next;
        S = S -> next;
     }
}
```

6

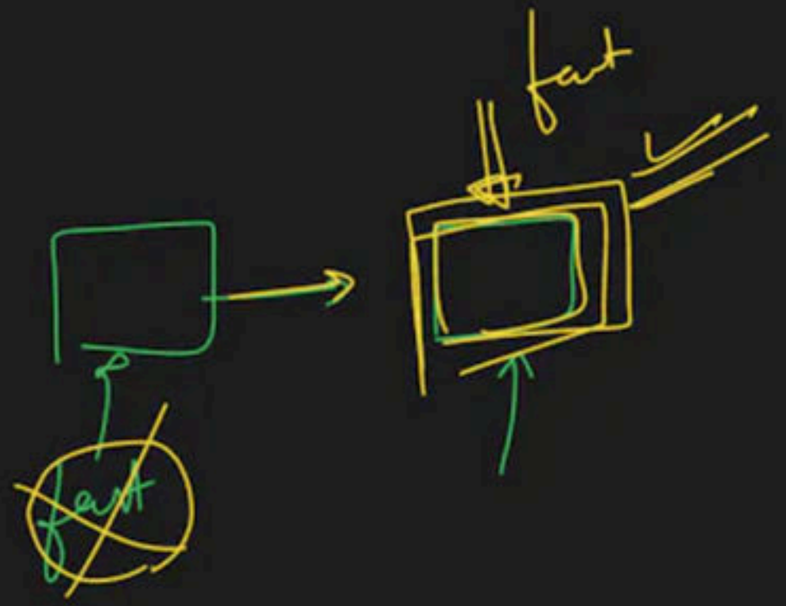
```
S = head;
F = head -> next;
```

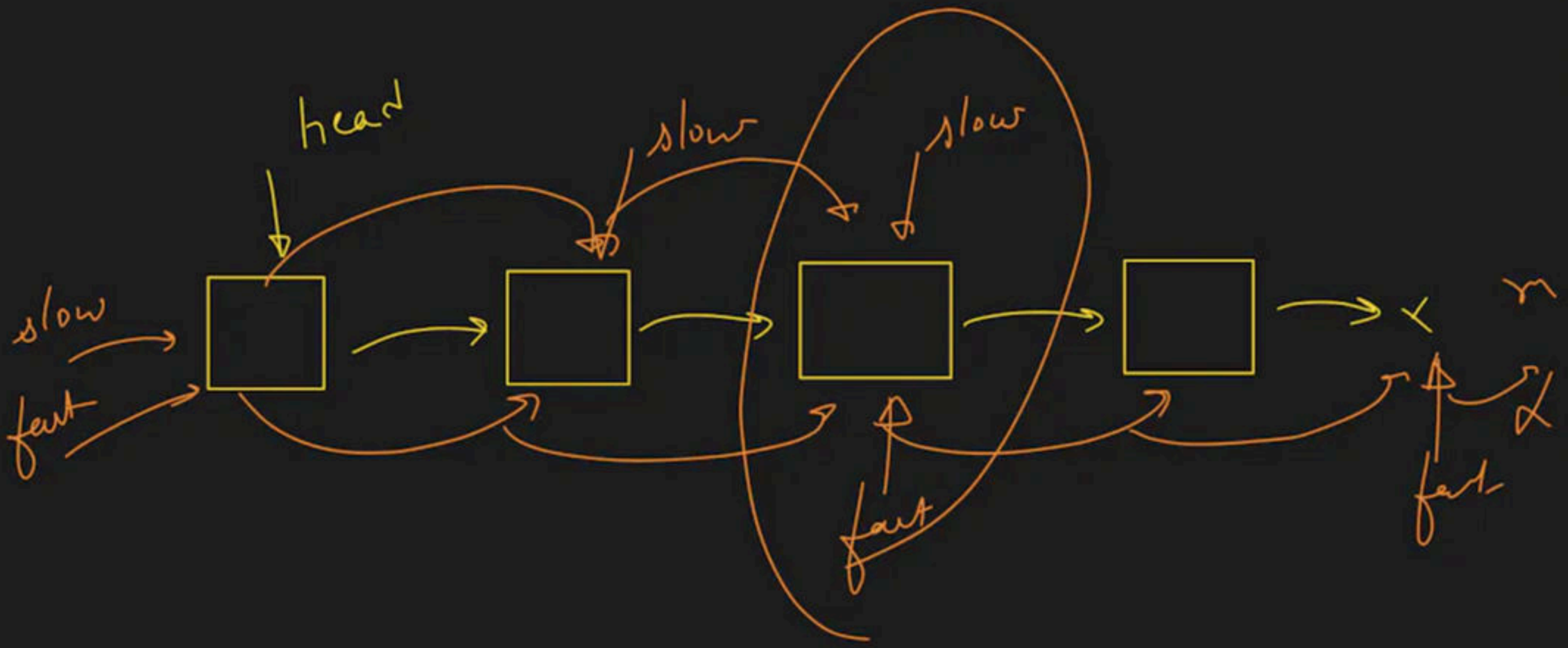
8

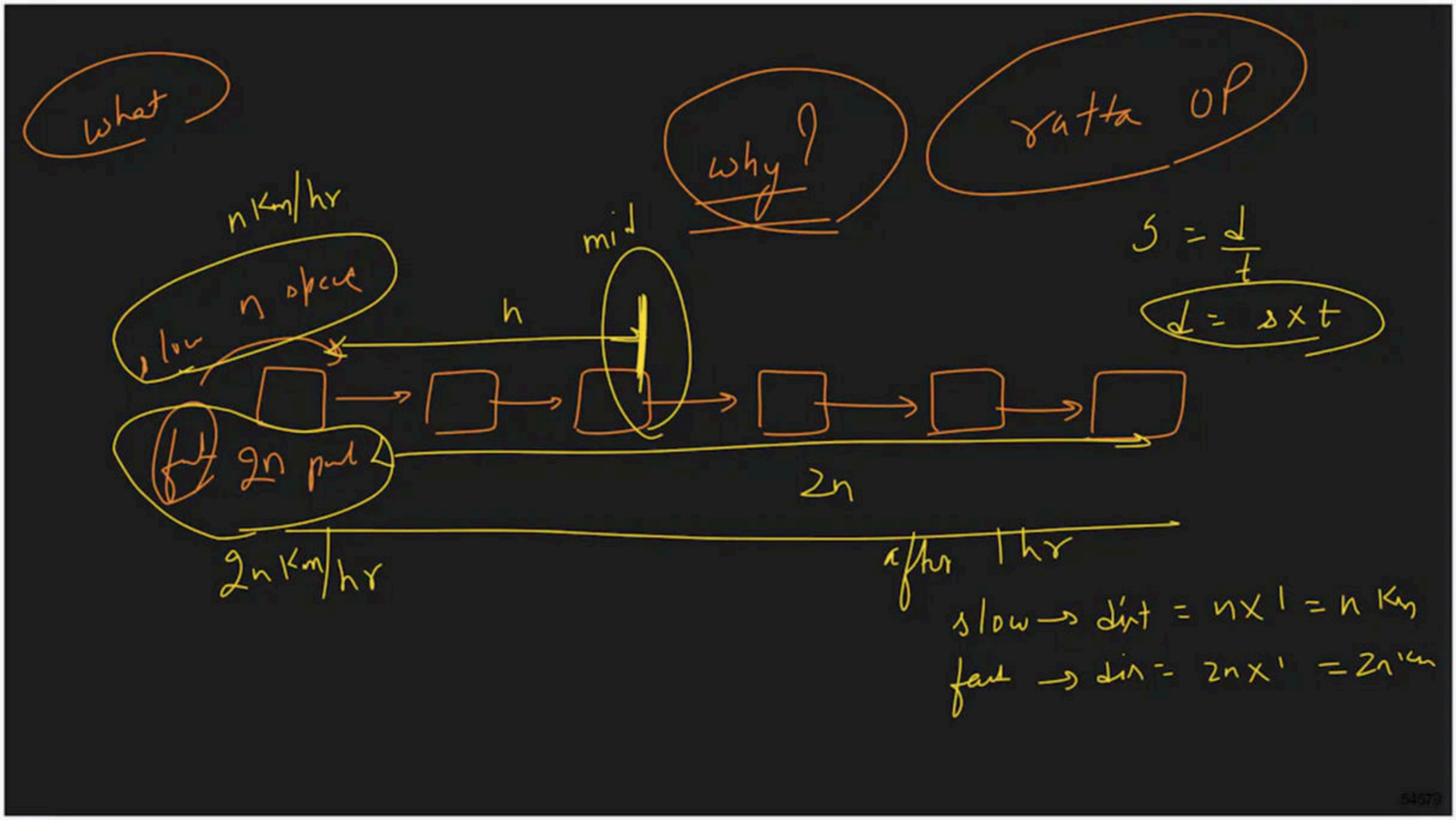
```
S = F = head
```


head
slow
slow
slow
mid
2
4
6
8
10
12
x
fast
fast
fast
fast
fast


mid node
if (head == NULL)
return;
Node * slow = head
Node * fast = head;
while (fast -> next != NULL)
fast = fast -> next;
if (fast -> next != NULL)
fast = fast -> next
slow = slow -> next
return slow;
LL is not empty
mid
Node * slow = head
Node * fast = head
while (fast != NULL)
fast = fast -> next;
if (fast != NULL)
fast = fast -> next;
slow = slow -> next;
return slow;


fault
fault

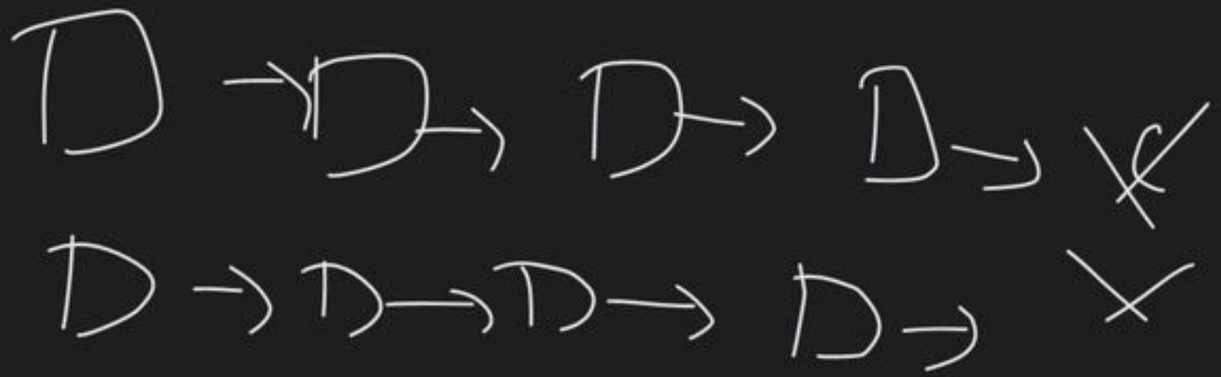

head
slow
slow
slow
fast
fast
fast
x
n
2


what
why ?
ratta OP
n Kms/hr
slow
n space
mid
h
fast
2n path
2n
2n Kms/hr
after 1 hr
S = d/t
d = s x t
slow → dist = n x 1 = n Kms
fast → dist = 2n x 1 = 2n Kms

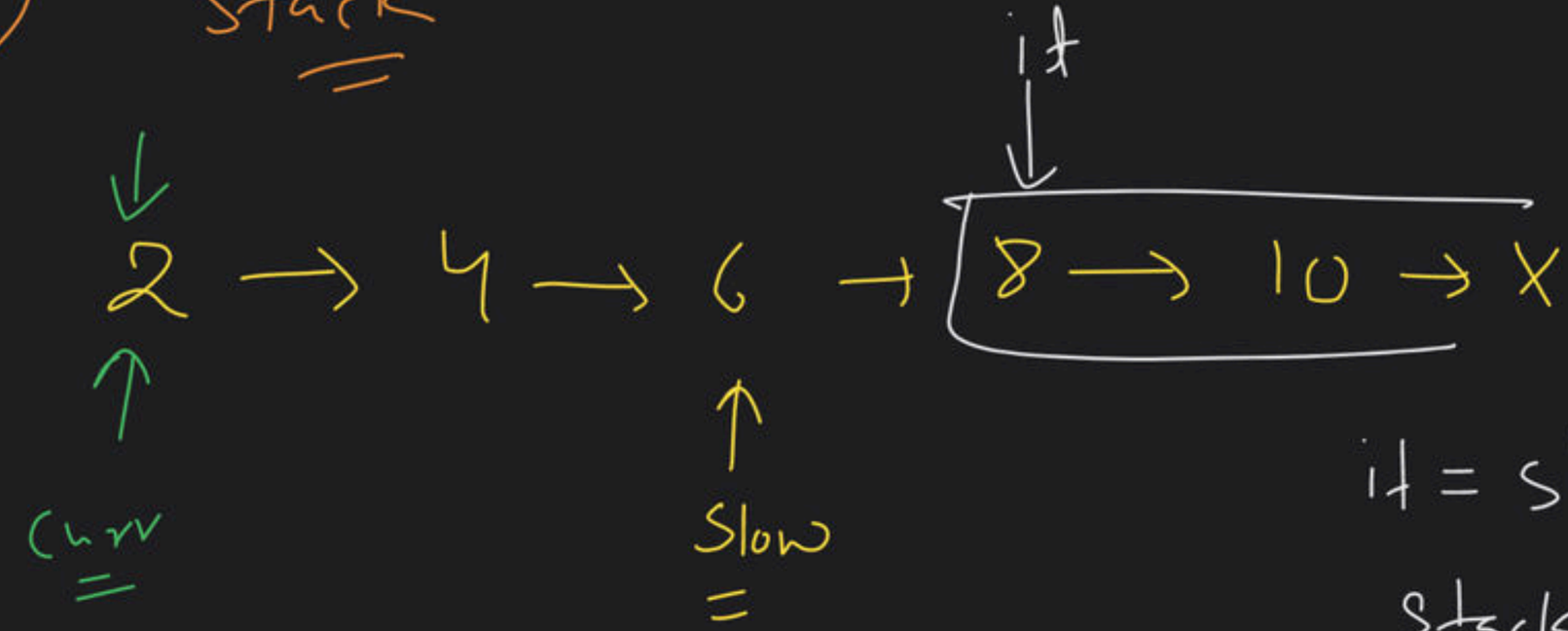
1. Newly Reversed LL Banao = Compare ✗ Mem→

2. (1) Break into half
   = (2) Second ko reverse kro
   (3) Compare

3.

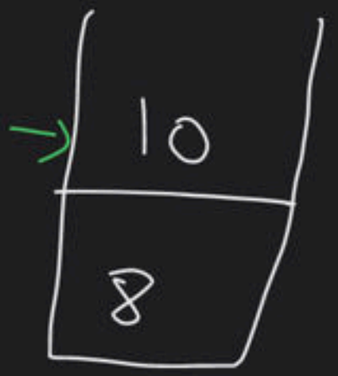


(1) Stack



it = slow->next;

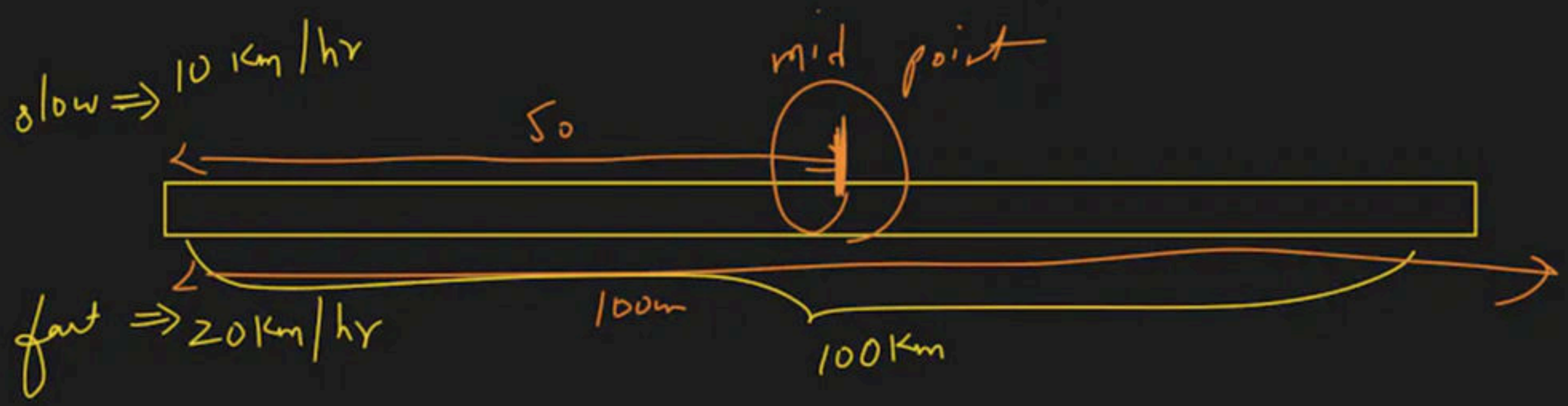
stack<int> s;



after 5 hr

slow → dist = 10 × 5 = 50 km

fast = 20 × 5 = 100 km

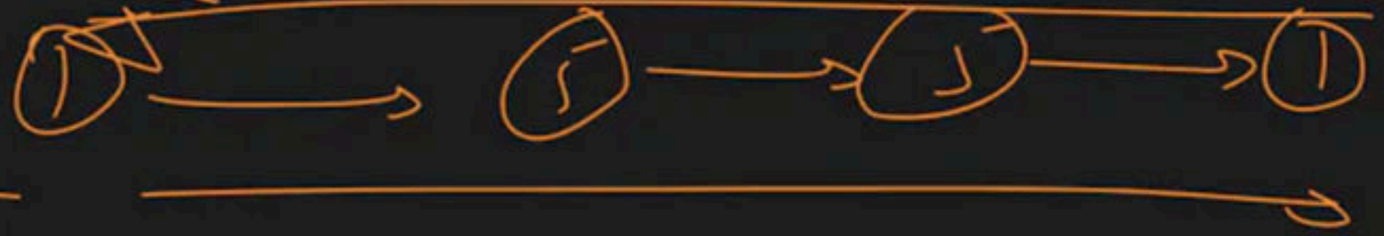


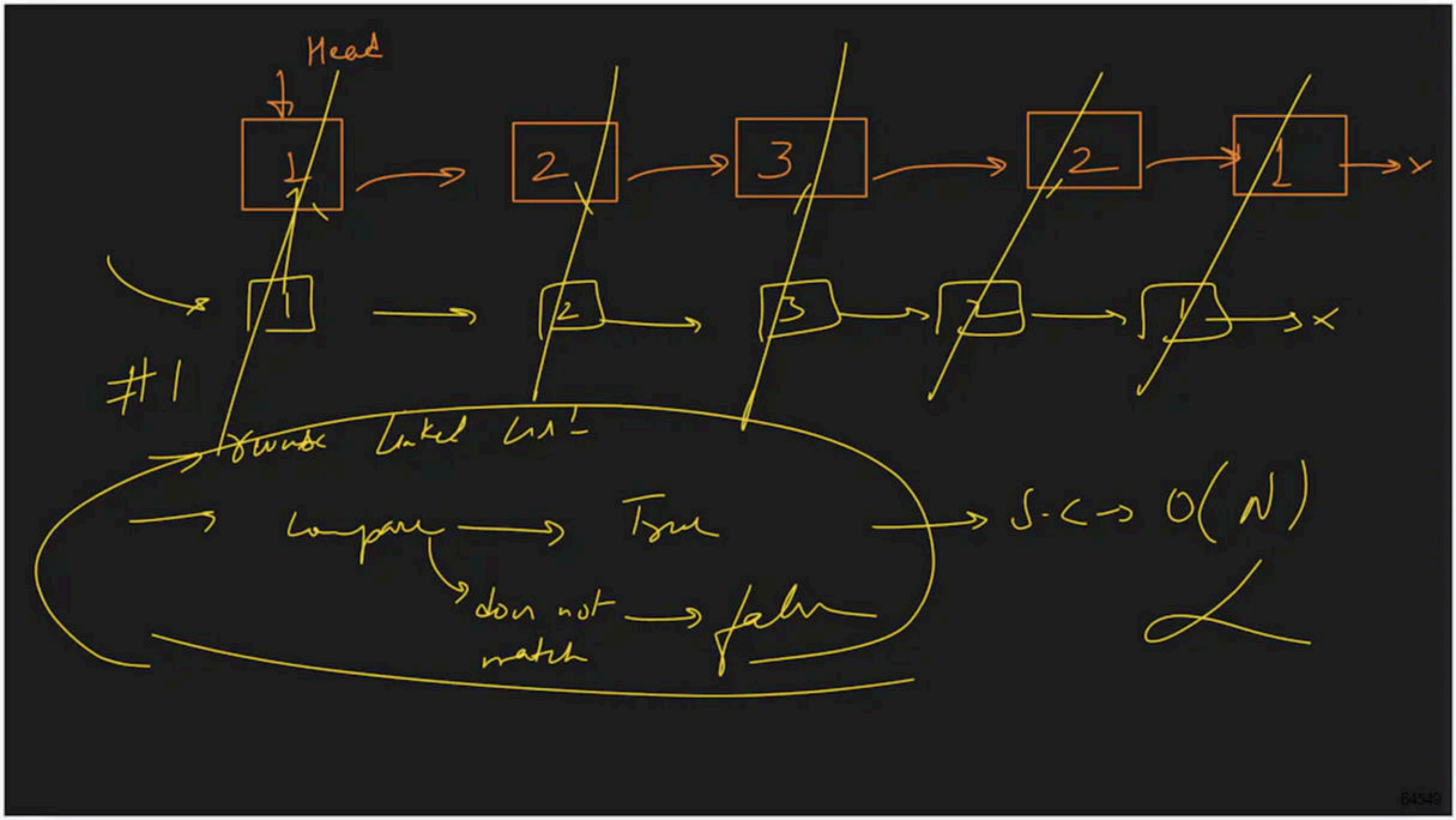
③ Palindrome

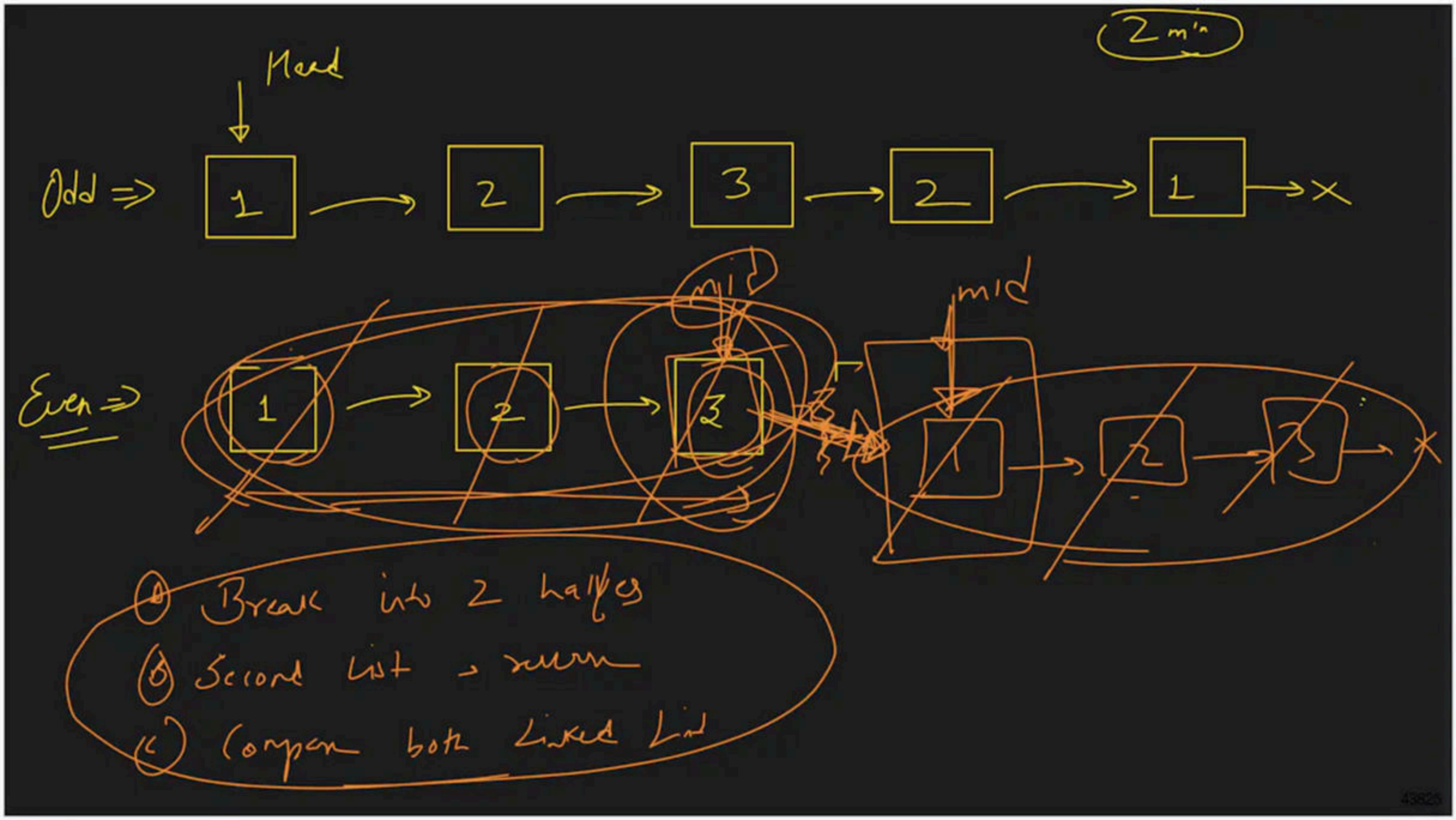
babbab

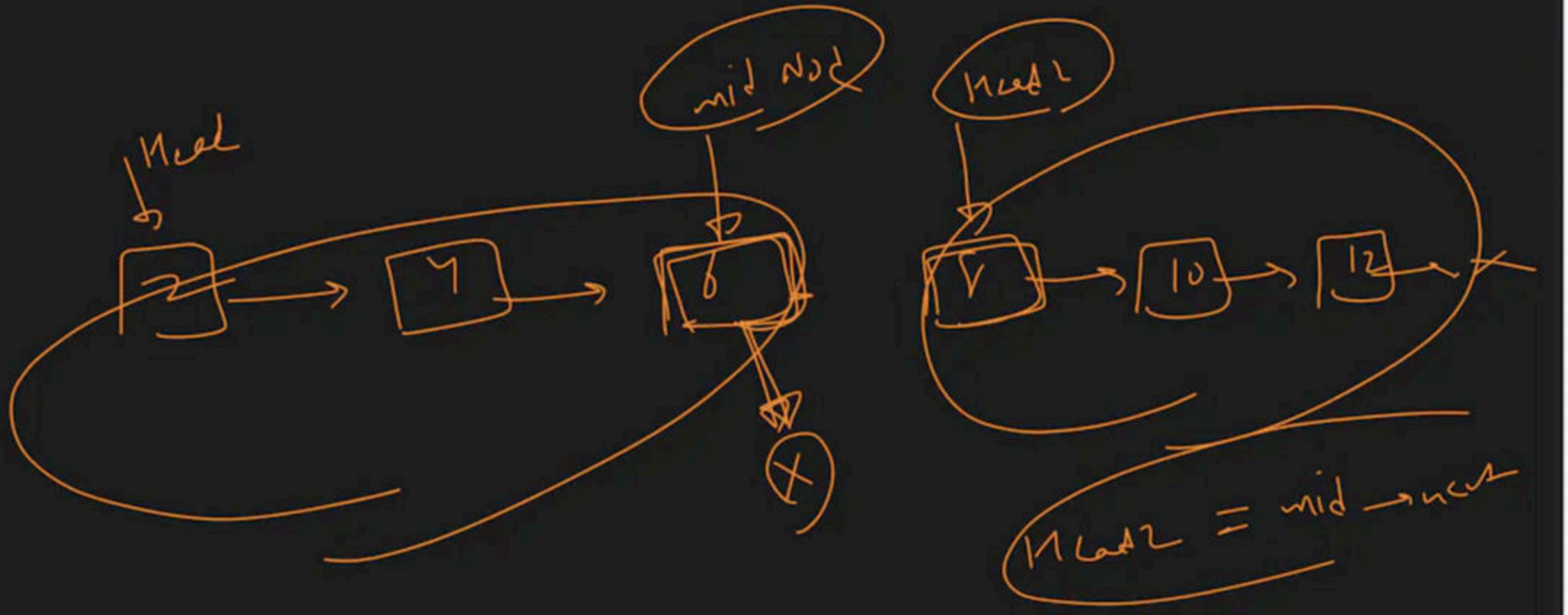
RACECAR

NITIN

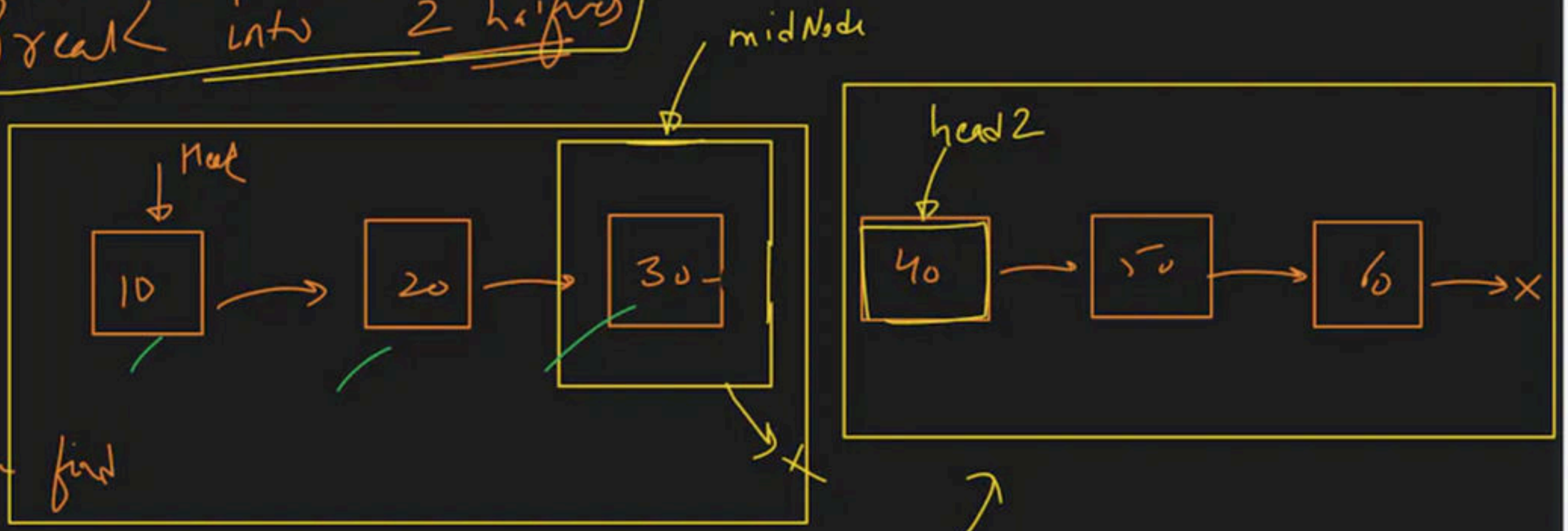



Head
1
2
3
2
1
x
1
2
3
x
#1
Traverse Linked List
Compare
True
False
do not match
O(N)


2 min
Head
Odd =>
1
2
3
2
1
X
Even =>
1
2
3
mid
mid
1
2
3
X
① Break into 2 halves
② Second list - reverse
③ Compare both Linked List


mid Node
Head
Head
2
6
6
7
10
12
X
Head = mid → next

(A) Break into 2 halves



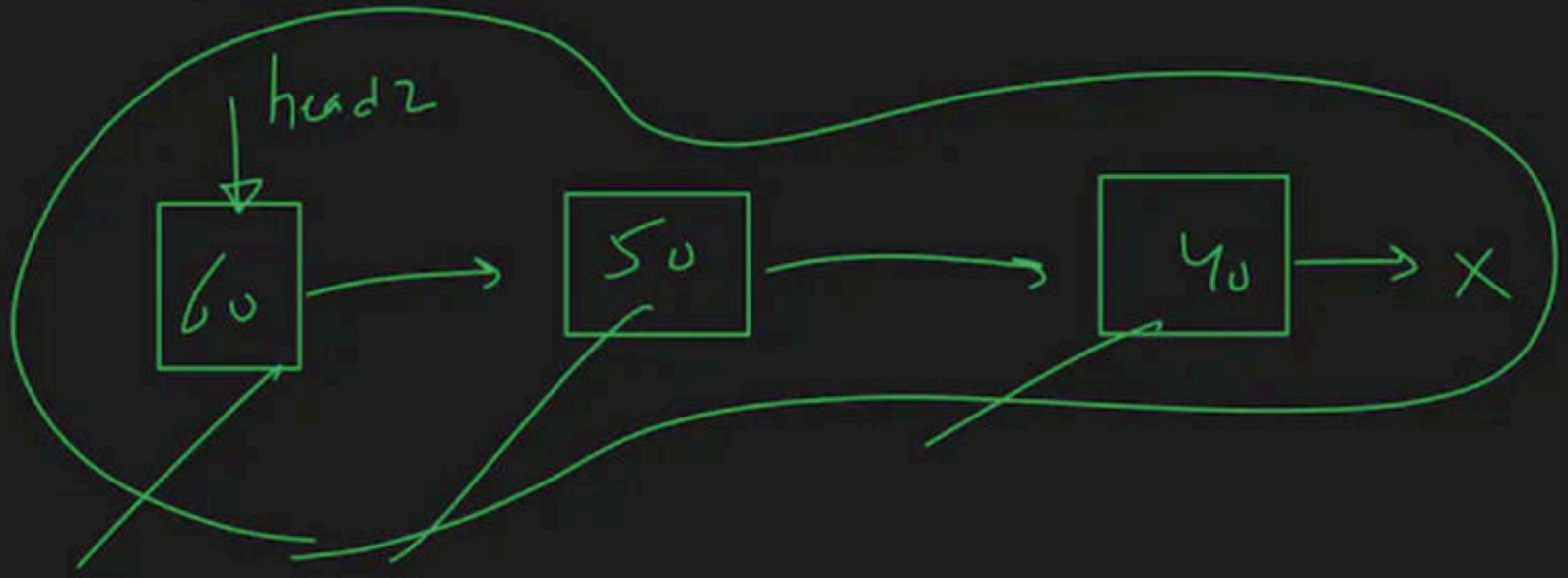
① midNode find

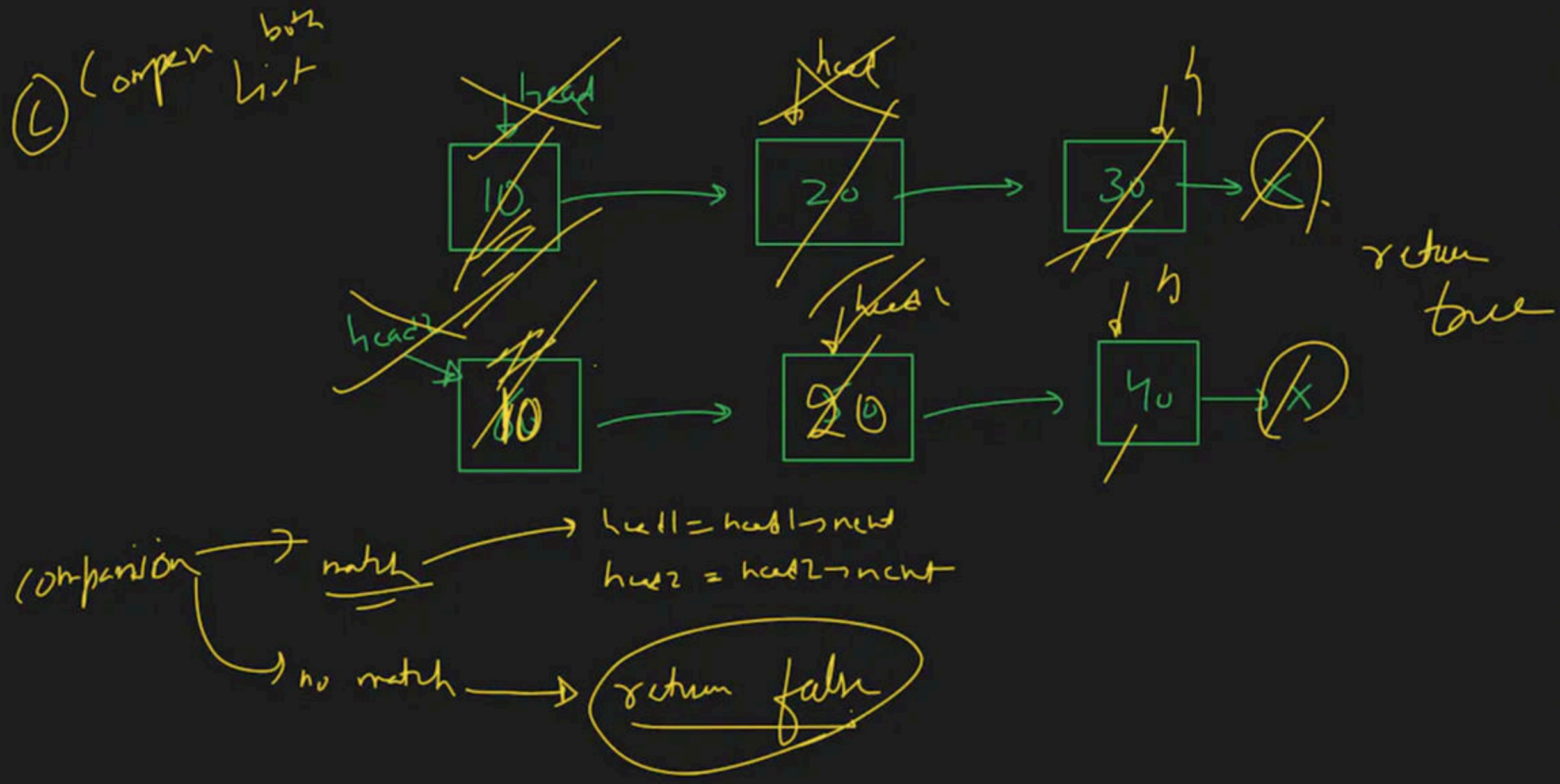
② head2 set

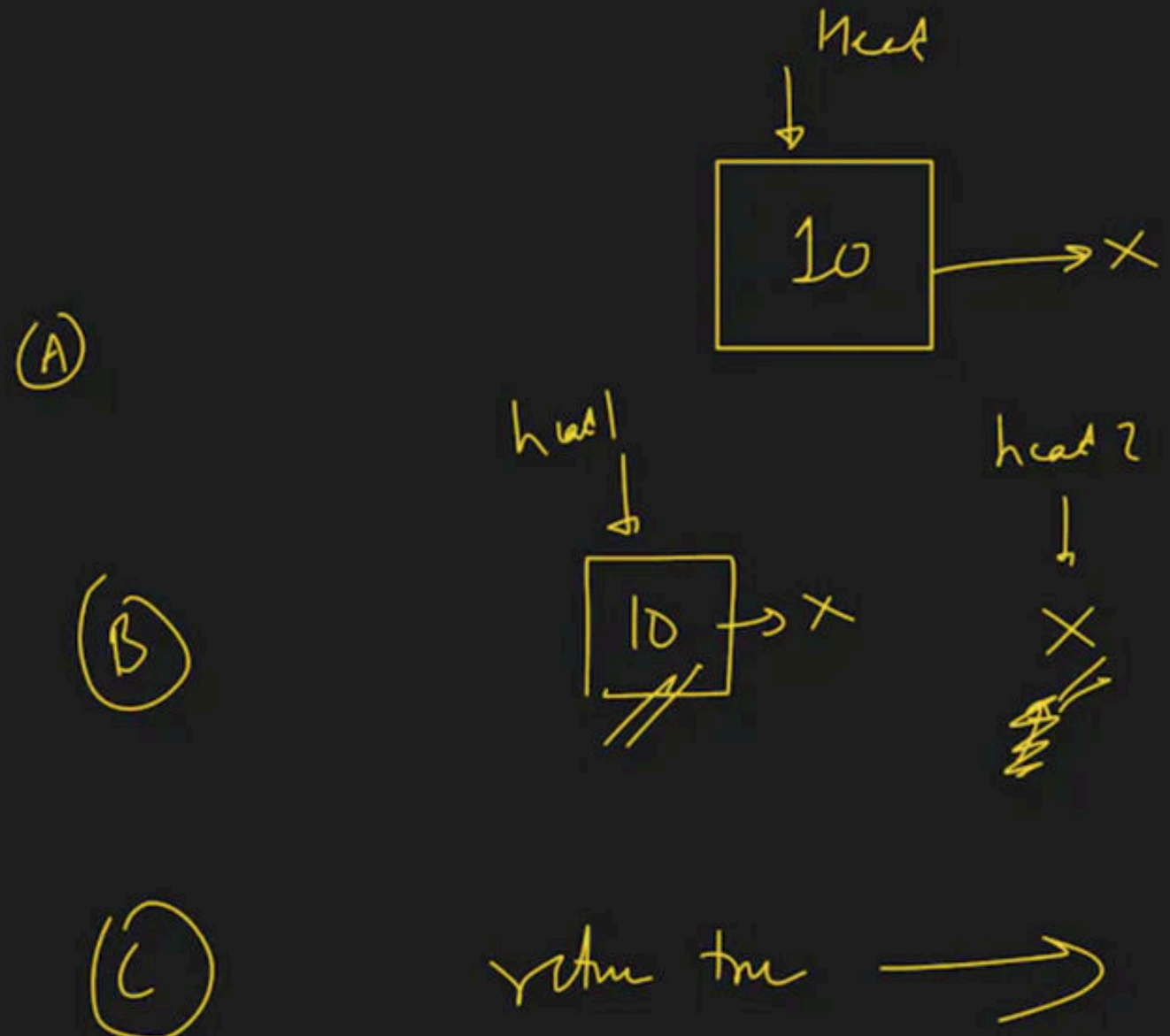
③ midNode → next = NULL

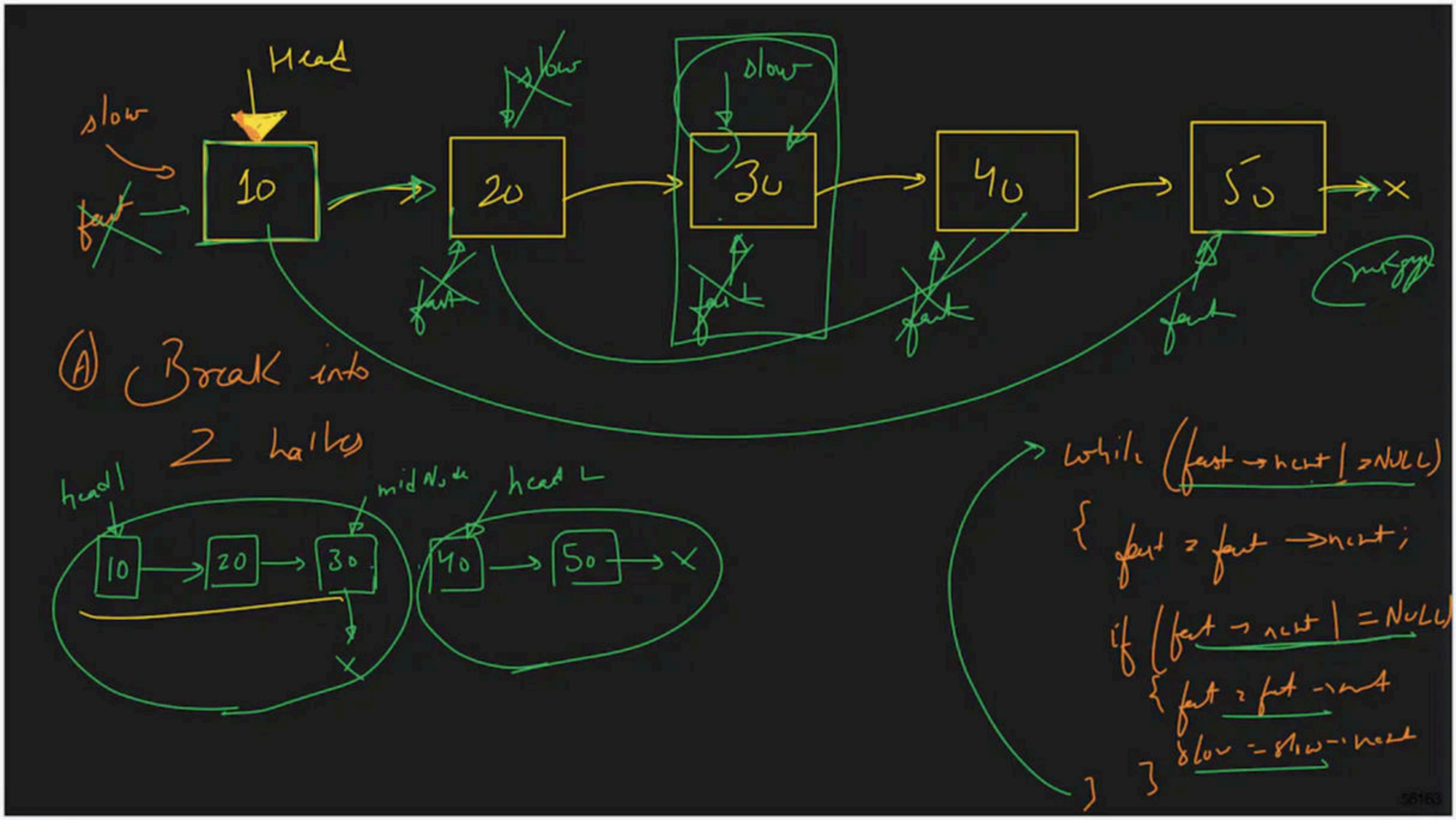
(B) Recur Second half

Ans

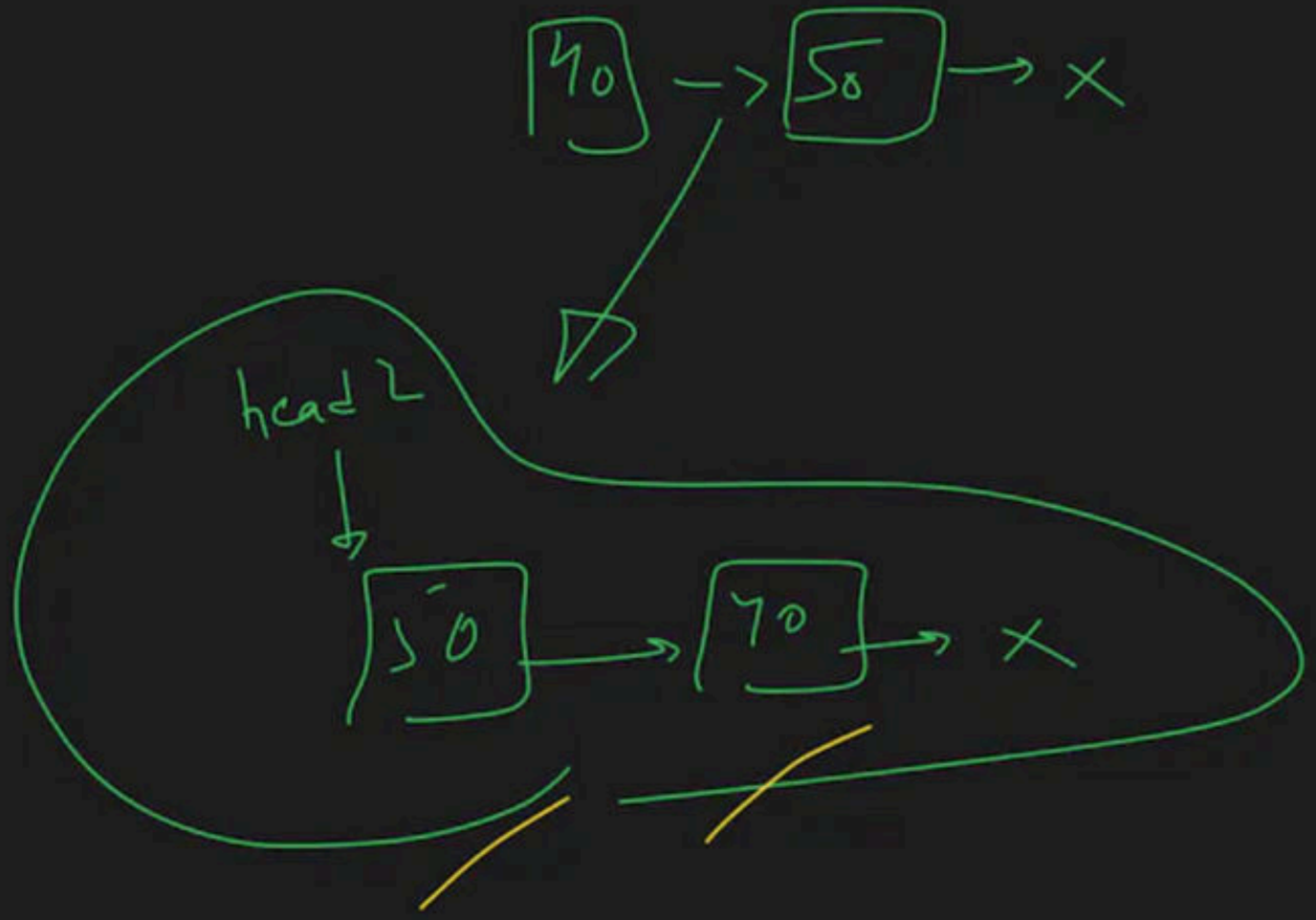


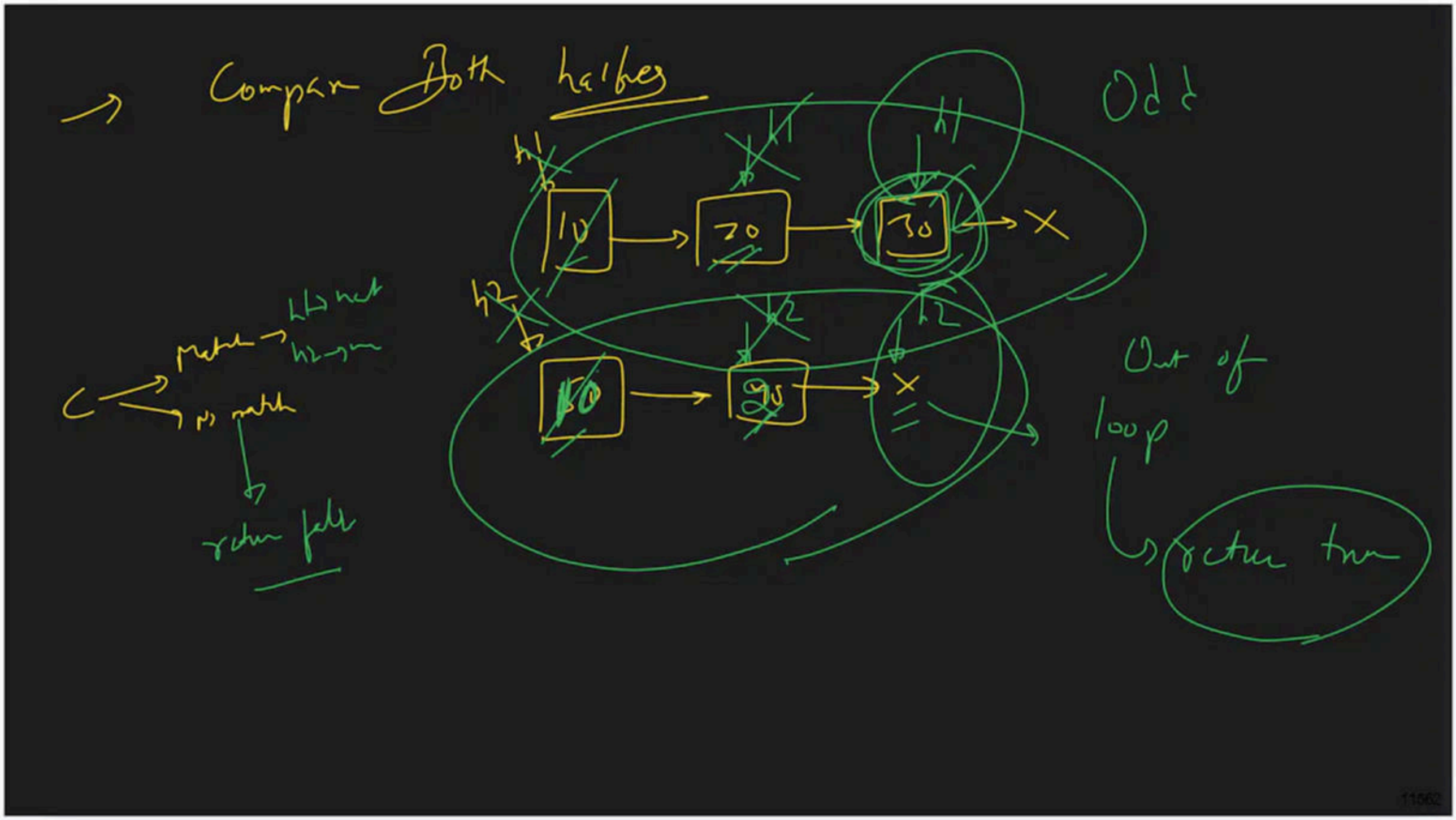

1) Compare both list
head
10
20
30
head2
10
20
40
return true
Comparison
match
ll = head1->next
head2 = head2->next
no match
return false


Head
IO
X
(A)
head
IO
X
head 2
X
(B)
(C)


Head
slow
fast
slow
slow
10
20
30
40
50
NULL
(A) Break into
2 halves
head
mid Node
head
10
20
30
40
50
while (fast->next != NULL)
{ fast = fast->next;
if (fast->next != NULL)
{ fast = fast->next
slow = slow->next
} }

(3) ~~first~~ Reverse 2nd half




Compare Both halves
Odd
10
20
30
X
40
50
X
Out of loop
return true
return false

(A) Break - mid → O(N)

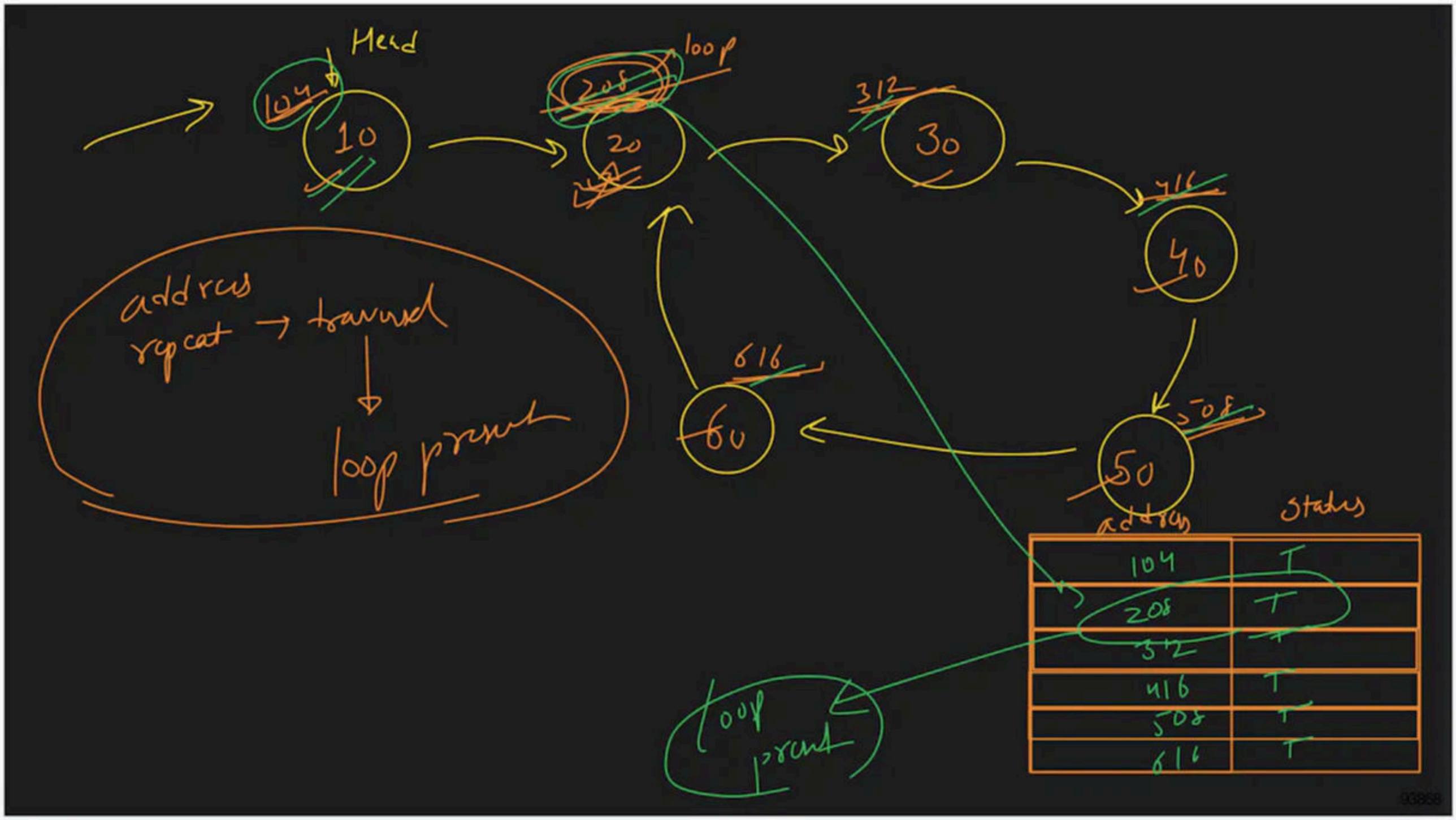
(B) Recur → O(N)

(C) Loop → O(N)

O(N)

Check cycle in a Linked List

1. Map


Head
10
20
30
40
50
60
loop
address repeat → traversed
loop present
loop present
Status
104
208
312
416
508
616
T

Tomorrow

↳ Monday → 9pm-1 Lakshay

Mentorship

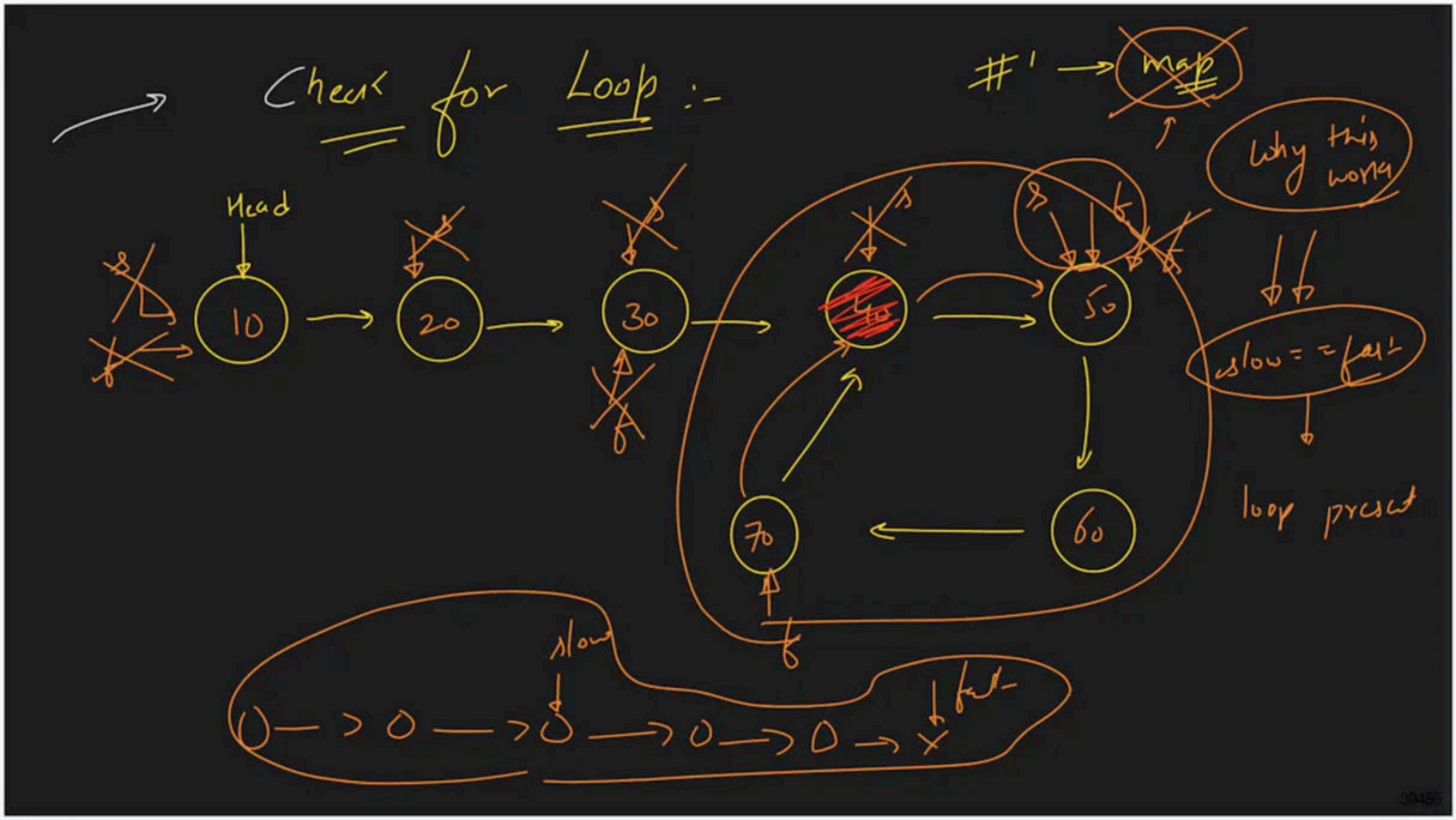
↳ Tuesday → 9pm → Love

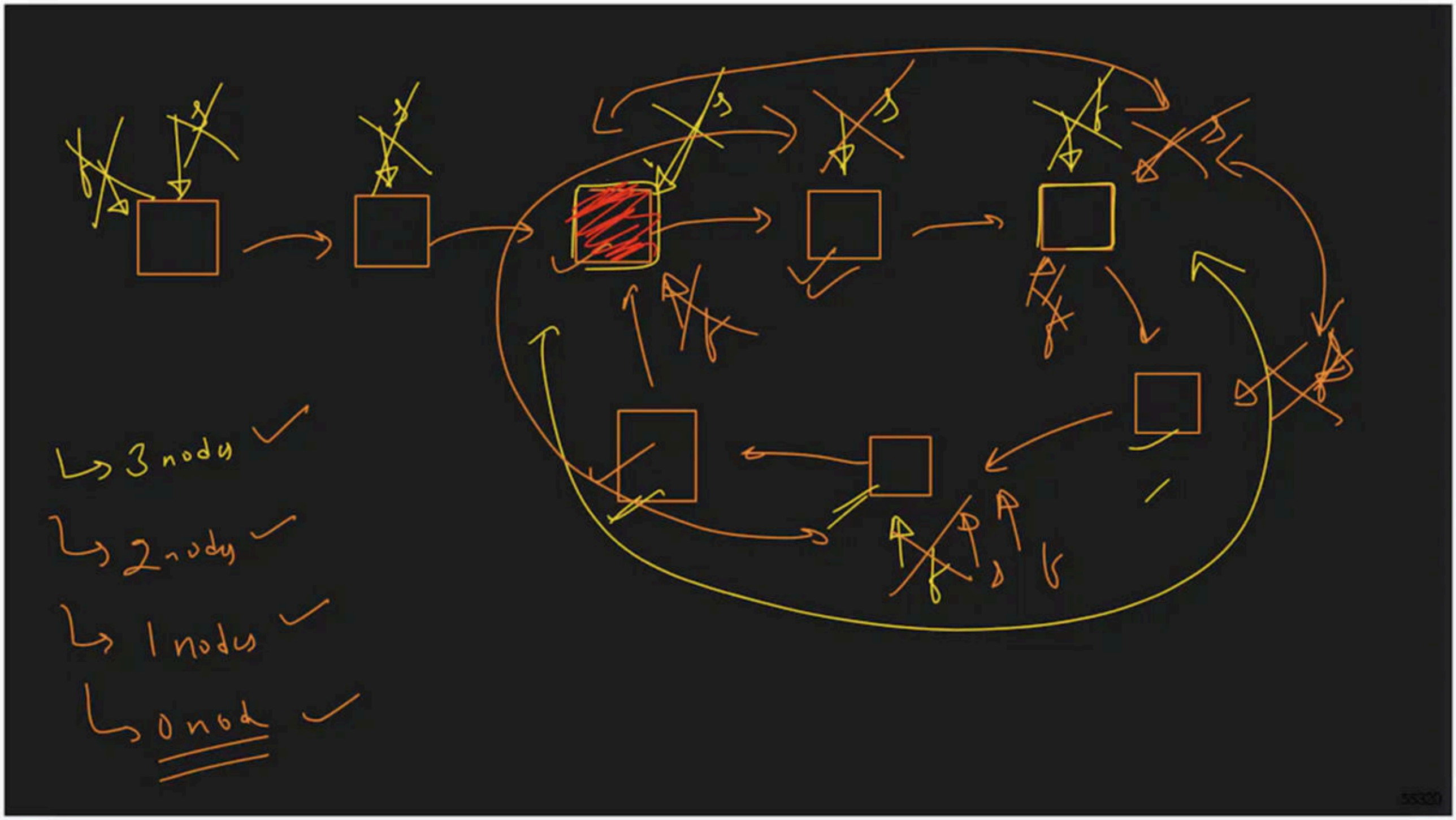
L-L → Class

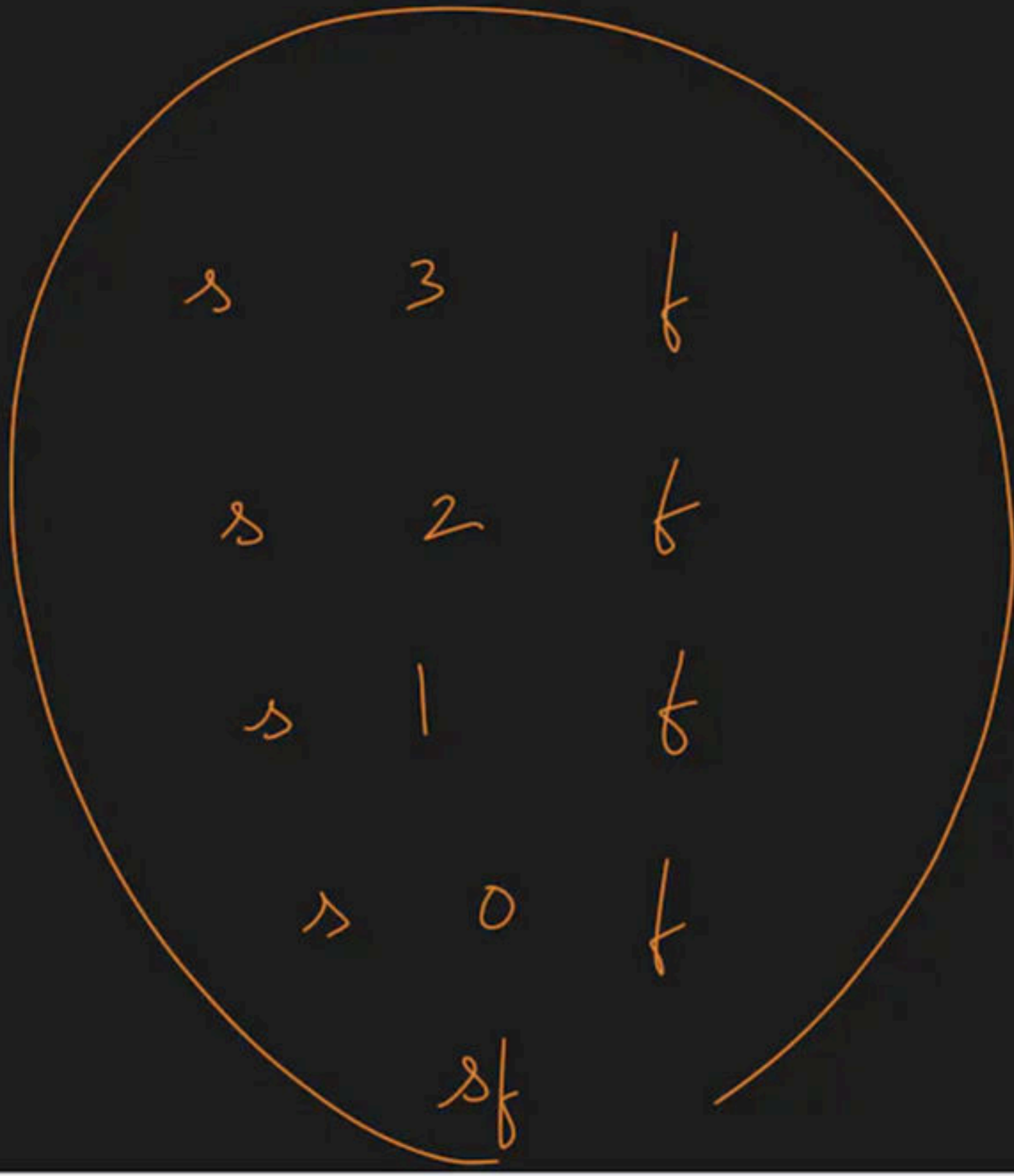
Linked List - Class 4

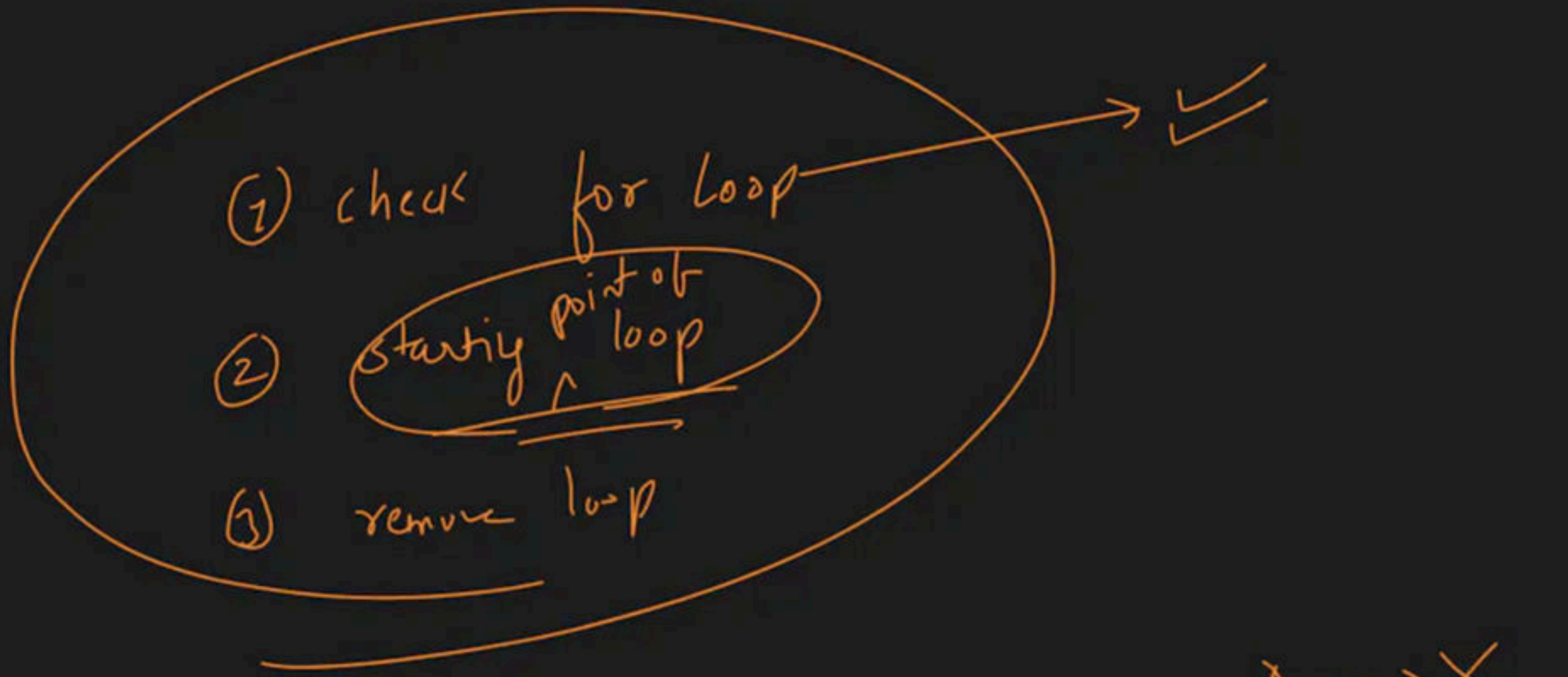
Special class

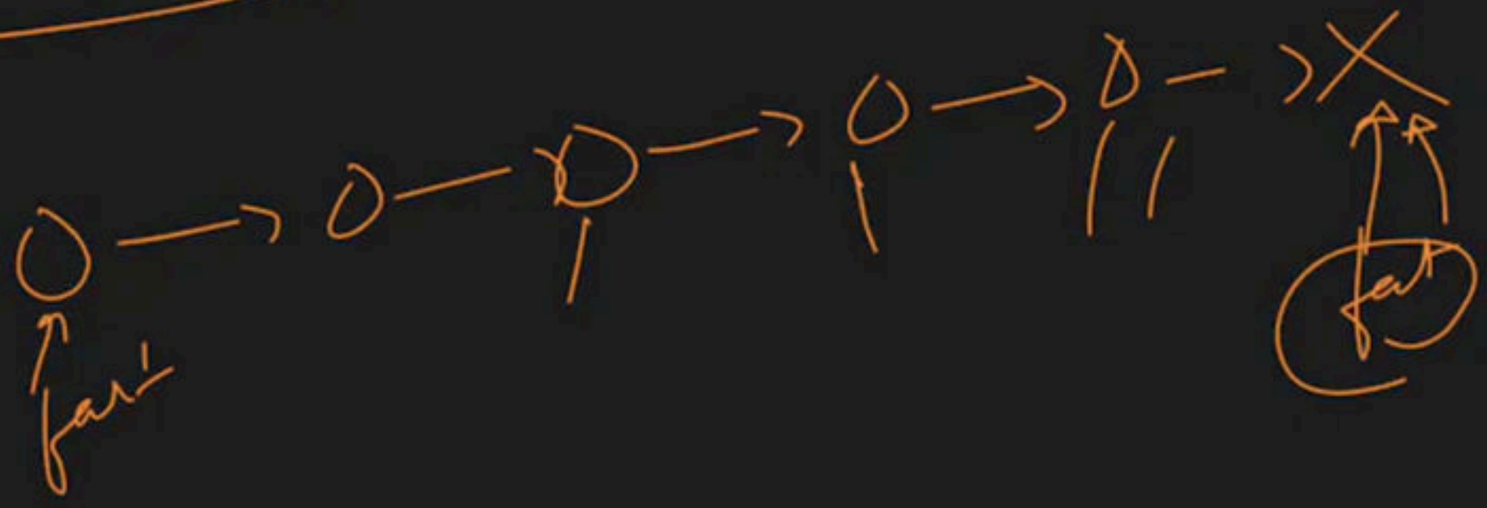
Love Babbar • Nov 6, 2023


Check for Loop :-
#1 → map
Why this work
Head
10
20
30
40
50
60
70
slow
fast
loop present


↳ 3 nodes
↳ 2 nodes
↳ 1 node
↳ 0 node

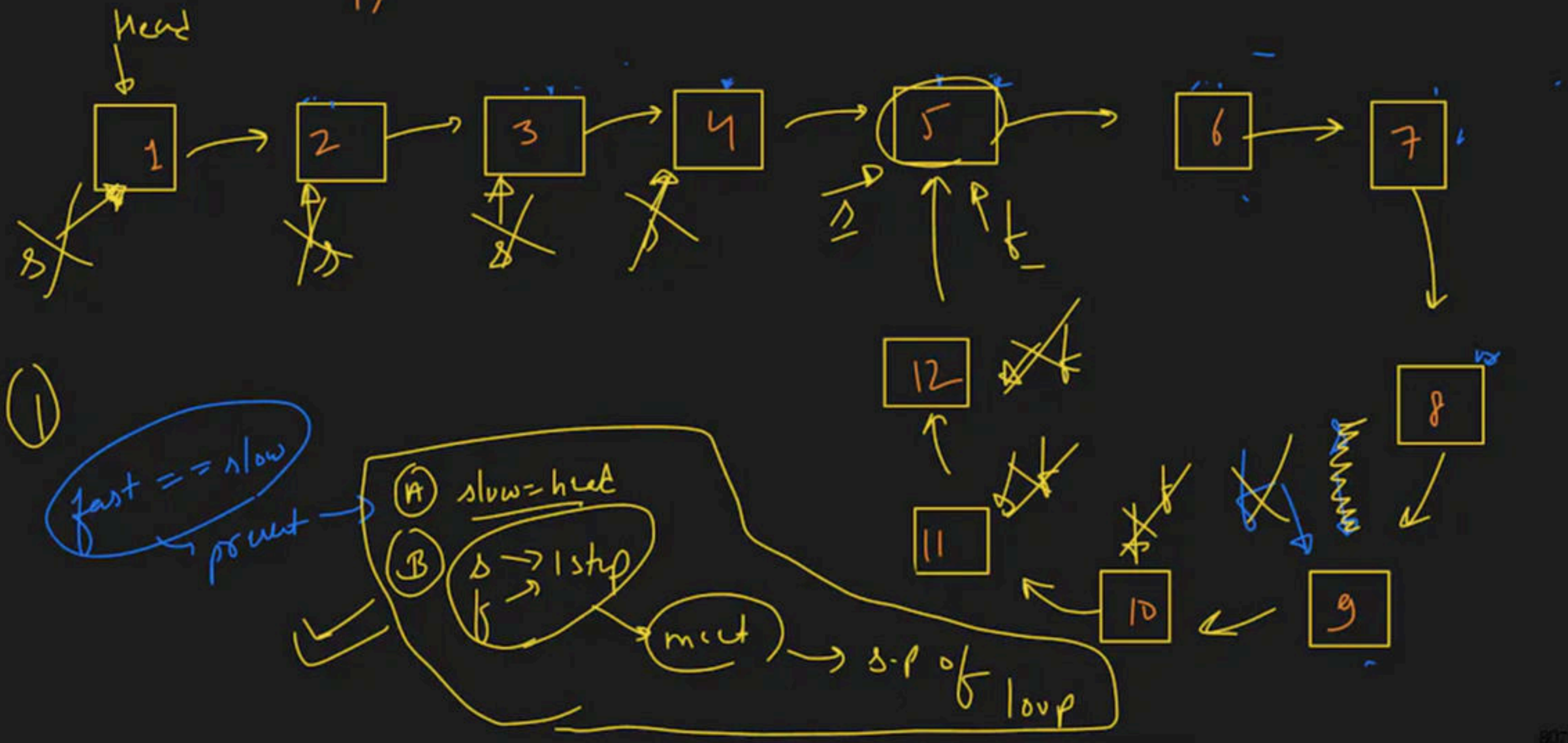



(1) check for loop
(2) starting point of loop
(3) remove loop


fast
fast

→ Starting Point of Loop :-

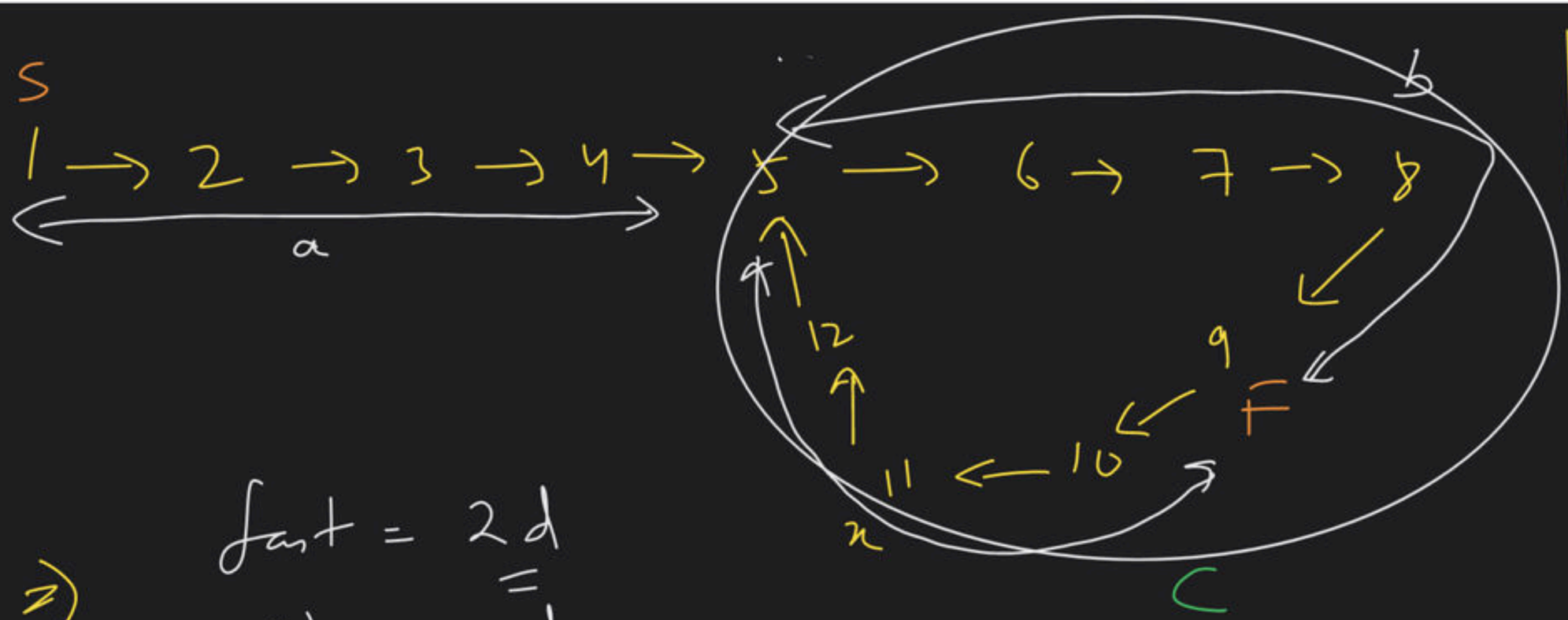
S.P of loop → ?



① Loop detect ⇒ S == F

② S = head.

③ while (S != F)
{ S++, F++;
}



$a+b = kC$

$\Rightarrow$

$\Rightarrow$ $fast = 2d$

$slow = d$

$\Rightarrow$ $D_f = 2 * D_s$

$\Rightarrow$ $a + k_1 C + b = 2(a + k_2 C + b)$

$\Rightarrow$ $a + k_1 C + b = 2a + 2k_2 C + 2b$

$(k_1 - k_2) C = a + b$

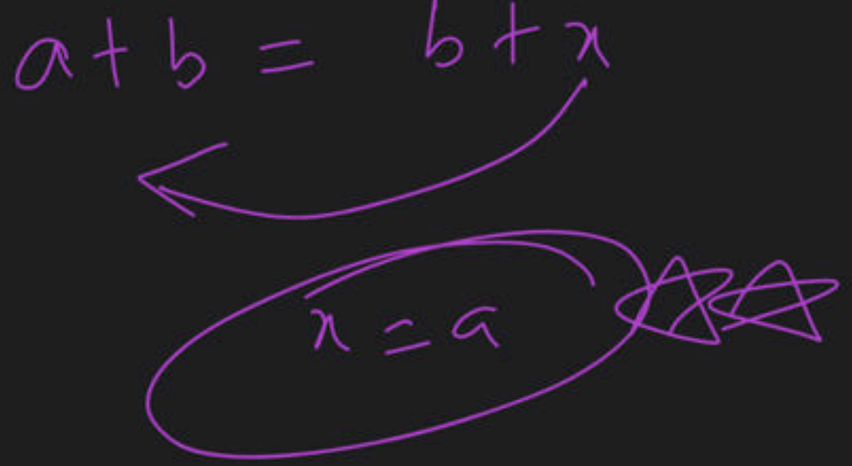
$$a+b = kc$$

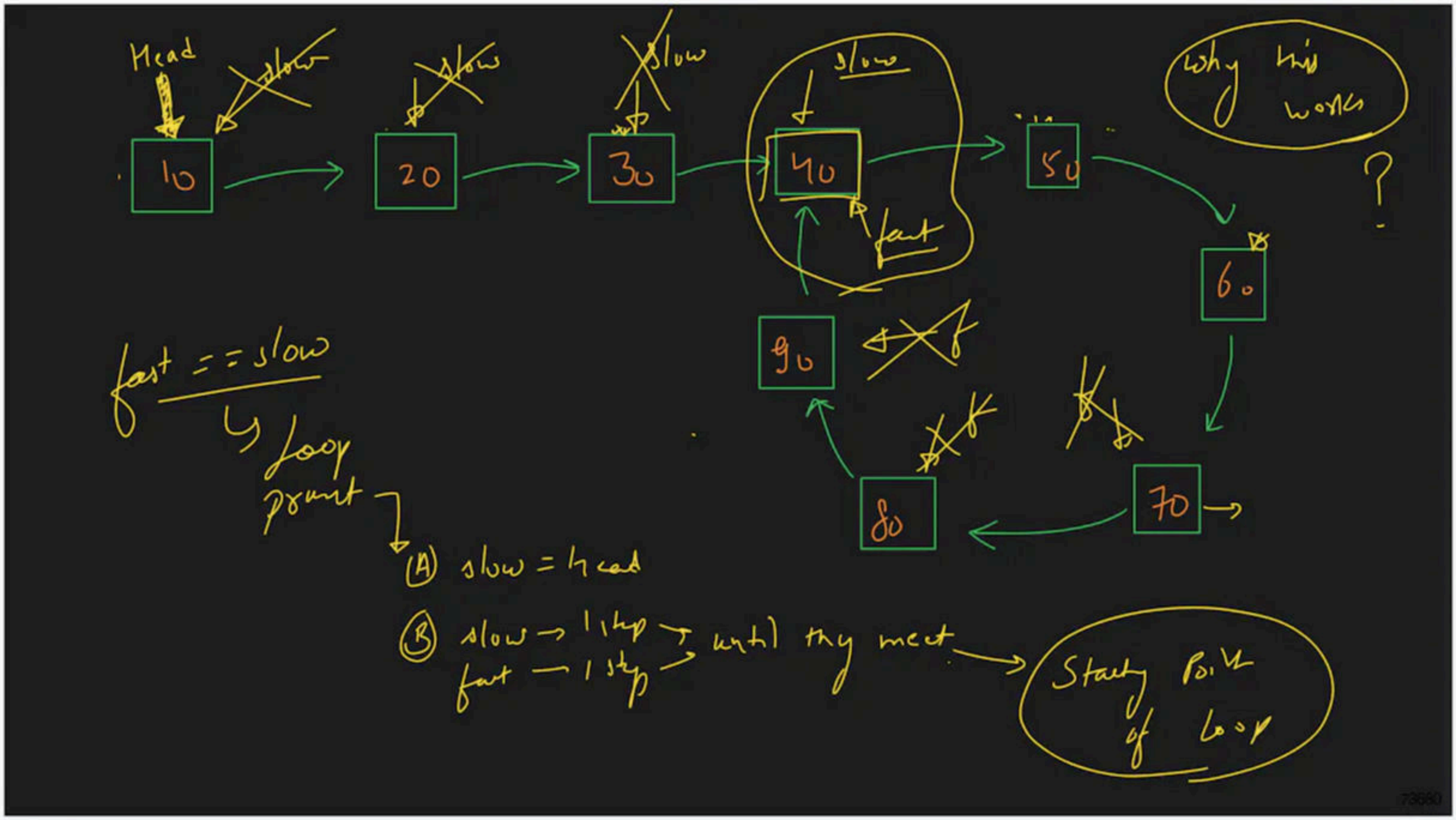
$$\Rightarrow c = b+x$$

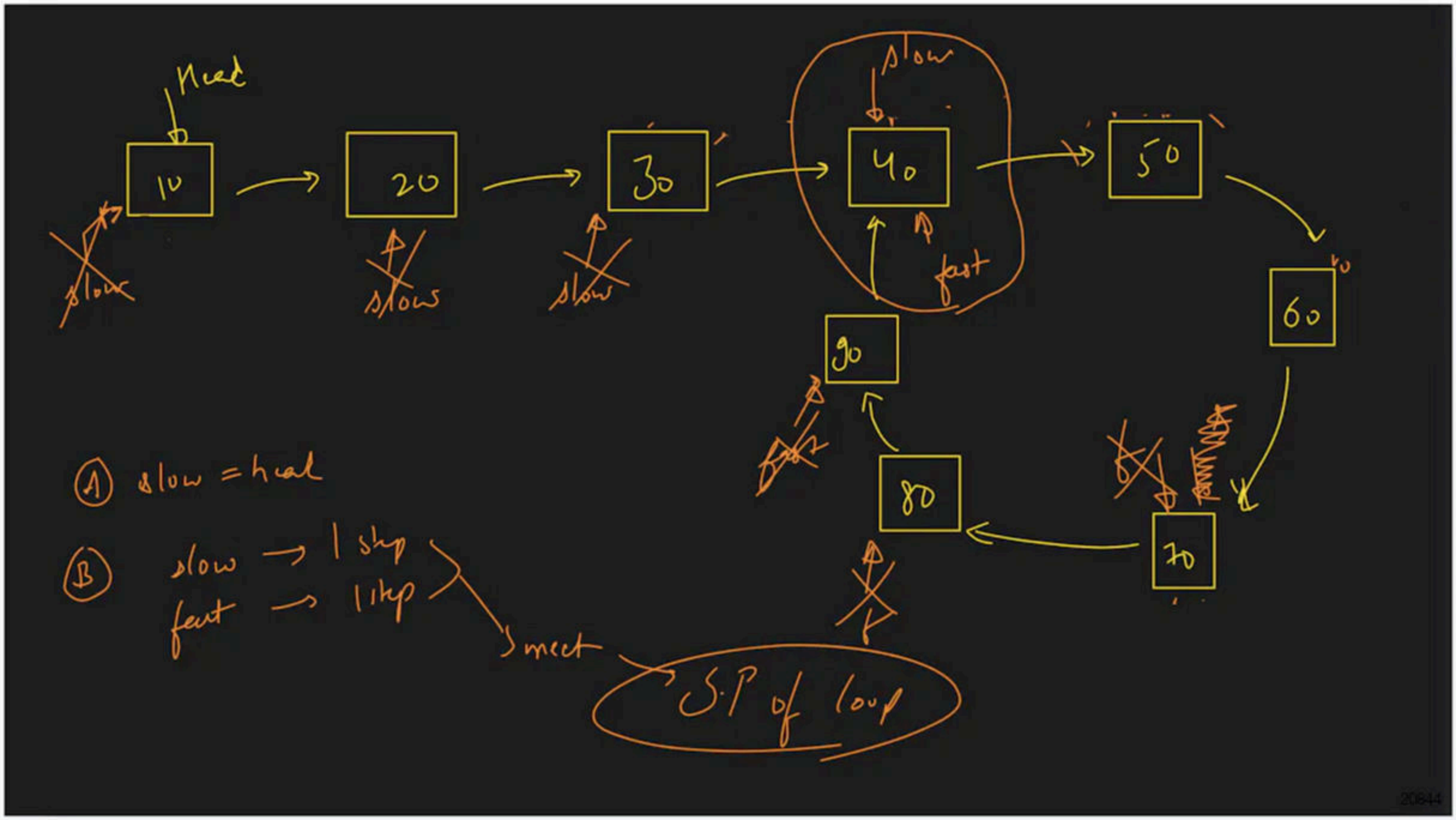
$$\Rightarrow a+b = c \quad \therefore$$

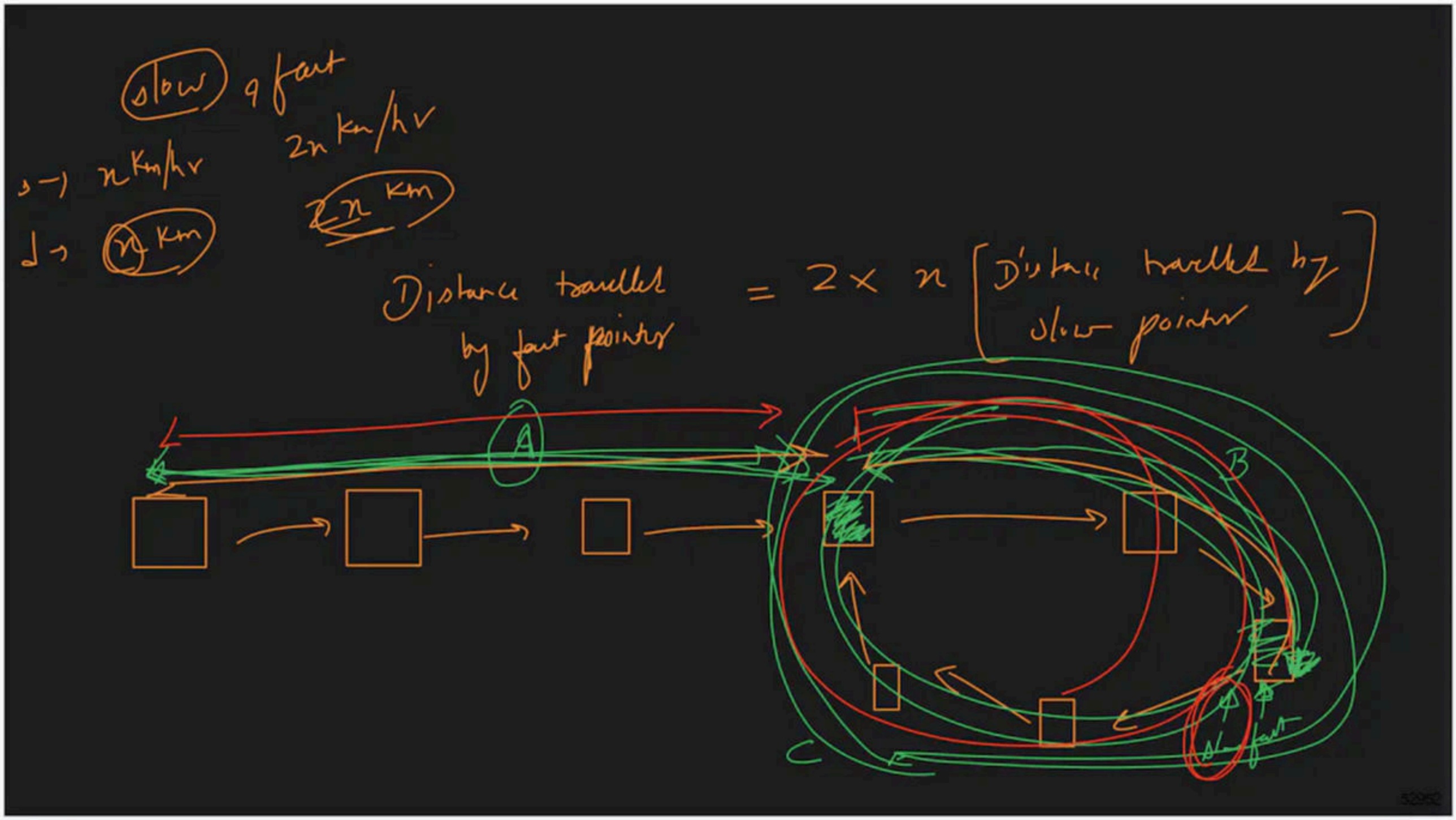
$a+b=c \Rightarrow a+\cancel{b} = \cancel{b}+x$

$x=a$


a + b = b + x
x = a


Head
slow
slow
slow
slow
10
20
30
40
50
60
70
80
90
fast
why this works ?
fast == slow
↳ loop present
(A) slow = head
(B) slow → 1 step
fast → 1 step
until they meet
Starting Point of Loop


Head
10
20
30
40
50
60
70
80
90
slow
slow
slow
slow
fast
(A) slow = head
(B) slow → 1 step
fast → 1 step
meet
S.P of loop


slow
fast
n Km/hr
2n Km/hr
n Km
2n Km
Distance travelled by fast pointer = 2 × n [Distance travelled by slow pointer]
A
B
C
slow
fast

Distance travelled by fast pointer = 2 × Distance travelled by slow pointer

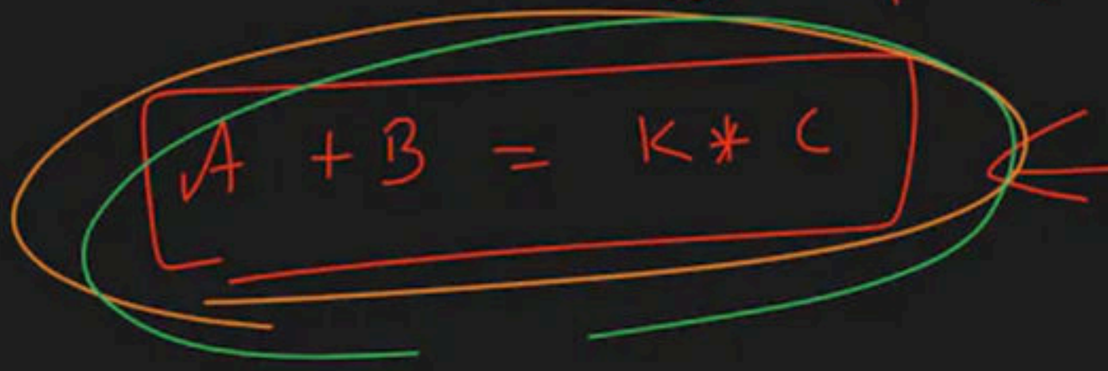
$$A + K_1 \times C + B = 2 * [A + K_2 C + B]$$

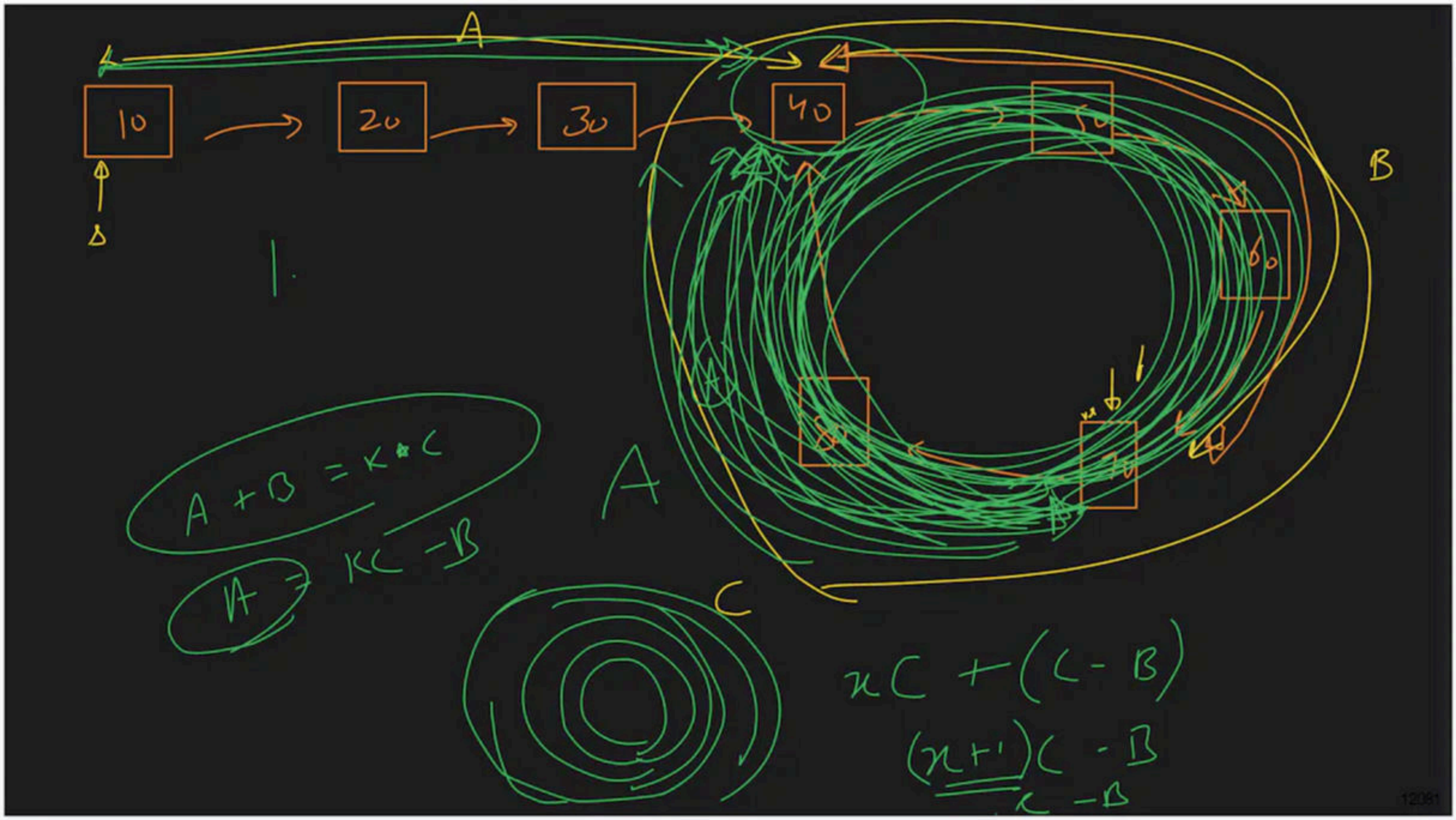
$$A + K_1 C + B = 2A + 2K_2 C + 2B$$

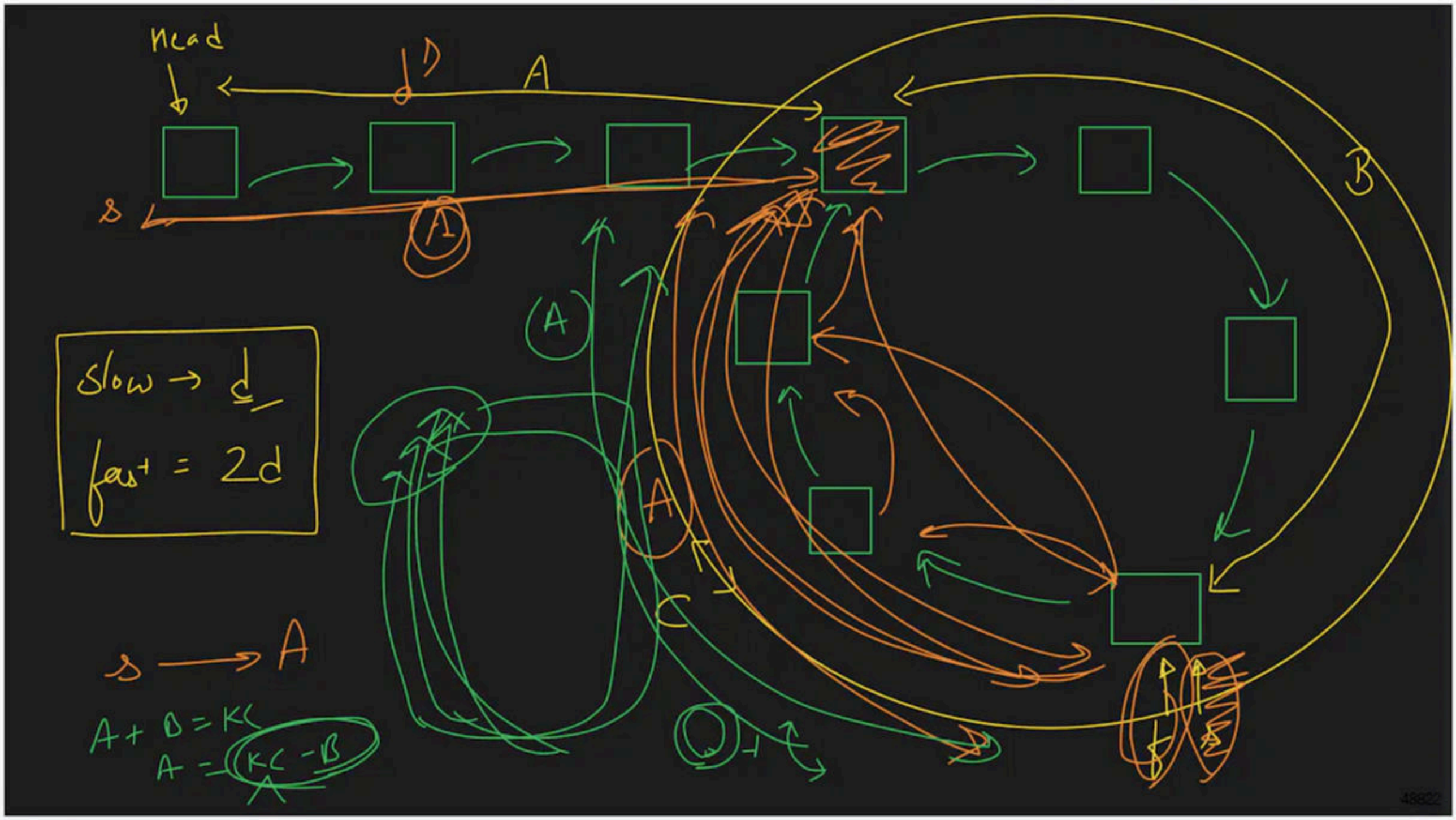
$$(K_1 - K_2) C = A + B$$

let $K_1 - K_2 = K$

$$A + B = K * C$$




A
10
20
30
40
50
60
B
A
1.
A + B = K * C
A = KC - B
C
nC + (C - B)
(n+1)C - B
C - B


head
D
A
B
A
A
A
C
slow → d
fast = 2d
s → A
A + B = KC
A = (KC - B)
A

fast = 2d

fast = 2 * slow

Ikram

distance travelled by fast = 2 * distance travelled by slow

$nC + C - B$

$(n+1)C - B$

$A + K_1 * C + B = 2 * [A + K_2 * C + B]$

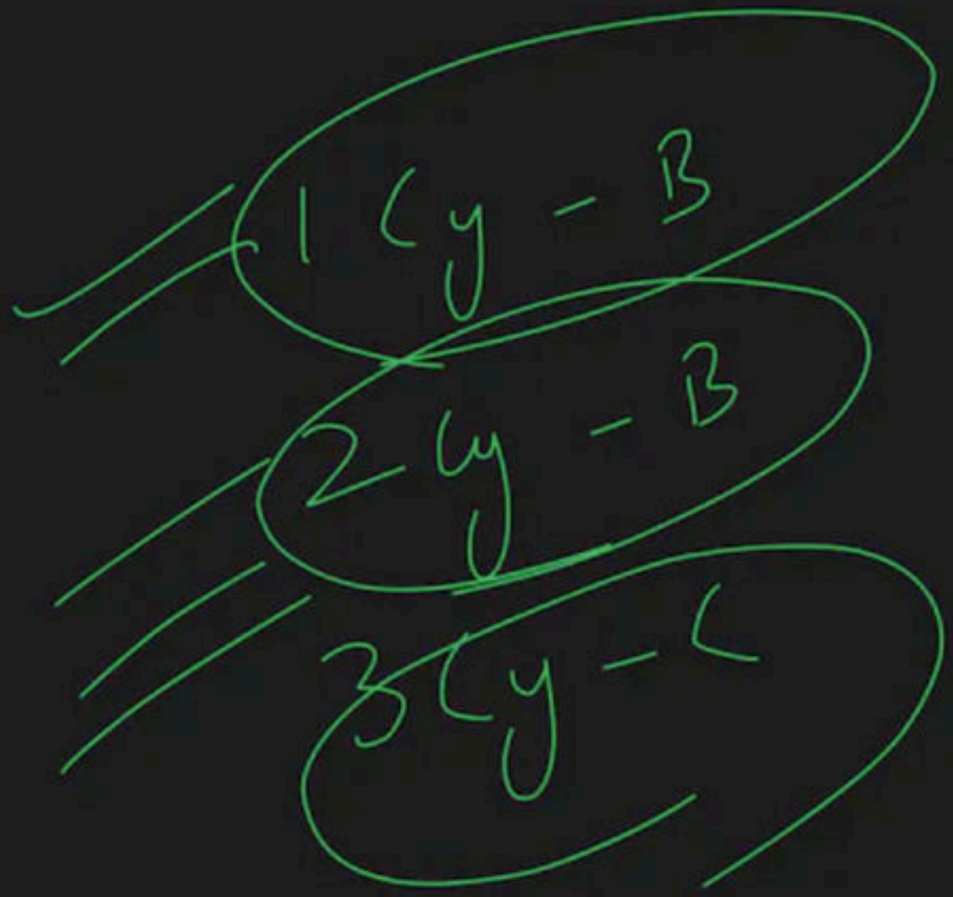
$A + B + K_1 C = 2A + 2B + 2K_2 C$

$K_1 - K_2 = K$

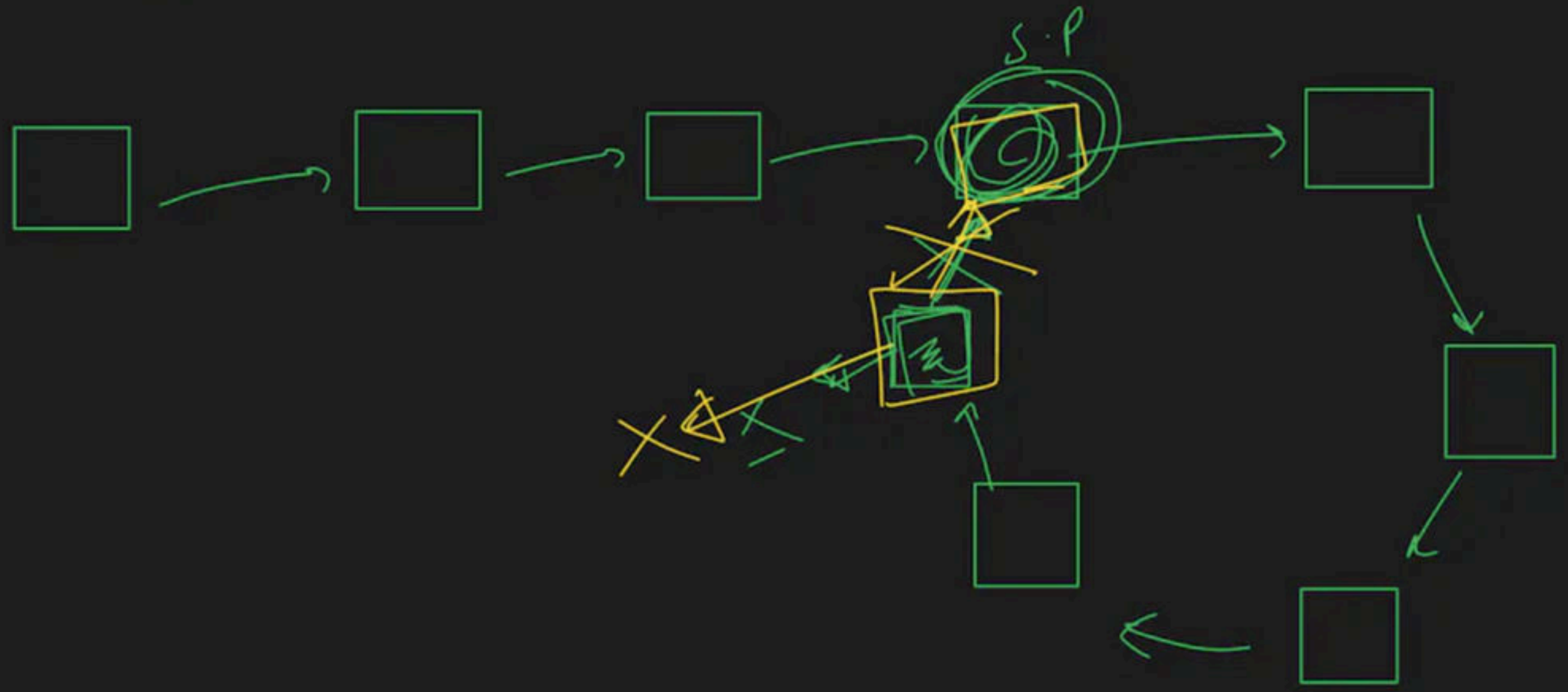
$K_1 C - K_2 C = 2A + 2B - A - B$

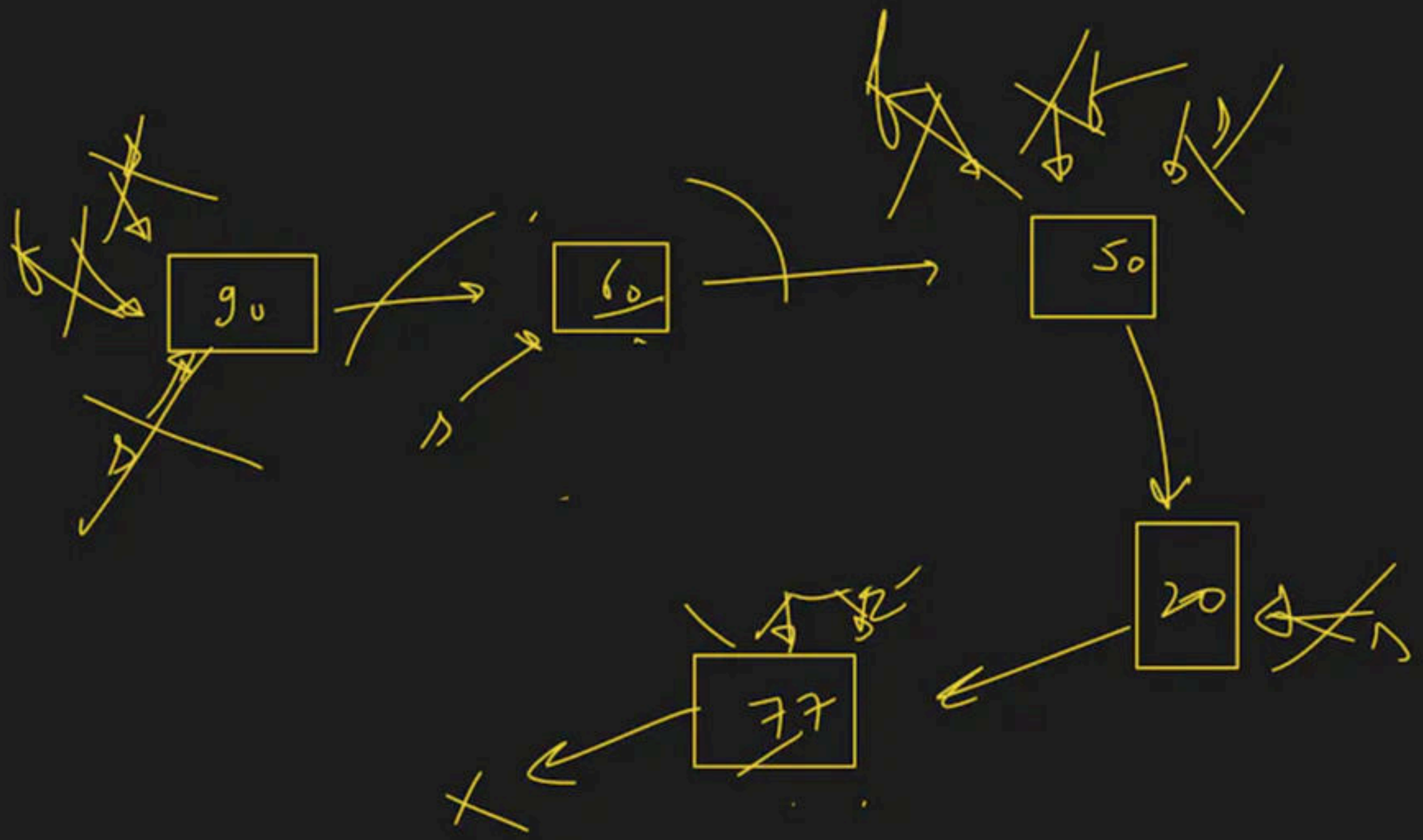
$A + B = K * C$

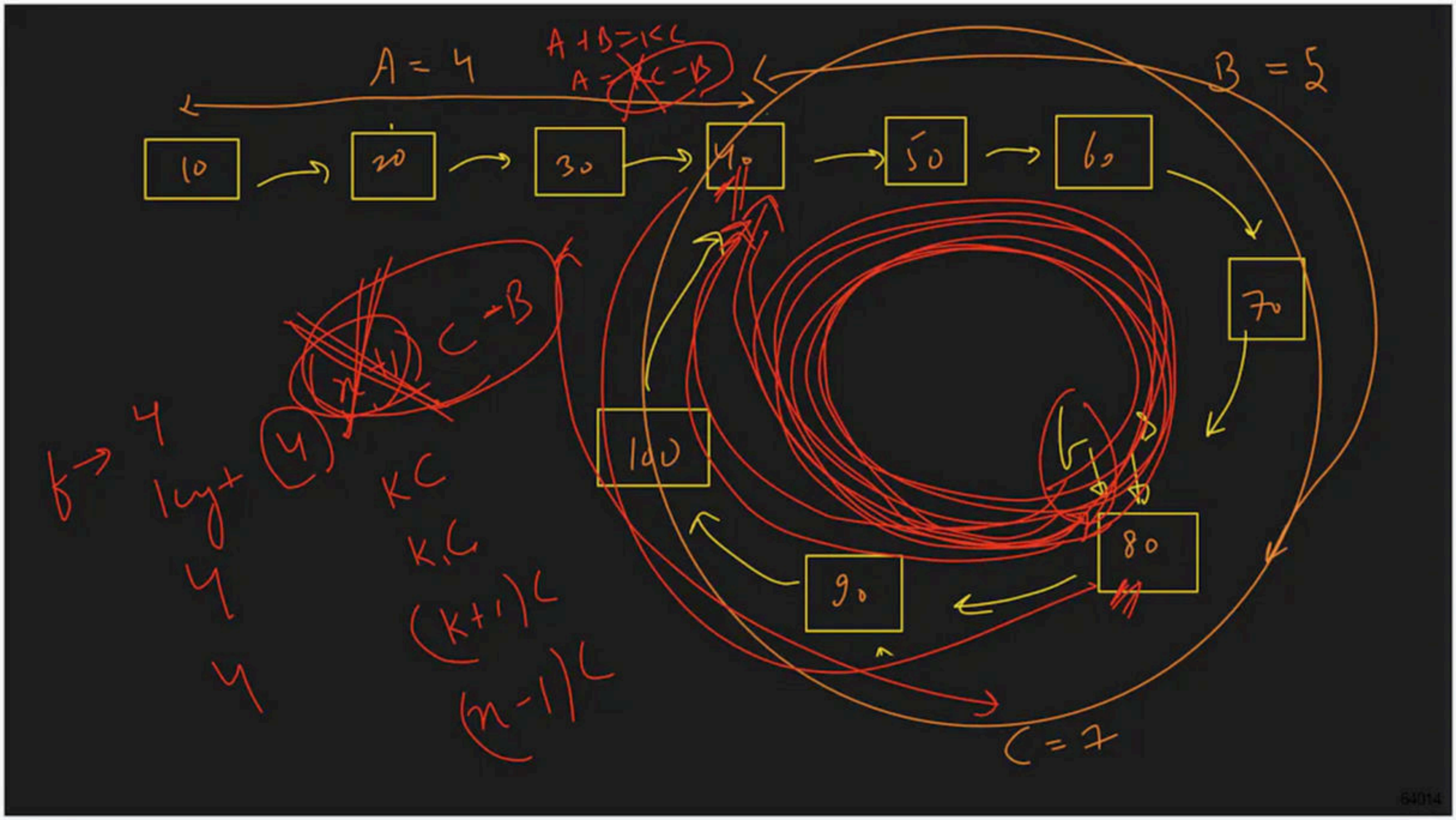
$(K_1 - K_2)C = A + B$


1 (y - B
2 y - B
3 (y - C

③ Remove a Loop

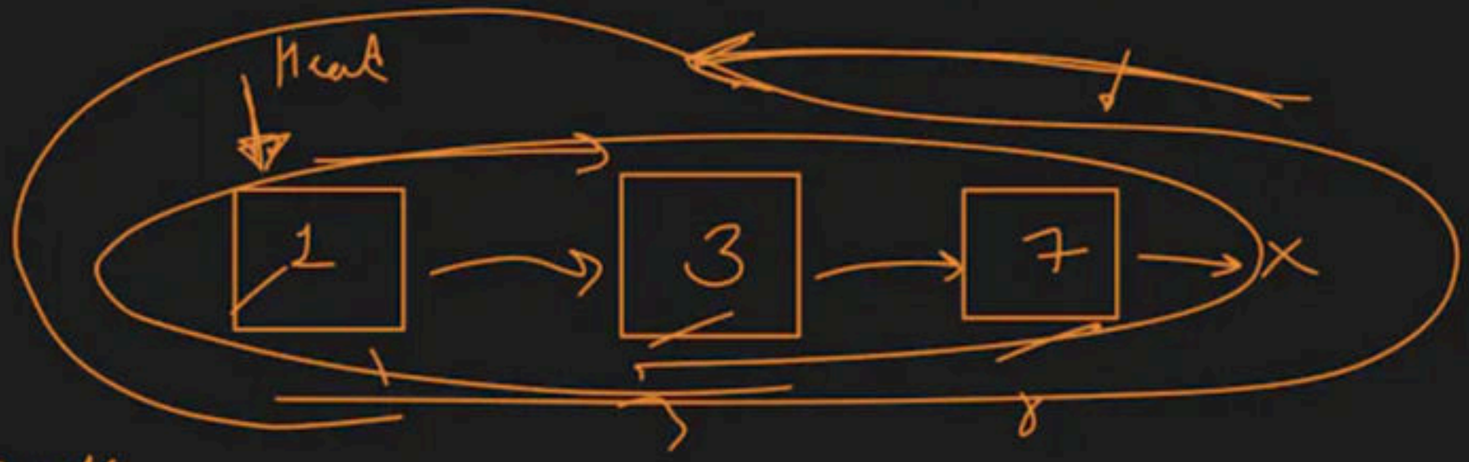



90
60
50
20
77
X


A = 4
A + B = 1 C C
A = C - B
B = 5
10
20
30
40
50
60
70
80
90
100
C - B
K C
K C
(K+1) C
(n-1) C
C = 7

2 min

Add 1 to a Linked List

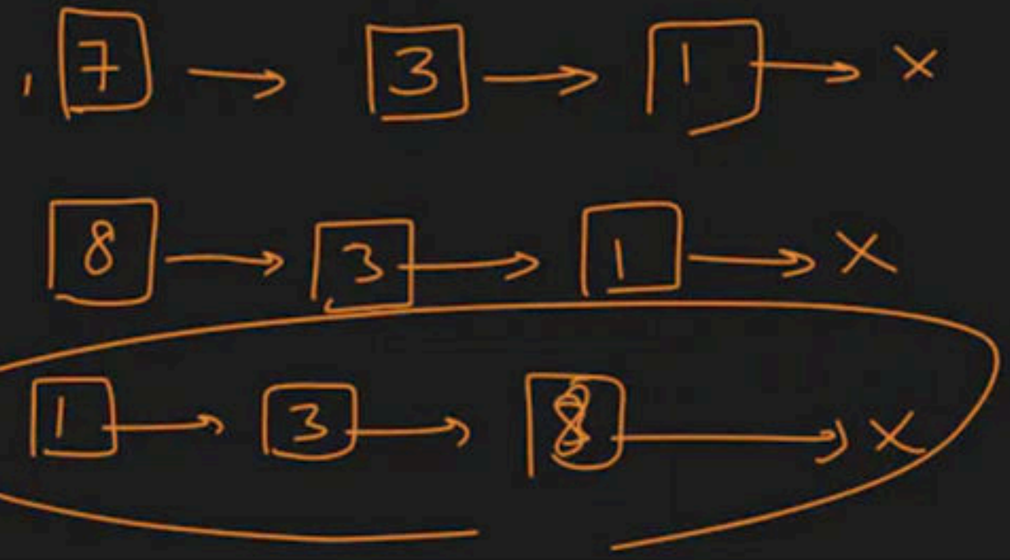


Carry -> 0 -> (Head

137

133
+1
134

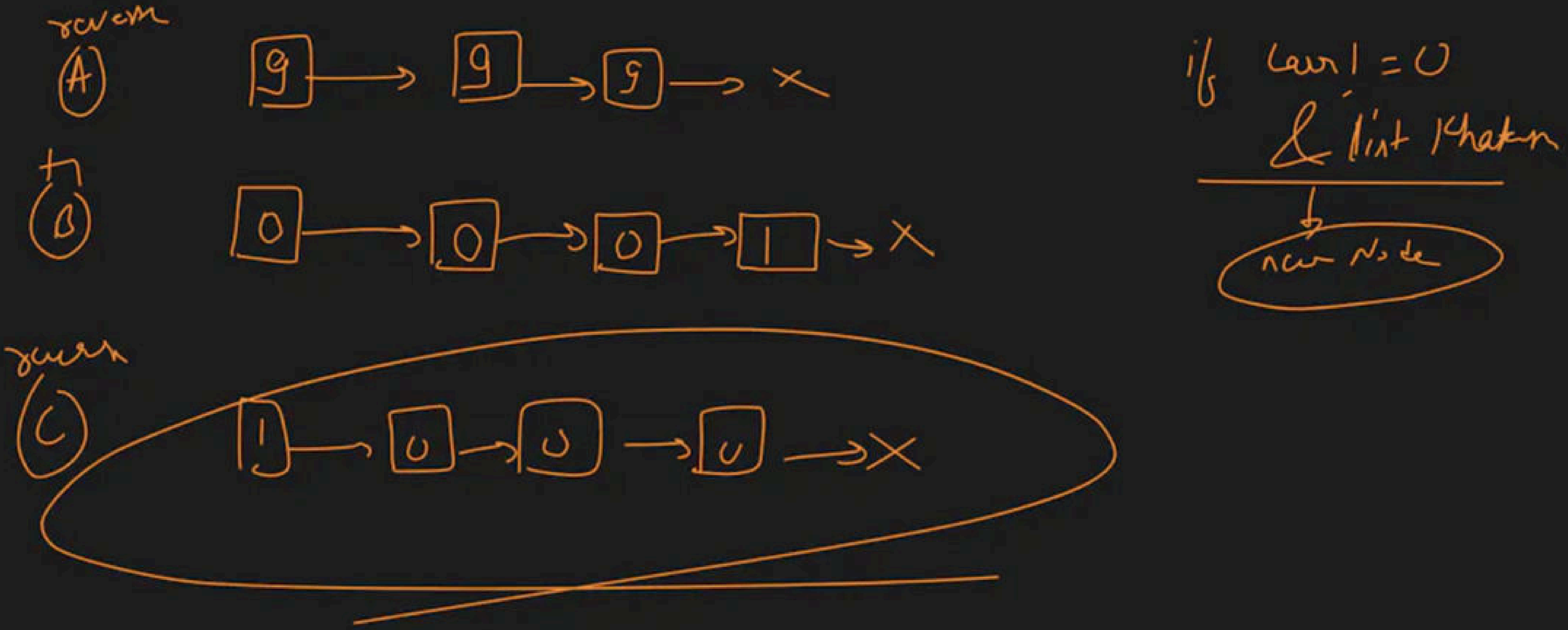
reverse
A
+
B
reverse
C



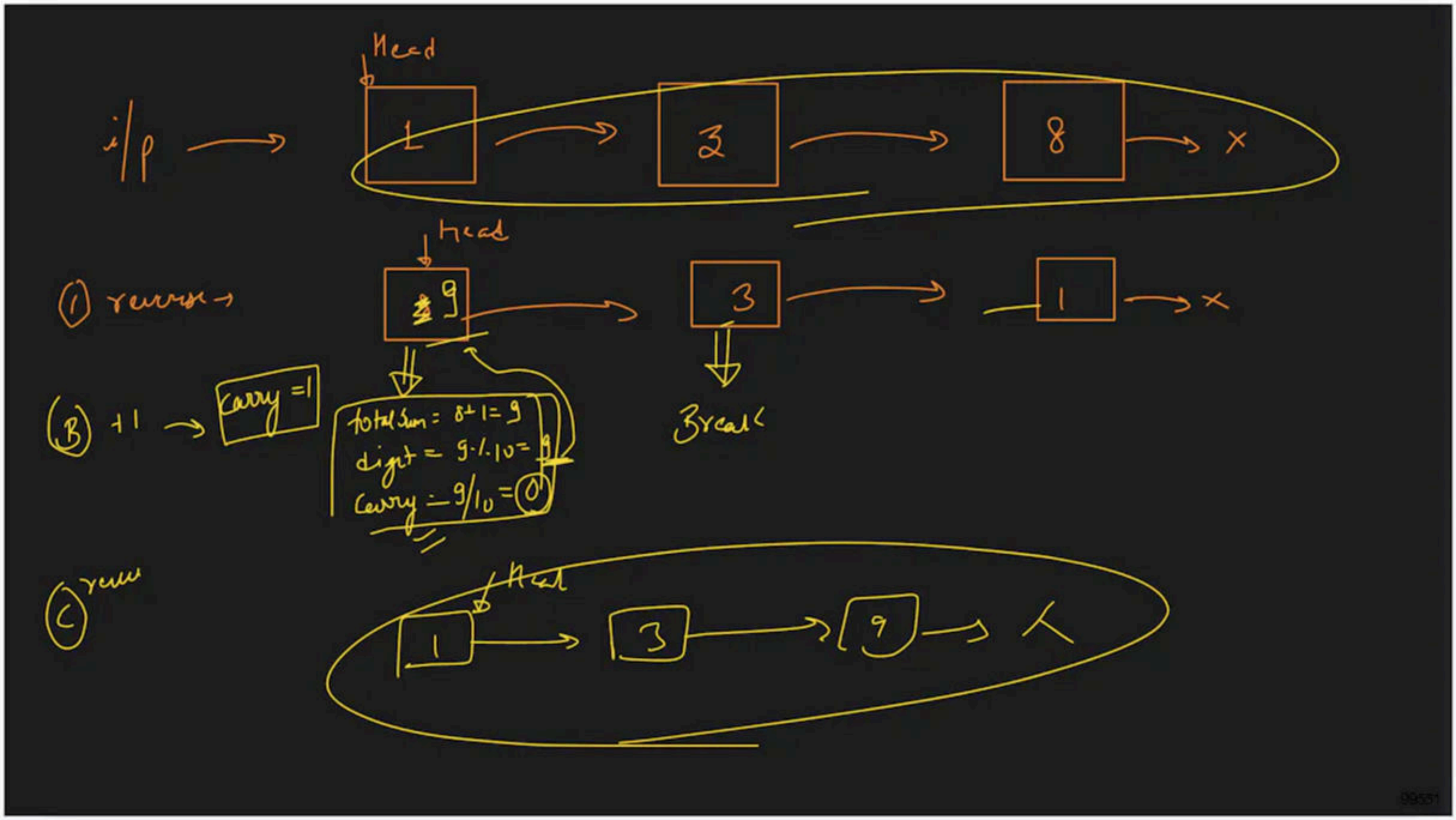
1) rev
2) add
3) rev

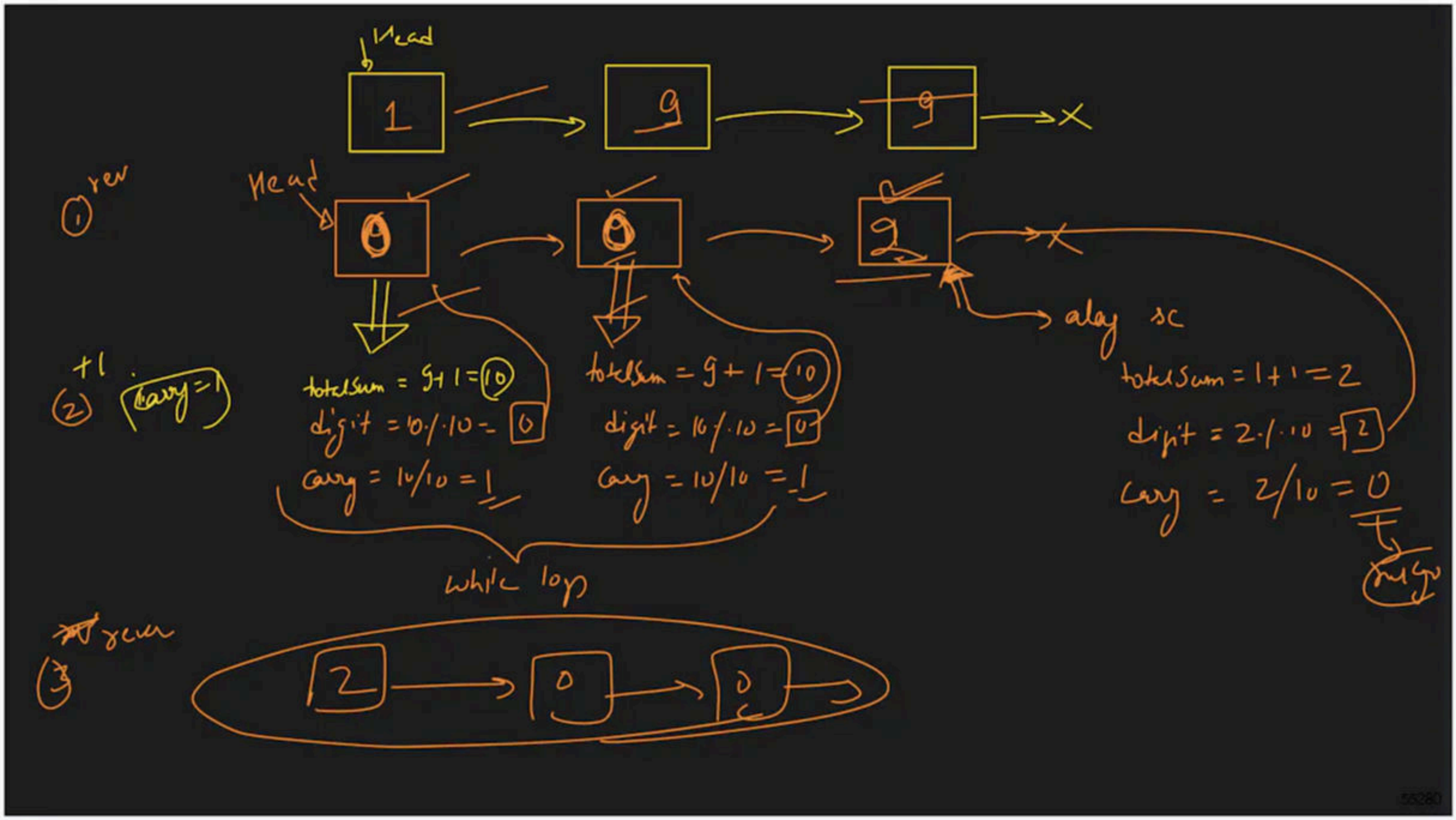
139
+1
140

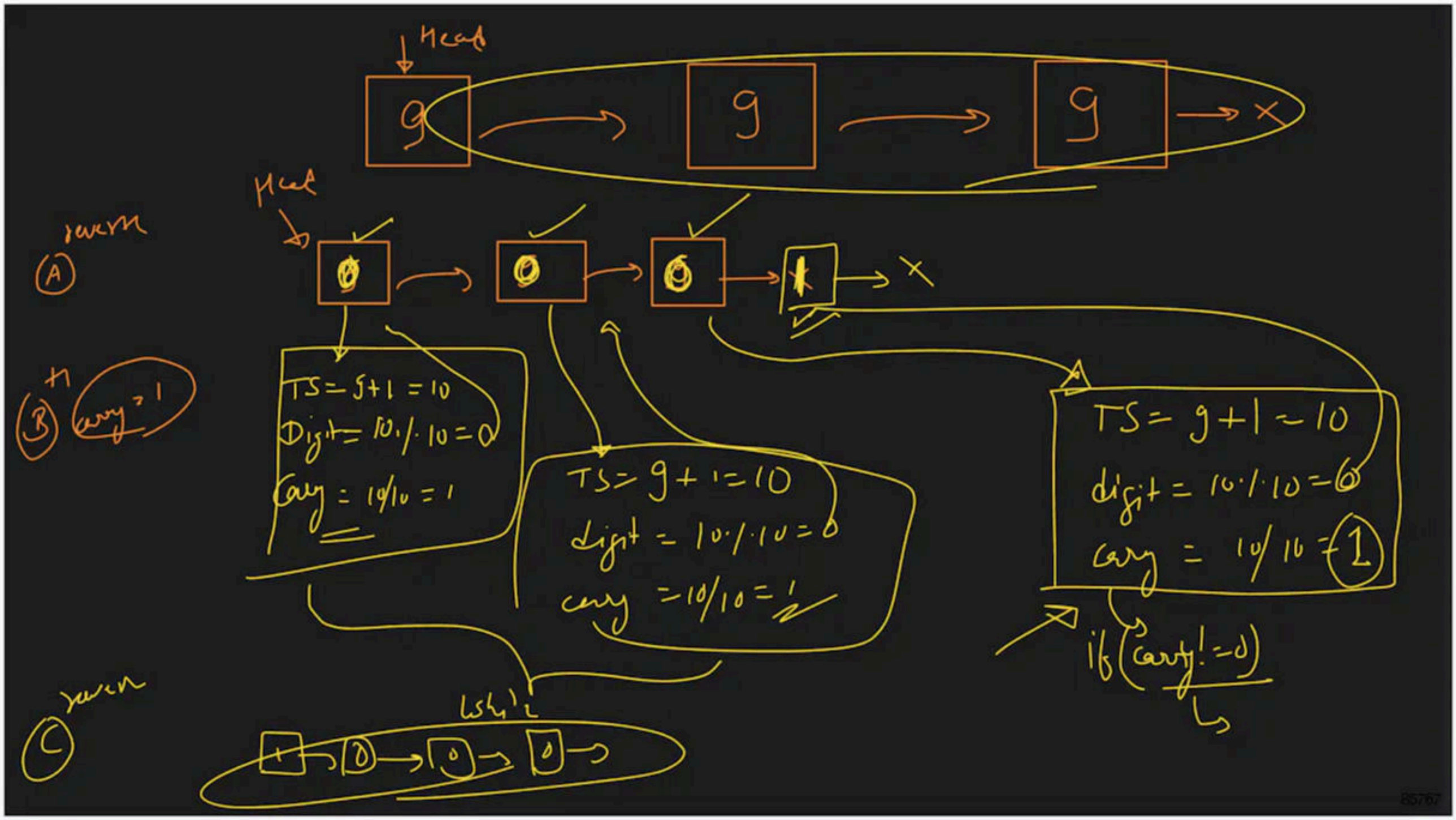
1
9 9 9
+1
1000



reverse
A
9 → 9 → 9 → X
+
B
0 → 0 → 0 → 1 → X
reverse
C
1 → 0 → 0 → 0 → X
if carry! = 0
& list khatam
new Node

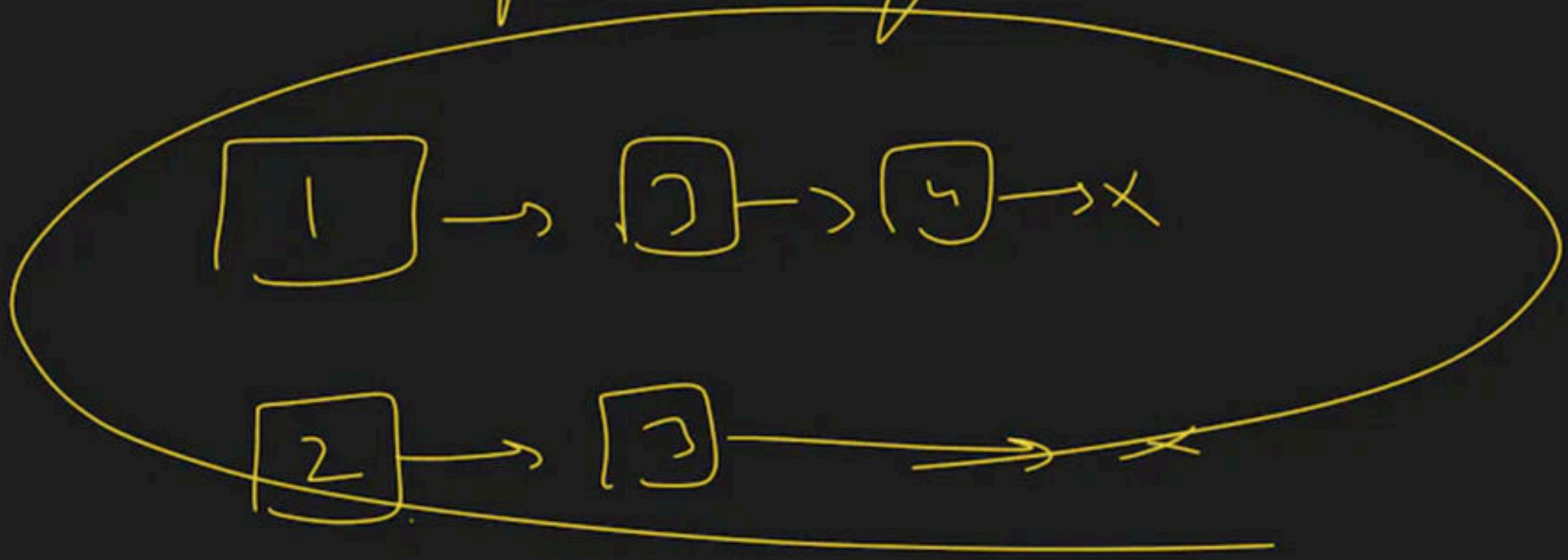

Head
i/p
1
3
8
x
Head
(1) recursion
9
3
1
x
(B) +1
carry = 1
totalSum = 8+1 = 9
digit = 9%10 = 9
carry = 9/10 = 0
3rd call
(C) recursion
Head
1
3
9
∧


head
1
9
9
X
rev
head
0
0
9
alay sc
+1
carry = 1
totalSum = 9+1 = 10
digit = 10 % 10 = 0
carry = 10/10 = 1
totalSum = 9+1 = 10
digit = 10 % 10 = 0
carry = 10/10 = 1
totalSum = 1+1 = 2
digit = 2 % 10 = 2
carry = 2/10 = 0
while loop
rev
2
0
0


Head
9
9
9
X
Head
0
0
0
1
X
reverse
A
B
carry = 1
TS = 9+1 = 10
Digit = 10 % 10 = 0
carry = 10/10 = 1
TS = 9+1 = 10
digit = 10 % 10 = 0
carry = 10/10 = 1
TS = 9+1 = 10
digit = 10 % 10 = 6
carry = 10/10 = 1
if (carry != 0)
C
reverse
1
0
0
0

add 2 numbers ✓✓

represented by Linked List



2 min Break

after that → Add 2 no. Rep in LL

=) a i/p 2 → 4 → 3 → X

b i/p 5 → 6 → 4 → X

o/p 

a => 342

b => 465

807

(1)

$3 \to 2 \to 4 \to X$

$9 \to X$

```
  0 1 0
  4 2 3
+   0 9
-------
  4 3 2
-------
```

(2)

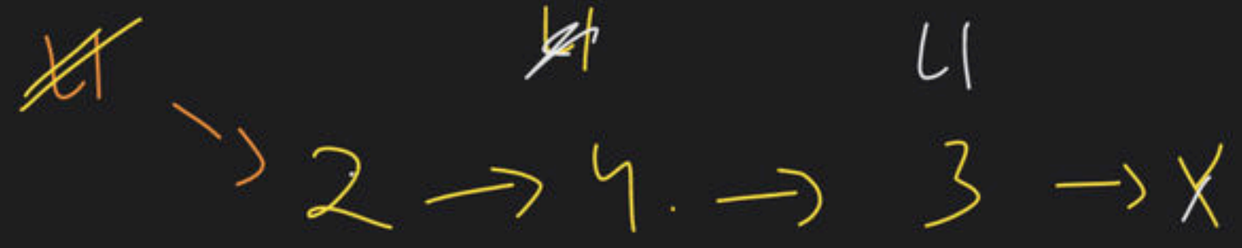
$9 \to 9 \to 9 \to X$

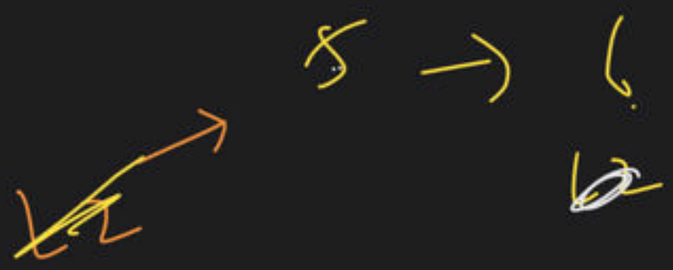
$1 \to X$

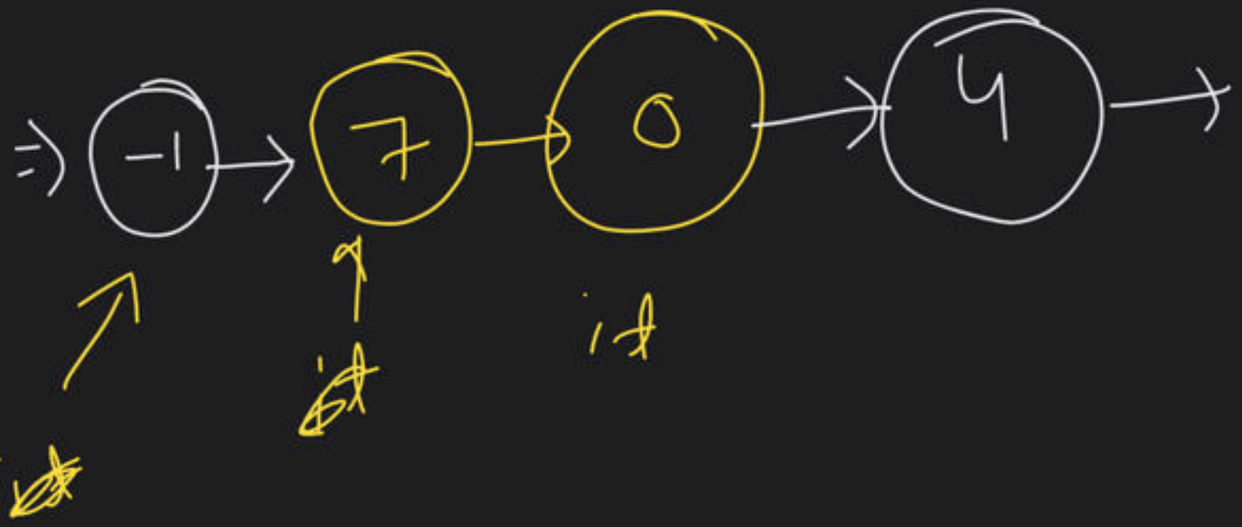
$0 \to 0 \to 0 \to 1 \to X$

```
  1 1 1
  9 9 9
+   1
-------
  1 0 0 0
```

```
  4 2 3
+ 0 0 9
-------
  4 3 2
==
```







=) l2.1 = 0
L2 → d

int carry = 0;

(3) 0 1

sum = (l1 → d) + L2 → d + carry

digit = sum % 10;

carry = sum / 10;

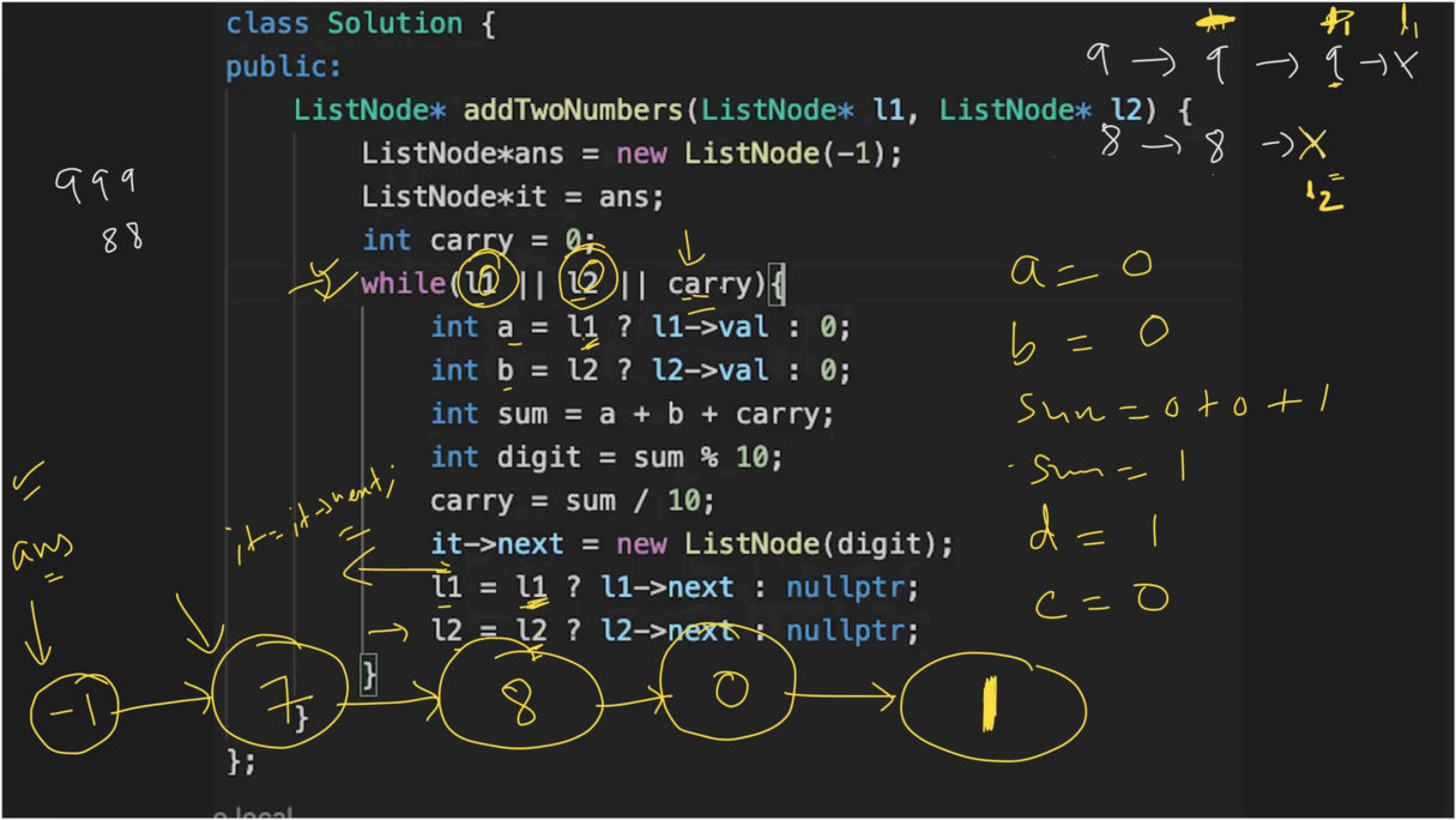
it → next = new Node(digit)

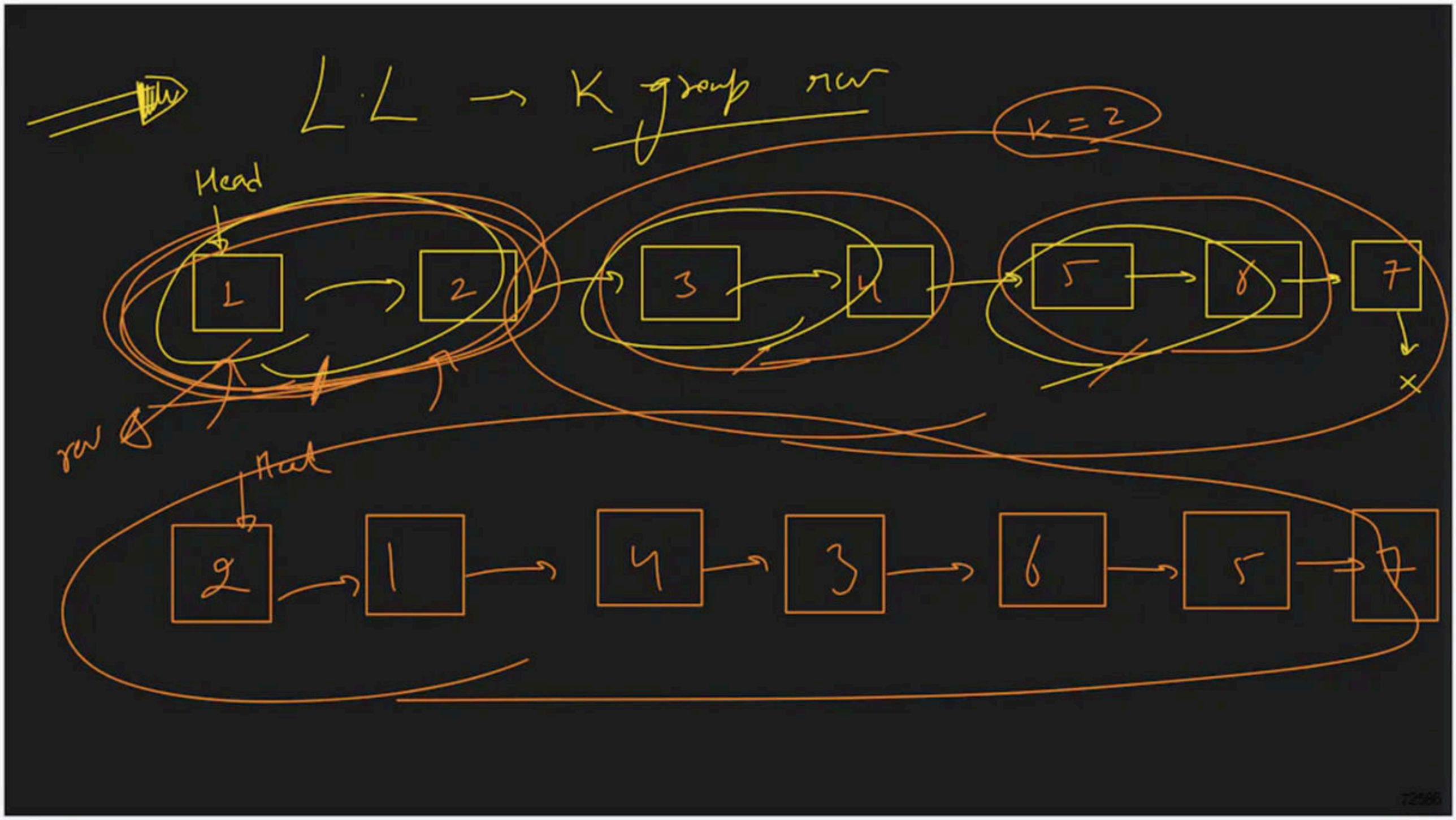
it it++

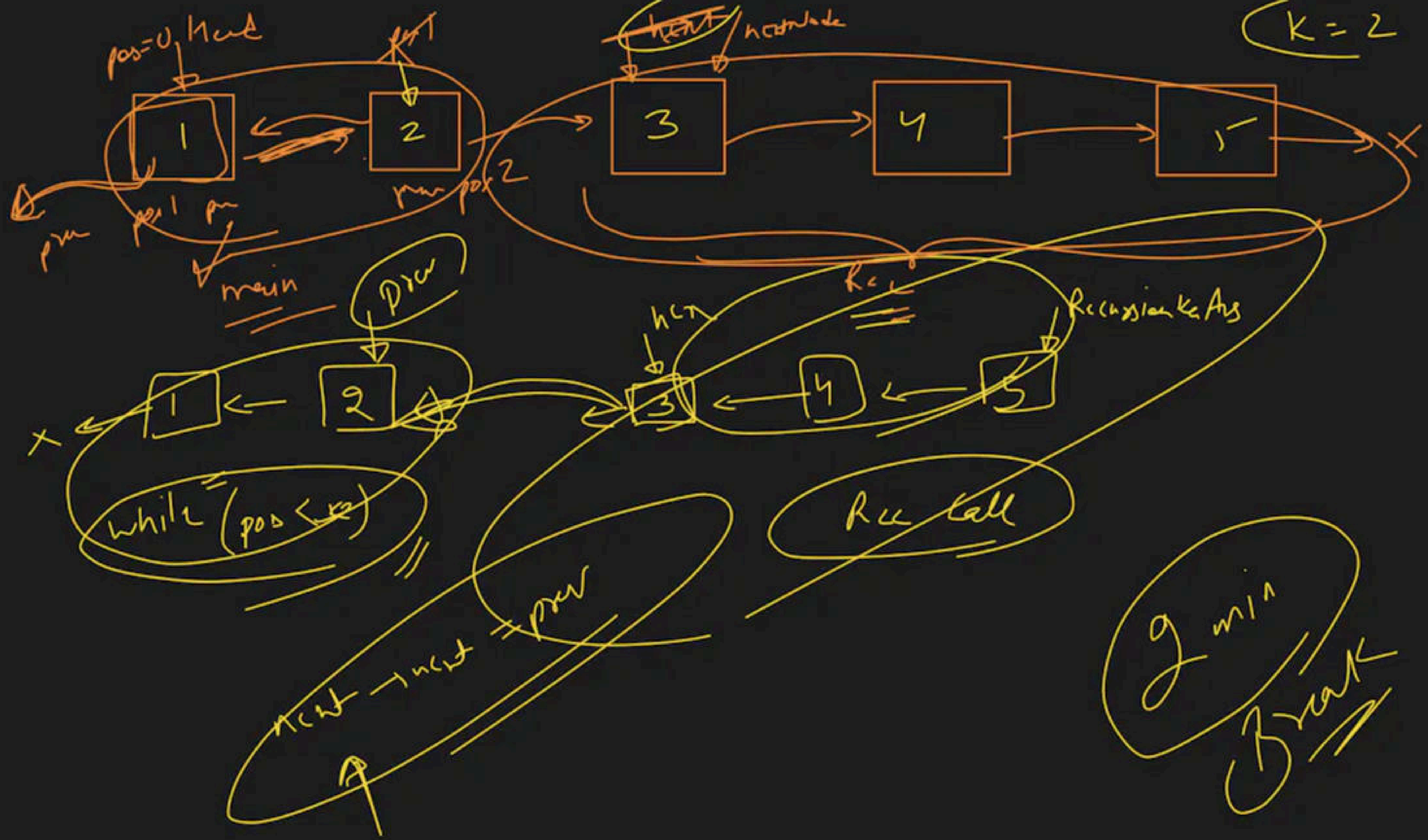
L1 it++

L2 it++

```cpp
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode*ans = new ListNode(-1);
        ListNode*it = ans;
        int carry = 0;
        while(l1 || l2 || carry){
            int a = l1 ? l1->val : 0;
            int b = l2 ? l2->val : 0;
            int sum = a + b + carry;
            int digit = sum % 10;
            carry = sum / 10;
            it->next = new ListNode(digit);
            l1 = l1 ? l1->next : nullptr;
            l2 = l2 ? l2->next : nullptr;
        }
        it = it->next;
        return ans->next;
    }
};
```




L.L → K group rev
K=2
Head
1
2
3
4
5
6
7
x
rev
Nat
2
1
4
3
6
5
7



K = 2
pos=0, hcurr
nextnode
1
2
3
4
5
prev
main
prev
Recursion ke Ans
Rec
while (pos < k)
Rec call
hcurr → next = prev
9 min
Break

(2)

ListNode * reverseKGroup(ListNode * head, int& K)

{

Leetcode 25

Linked List Reverse in K-groups

```
if (head == NULL)
    return head;

if (head -> next == NULL)
    return head;
```

// B.C

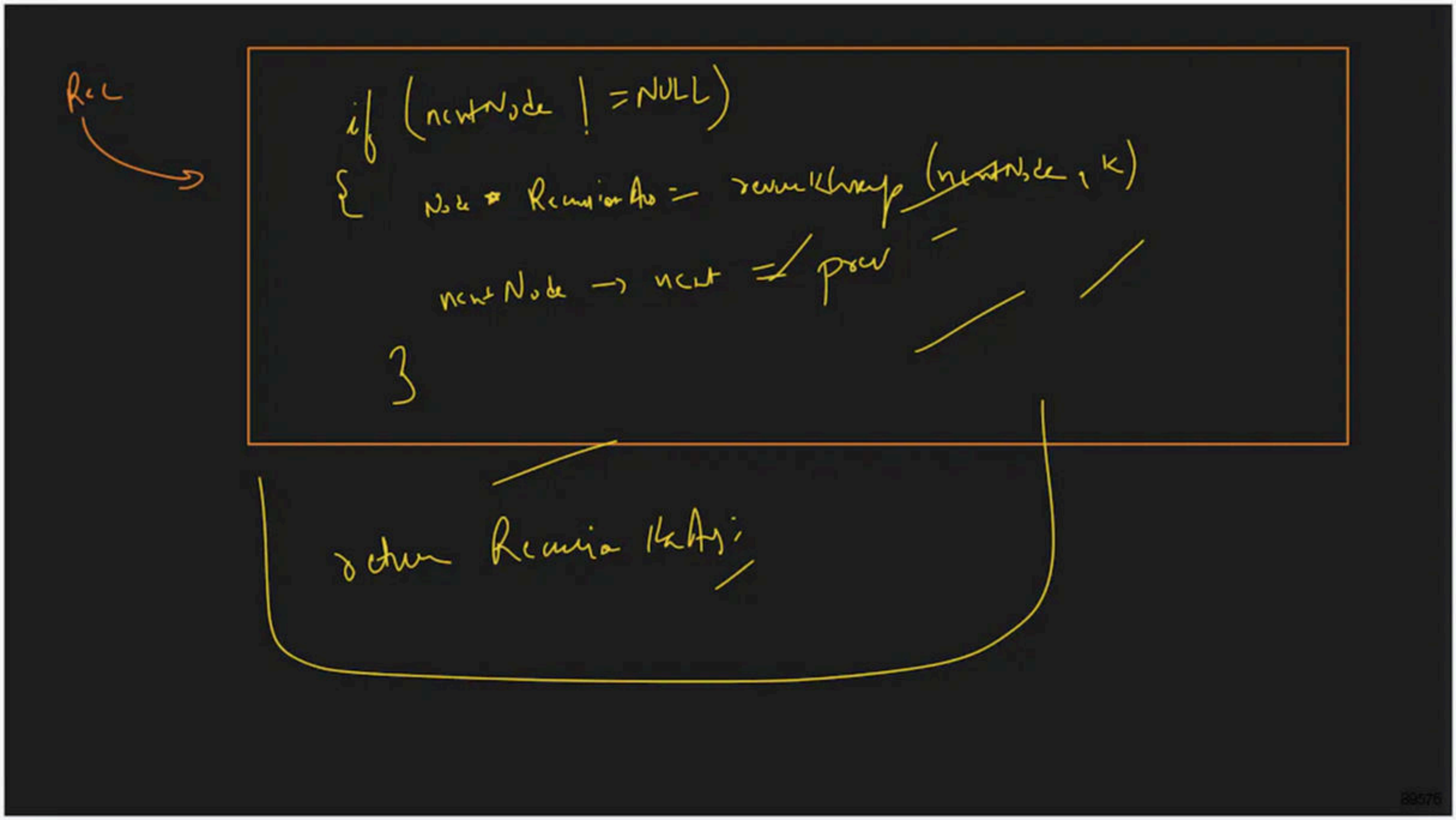
// I can first k node

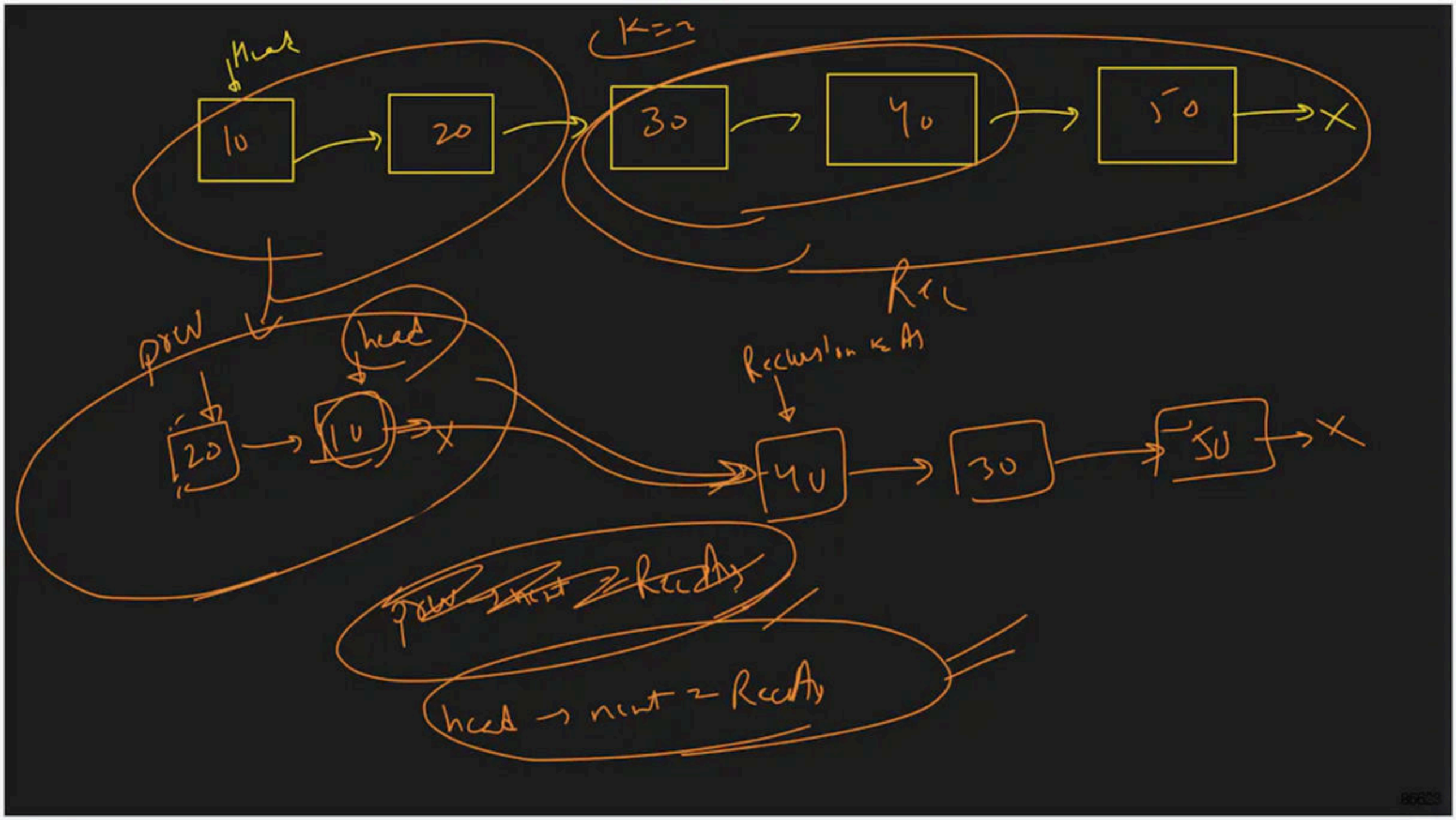
```
if (len >= K)
Node * prev = NULL;
while (pos < K)
{
    pos++;
    nextNode = curr -> next;
    curr -> next = prev;
    prev = curr
    curr = nextNode;
}
```

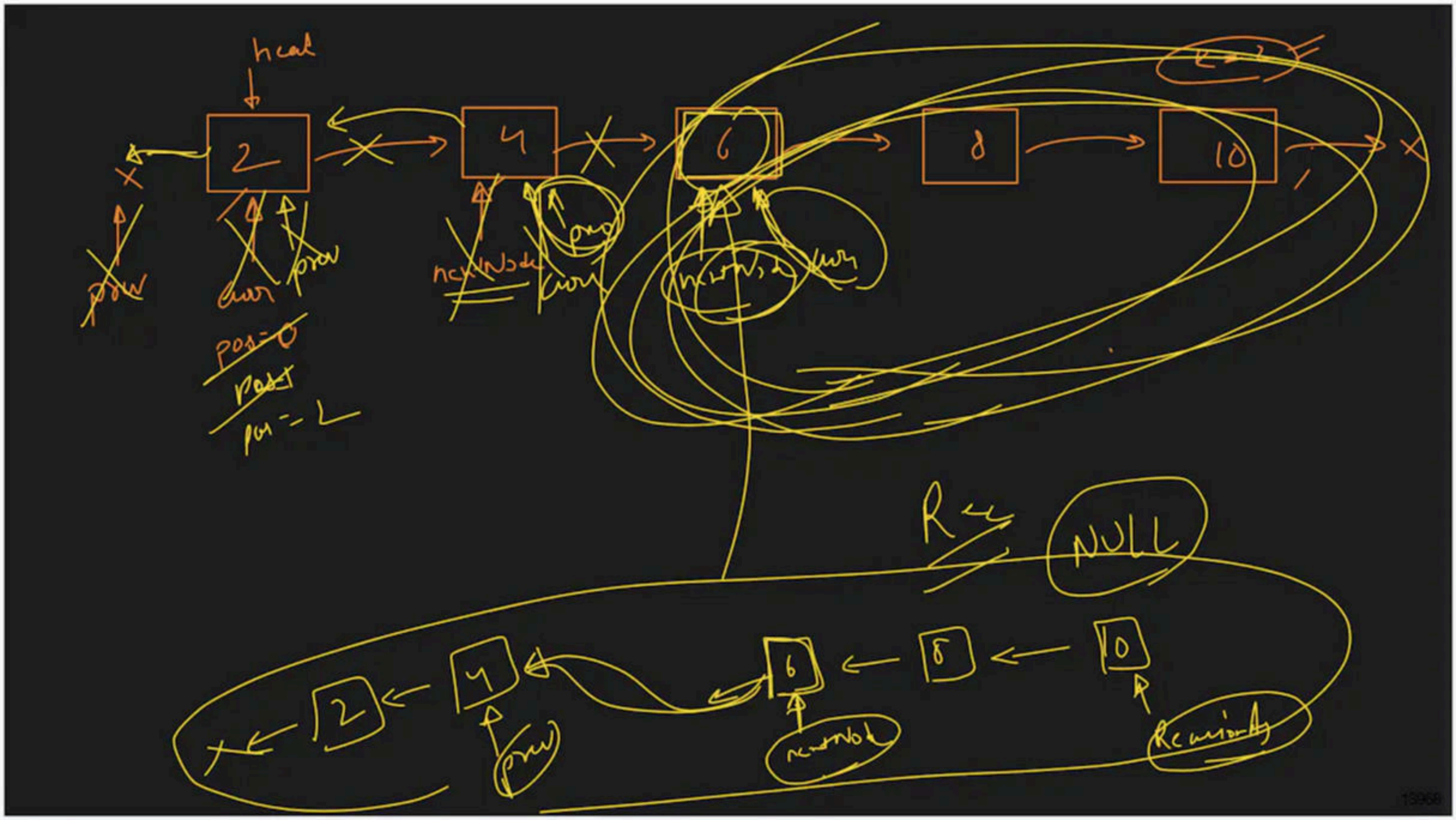
Node * next = curr -> next;

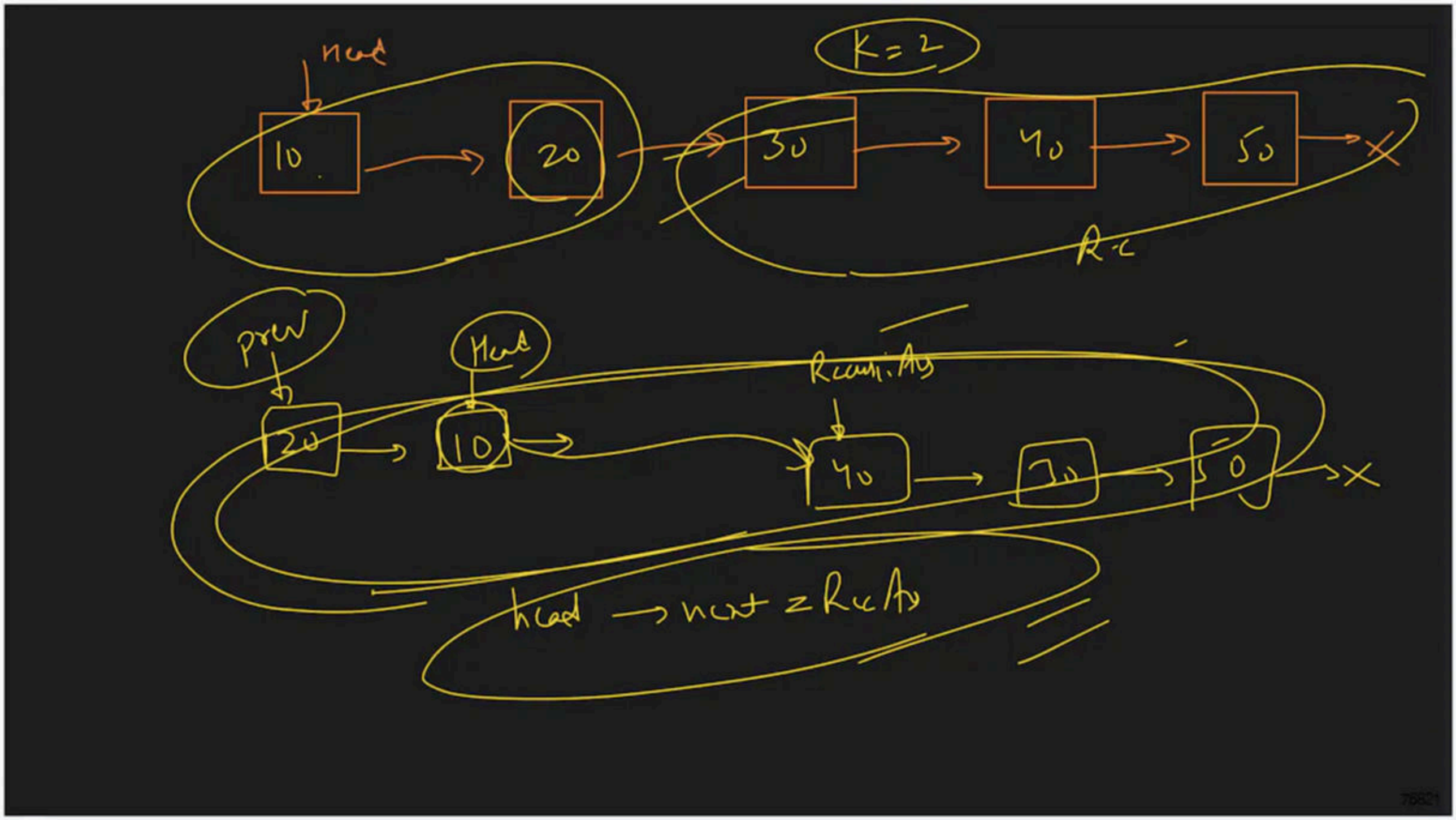
Node * curr = head;

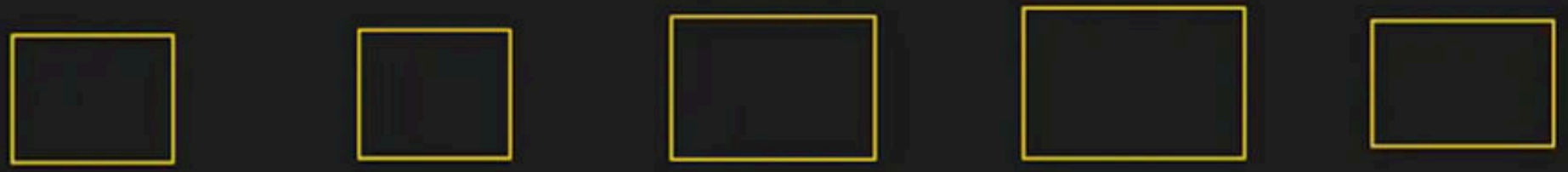
int pos = 0

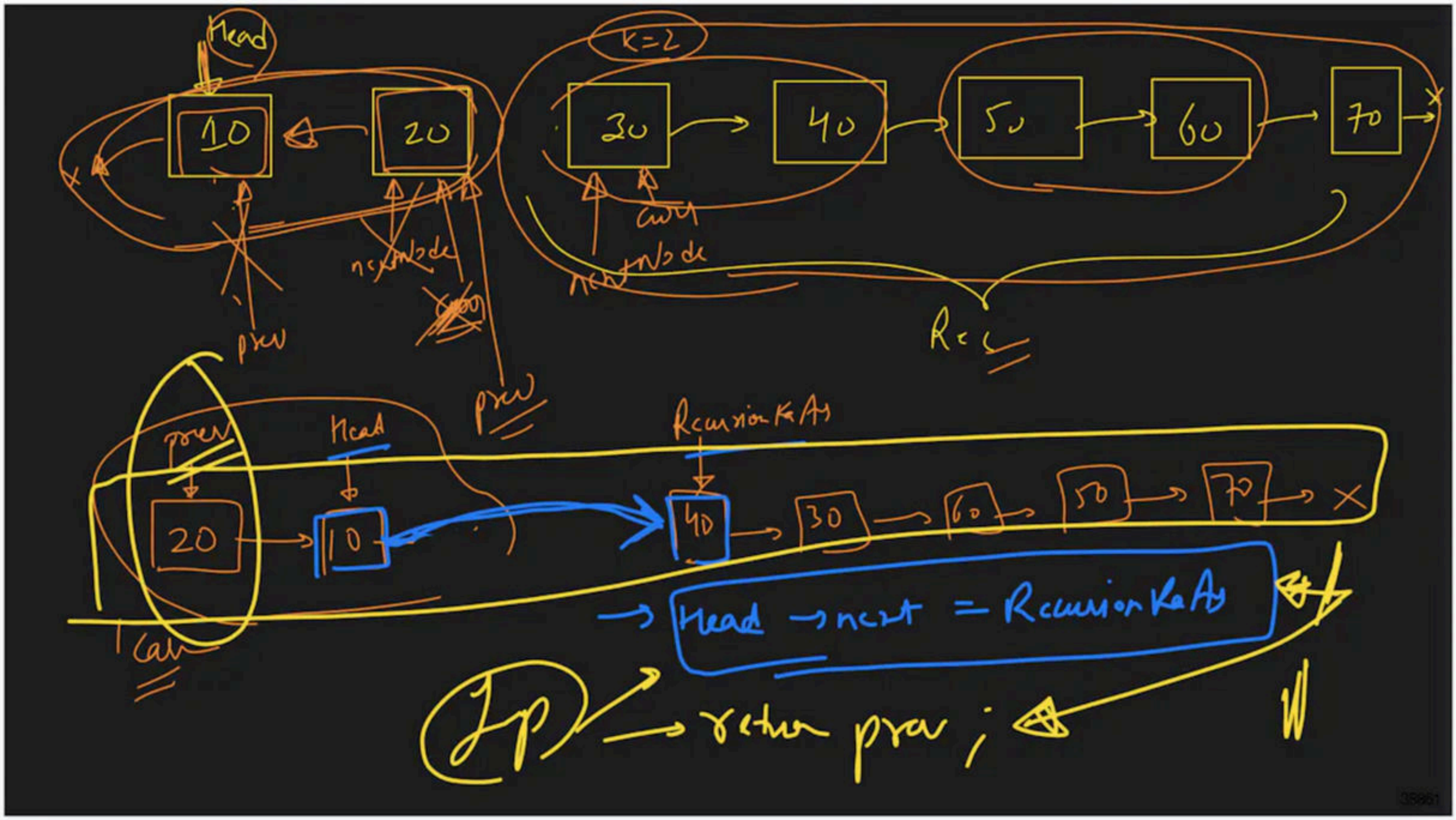

Rec
if (nextNode != NULL)
{
Node * RecursionAns = recursiveSearch(nextNode, k)
nextNode -> next = prev
3
other Recursive Work;

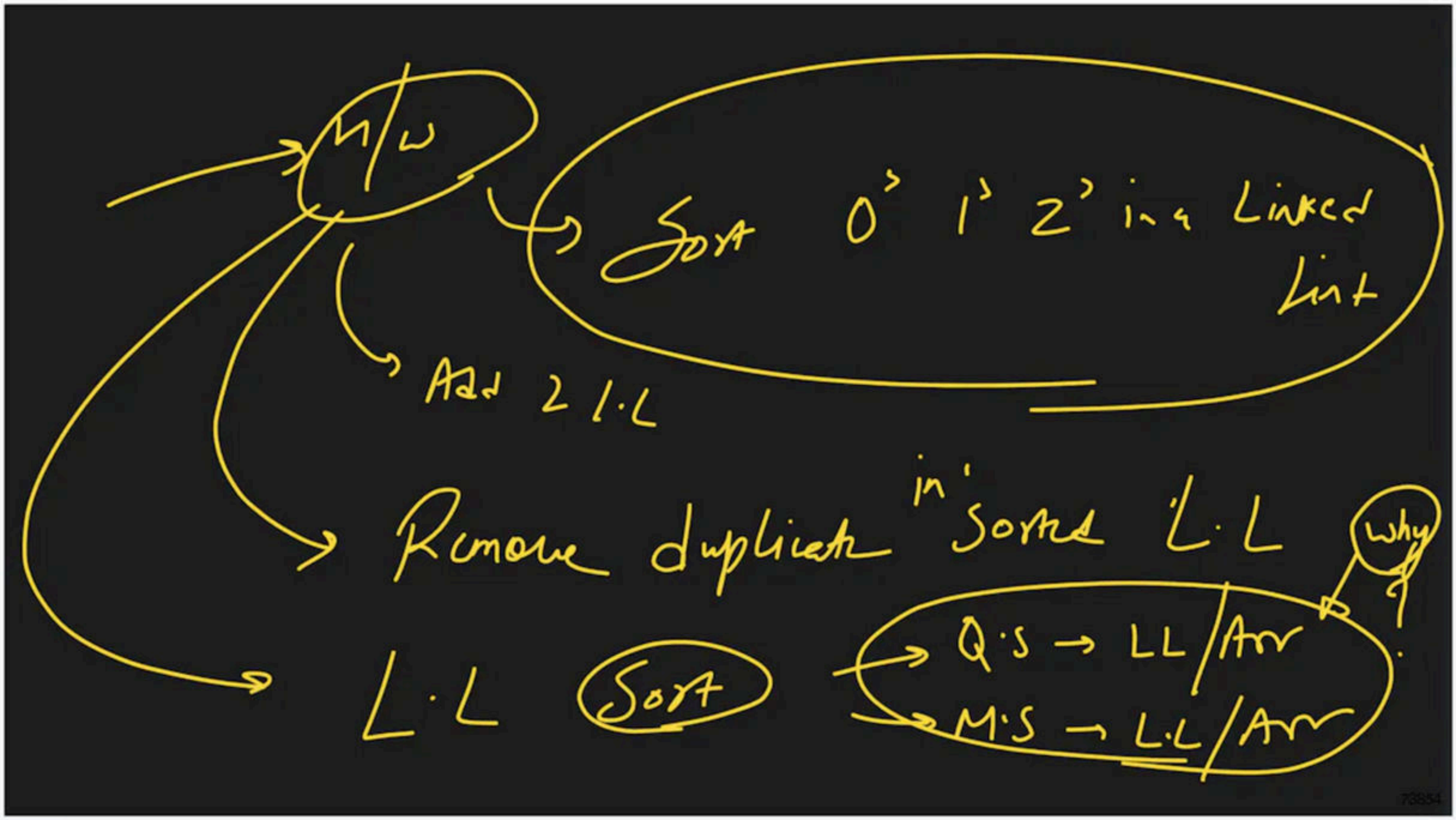

head
10
20
K=2
30
40
50
X
prev
head
20
10
X
Recursion ka Ans
40
30
50
X
prev -> next = Recursion
head -> next = Recursion


head
2
4
6
8
10
NULL

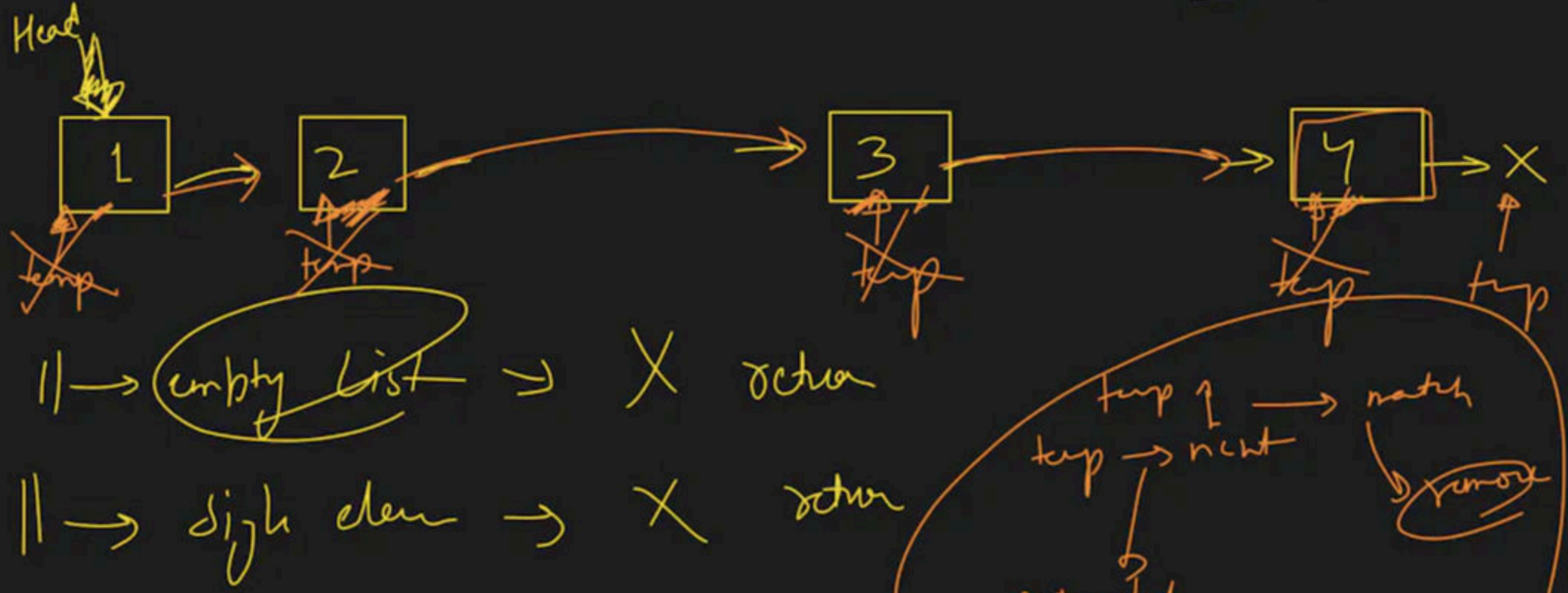

head
10
20
K=2
30
40
50
R.C
prev
head
10
20
Recur.Ab
40
50
head → next = Rec Ab




Head
10
20
30
40
50
60
70
k=2
prev
Recursion Ka Ans
20
10
40
30
60
50
70
Head → next = Recursion Ka Ans
return prev;


H/W
Sort 0's 1's 2's in a Linked List
Add 2 L.L
Remove duplicate in Sorted L.L
Why
L.L
Sort
Q.S → LL/Arr
M.S → L.L/Arr

→ Remove duplicate in Sorted L.L

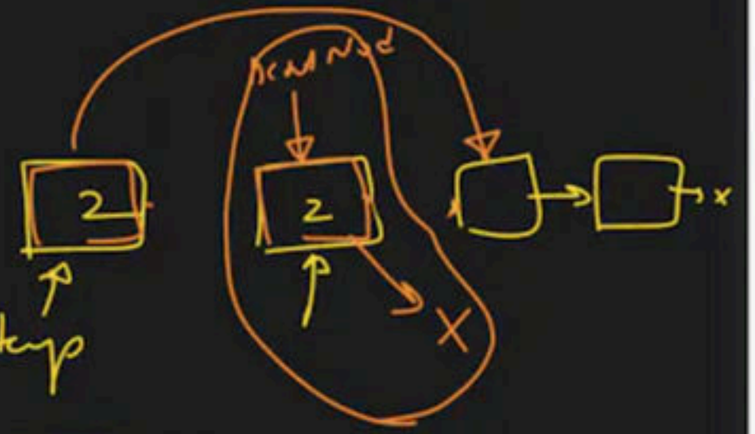


Node * Solve(Node * head)

{

if (head == NULL)

return head;



if (head -> next == NULL)

return head;

Node * temp = head;

while (temp != NULL)

{

if (temp->next != NULL && temp -> data == temp->next->data)

{

// remove

}

else { temp = temp->next; }

return head; }

// remove

Node * nextNode = temp->next

temp -> next = nextNode->next

nextNode -> next = NULL

delete nextNode