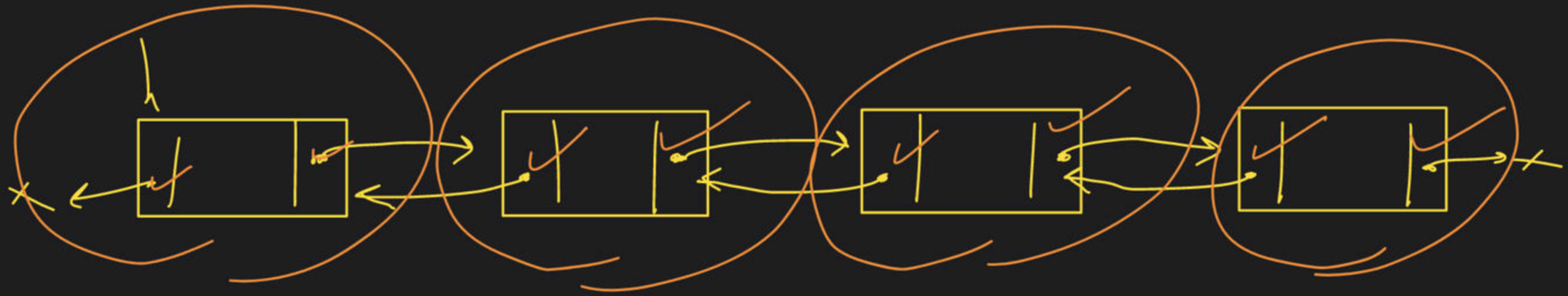
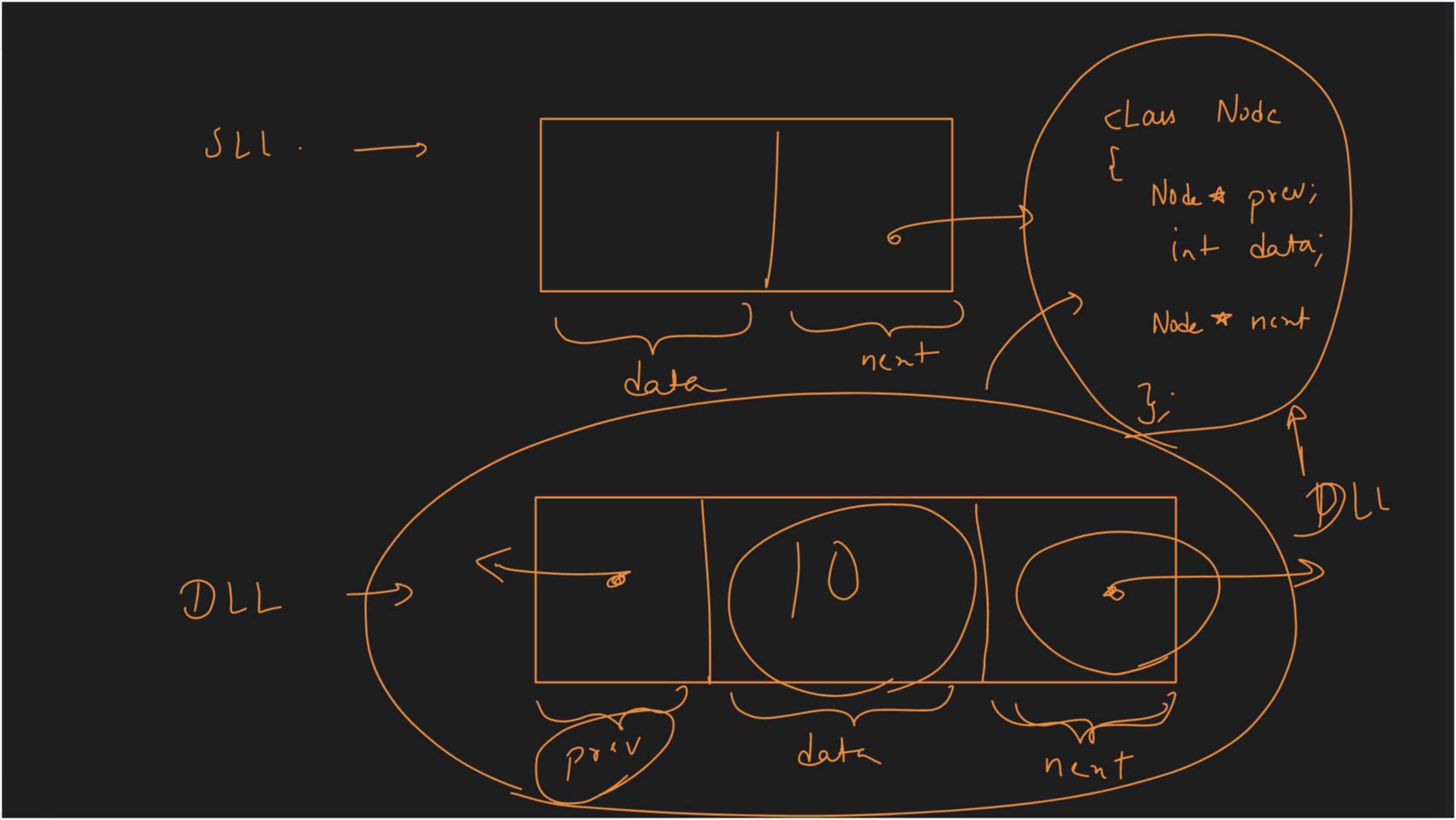
Linked List - Class 2

Special class

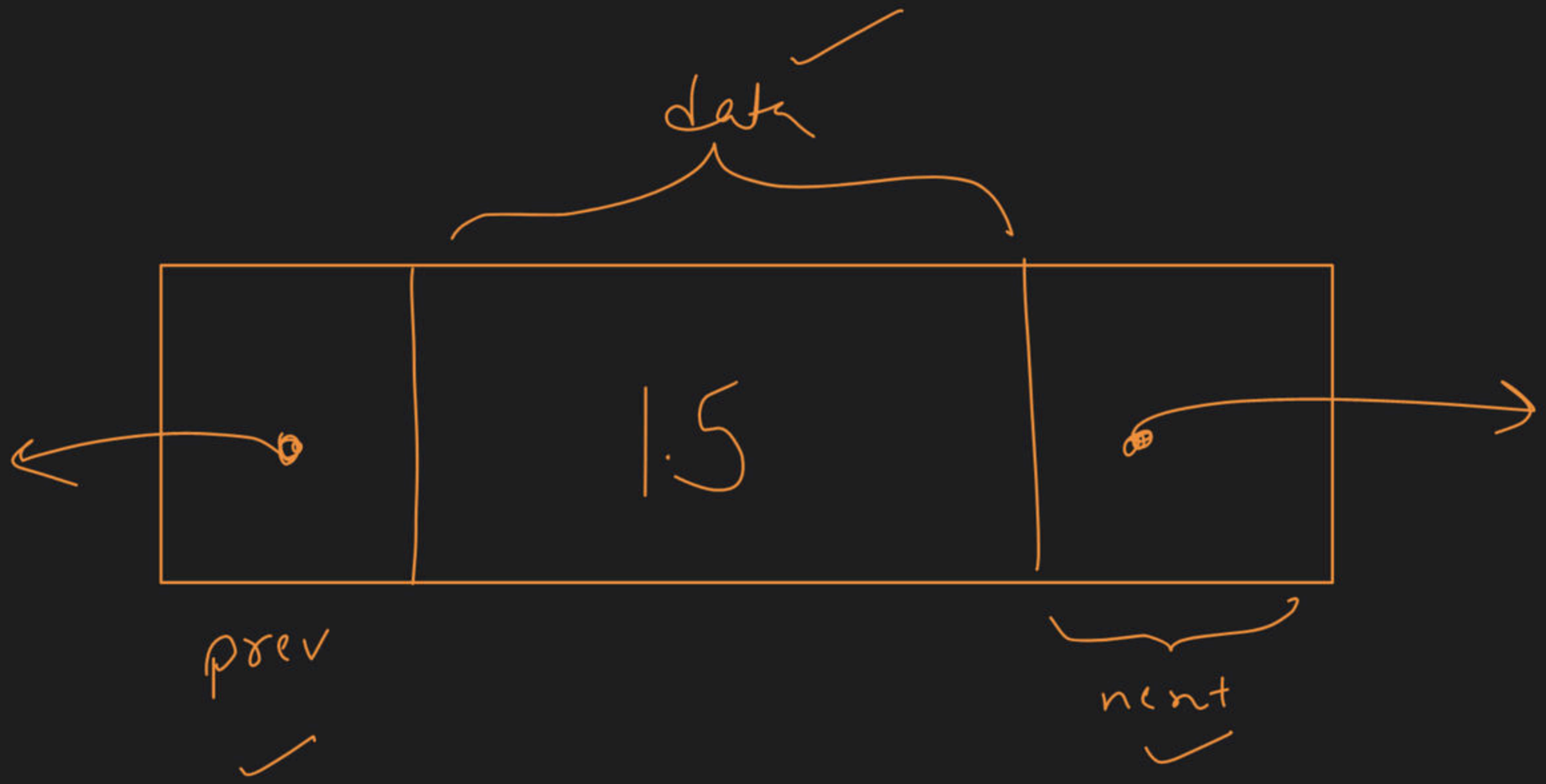
Love Babbar • July 17, 2024

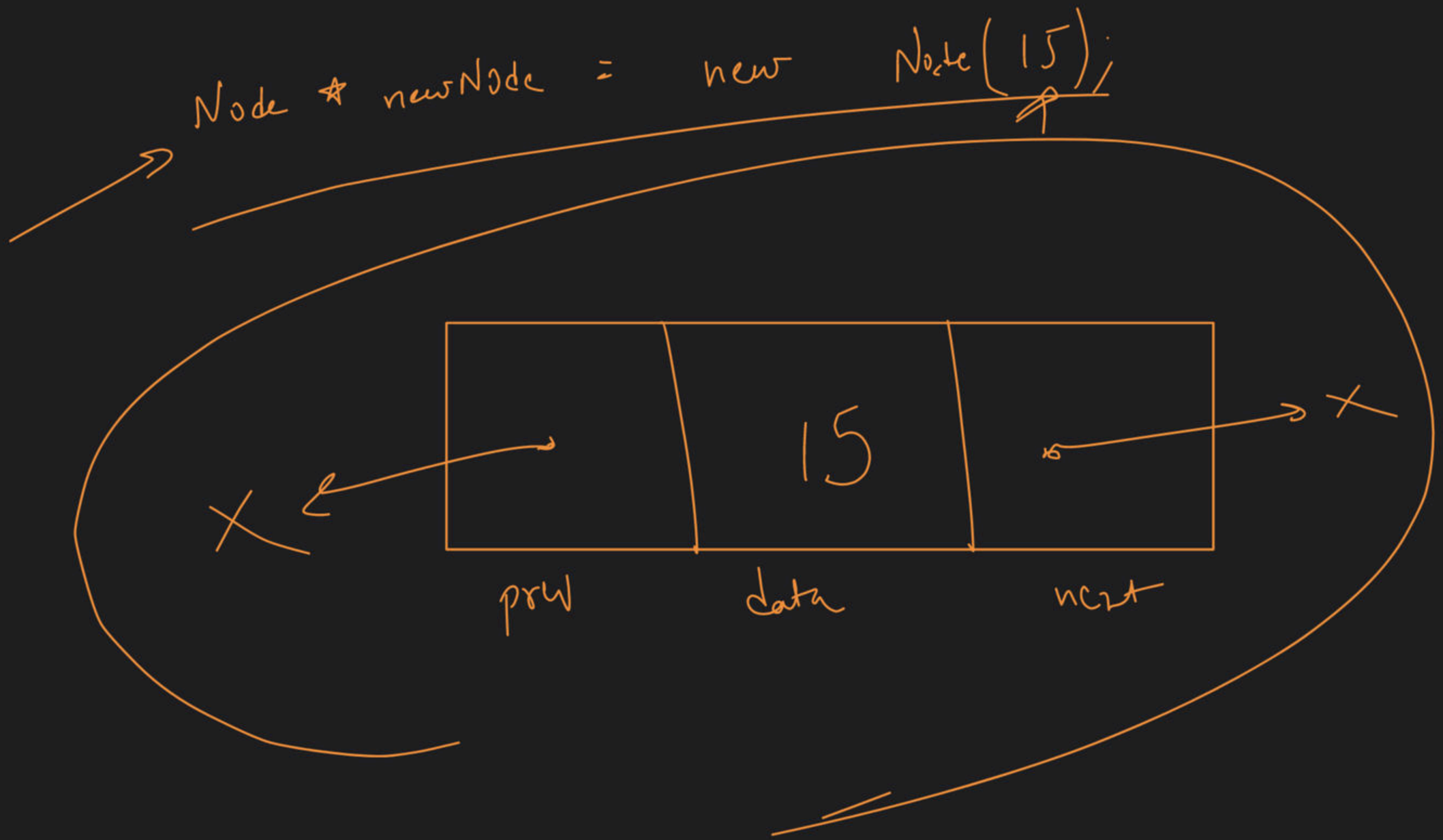
Doubly Linked List → diff -> node ->




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

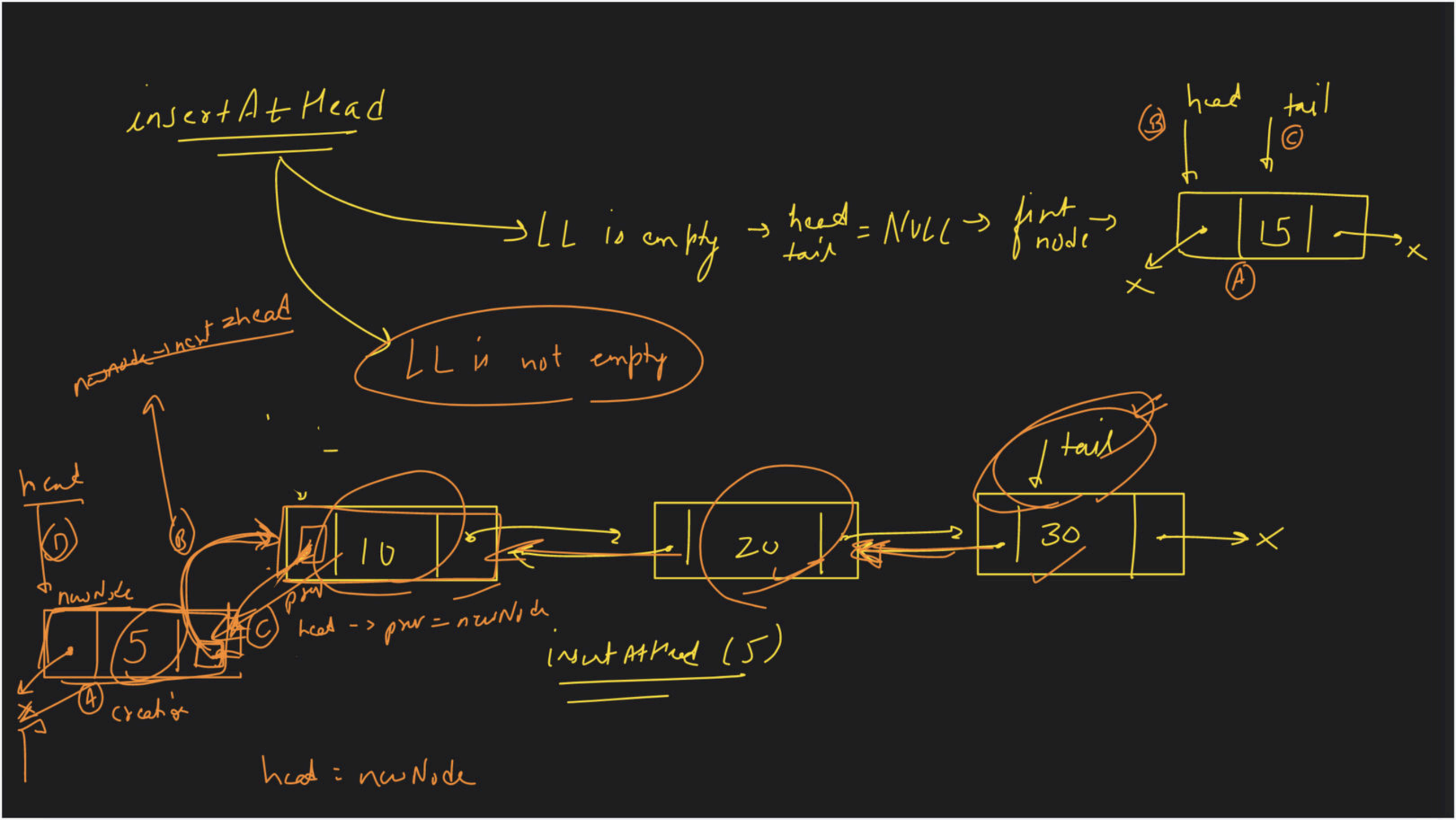
Node:-




Node * newNode = new Node(15);
15
prev
data
next

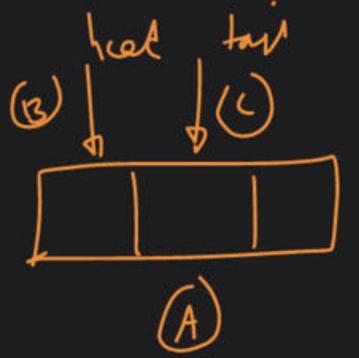
Insertion :-

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

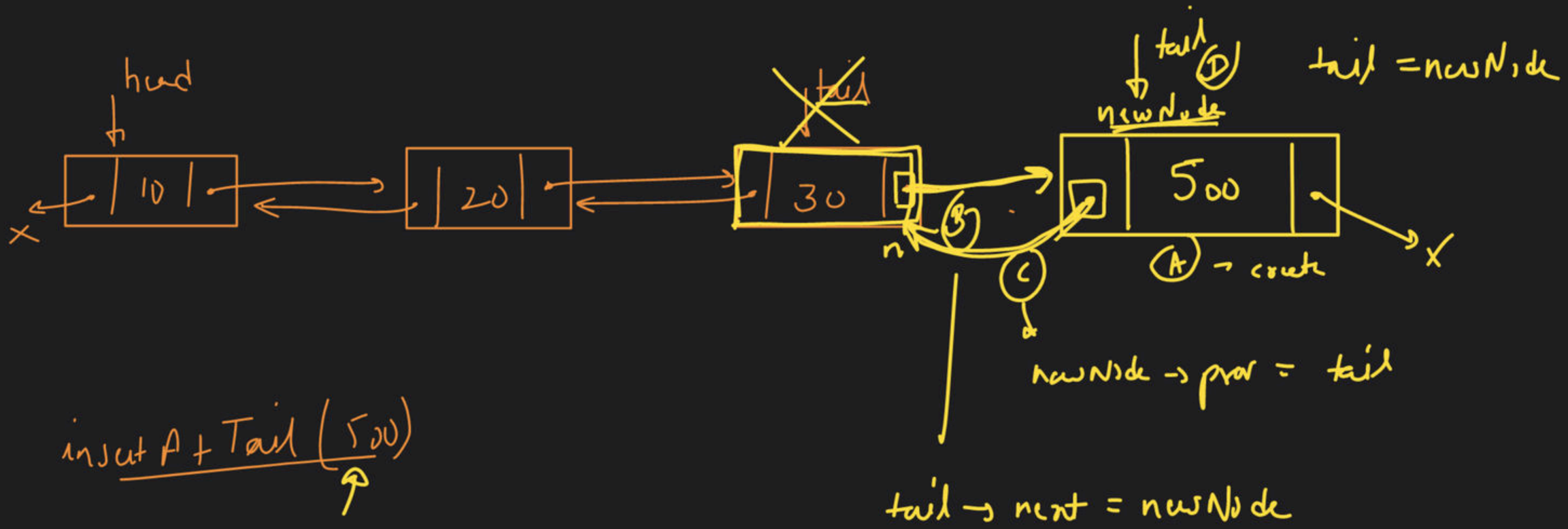

insertAtHead
LL is empty → head = tail = NULL → first node →
head
tail
15
LL is not empty
newNode = new zhead
head
10
20
30
tail
newNode
5
head -> prev = newNode
insertAtHead (5)
head = newNode

→ insertAtTail(.)

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

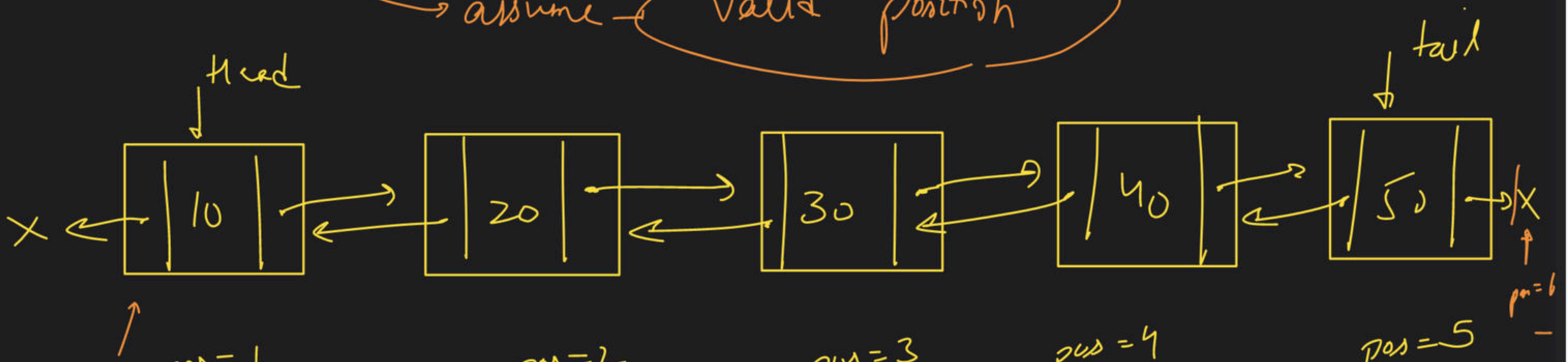


LL is not empty


head
10
20
30
tail
tail
newNode
D
tail = newNode
500
A
create
B
C
n
X
X
newNode -> prev = tail
tail -> next = newNode
insert At Tail (500)

→ insertAtPosition (value)

→ assume — valid position

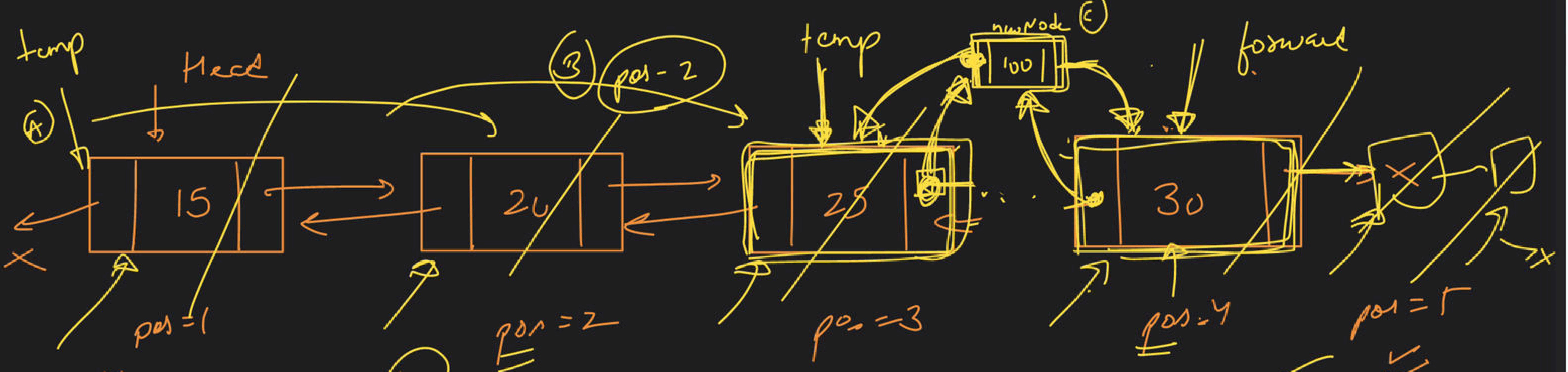


insertAtPos (1) → insertAtHead ( )

insertAtPos (6) → insertAtTail ( )

→ reuse

→ logic



0 1 2 3 4 5 6

newNode -> next -> prev = newNode

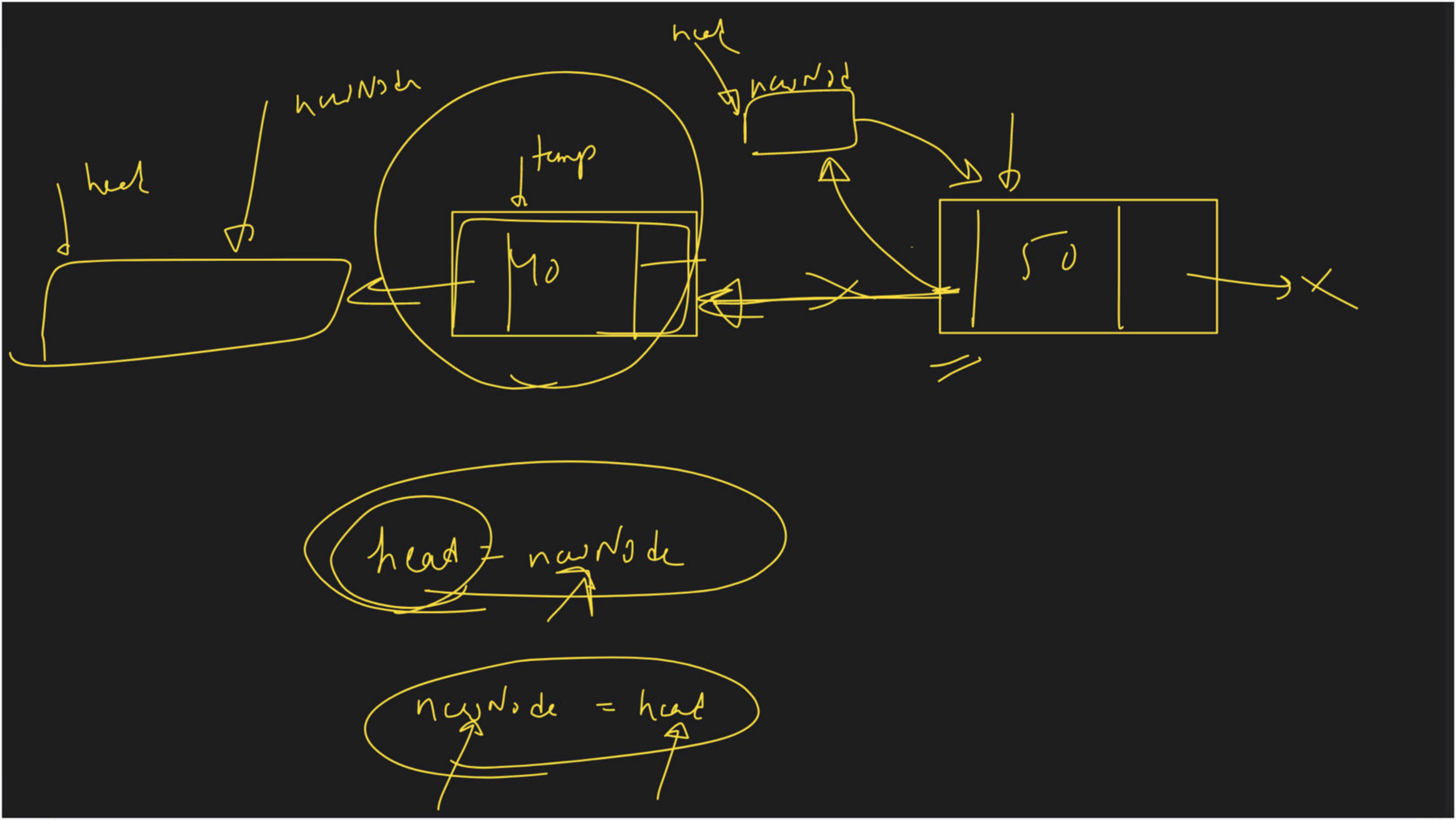
insertAtPos(4)

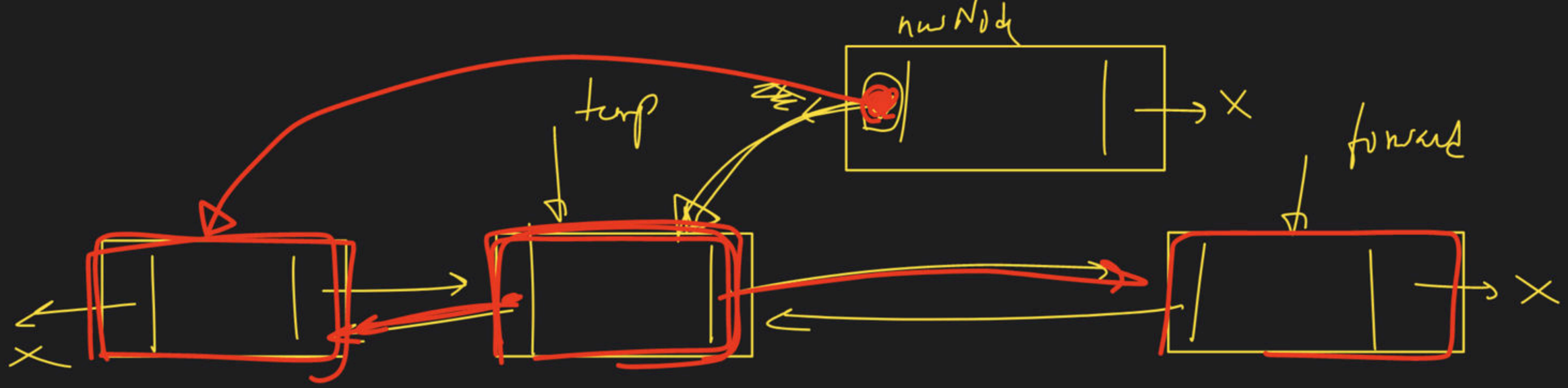
4 pos -> 2 st
5 pos -> 3 st
6 pos = 4 st
7 pos = 5 st
8 pos = 6 st

-> n pos -> (n - 2) step

1. temp = head
2. temp -> pos - 2 st
3. create node
4. forward = temp -> next

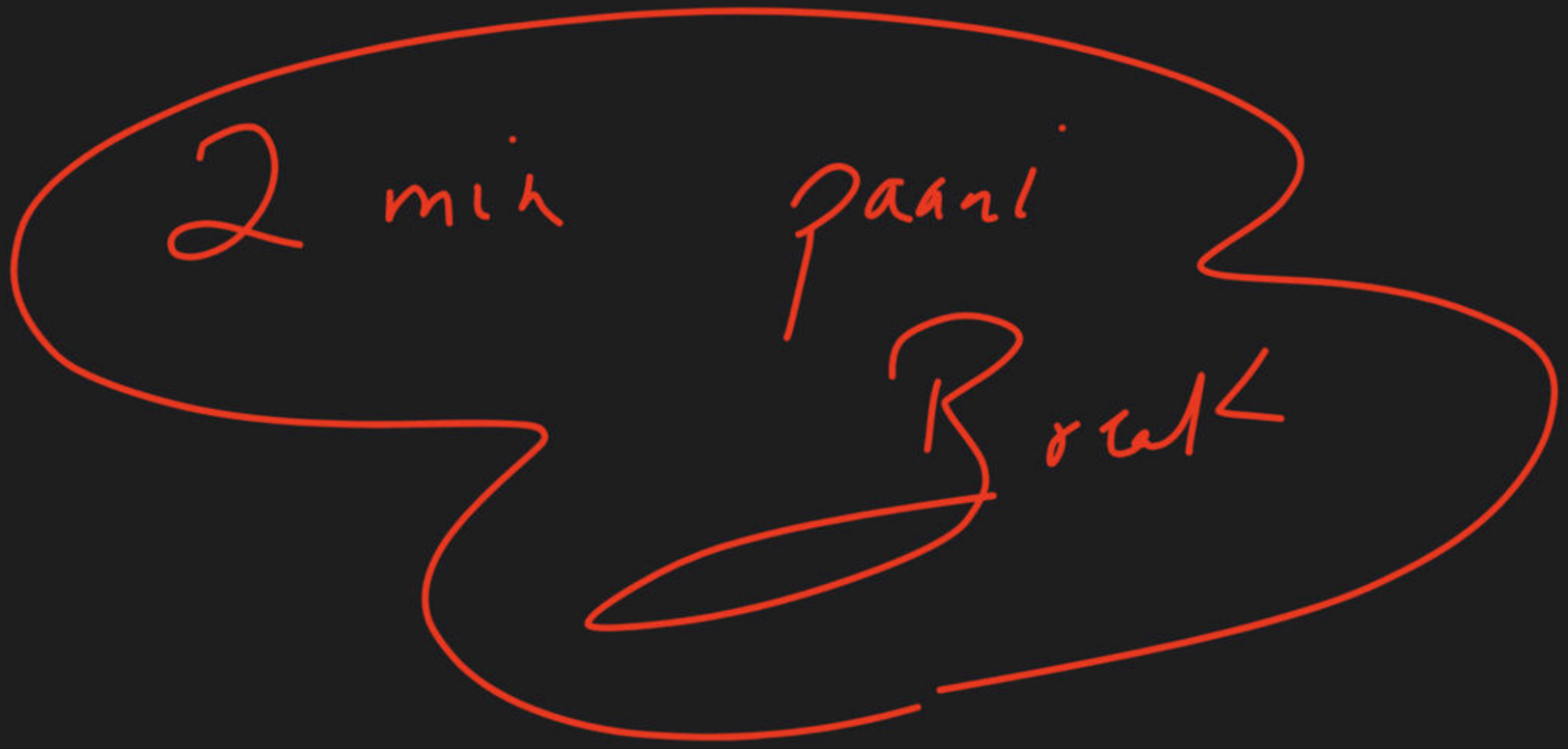
temp -> next = newNode
newNode -> prev = temp
newNode -> next = forward
forward -> prev = newNode

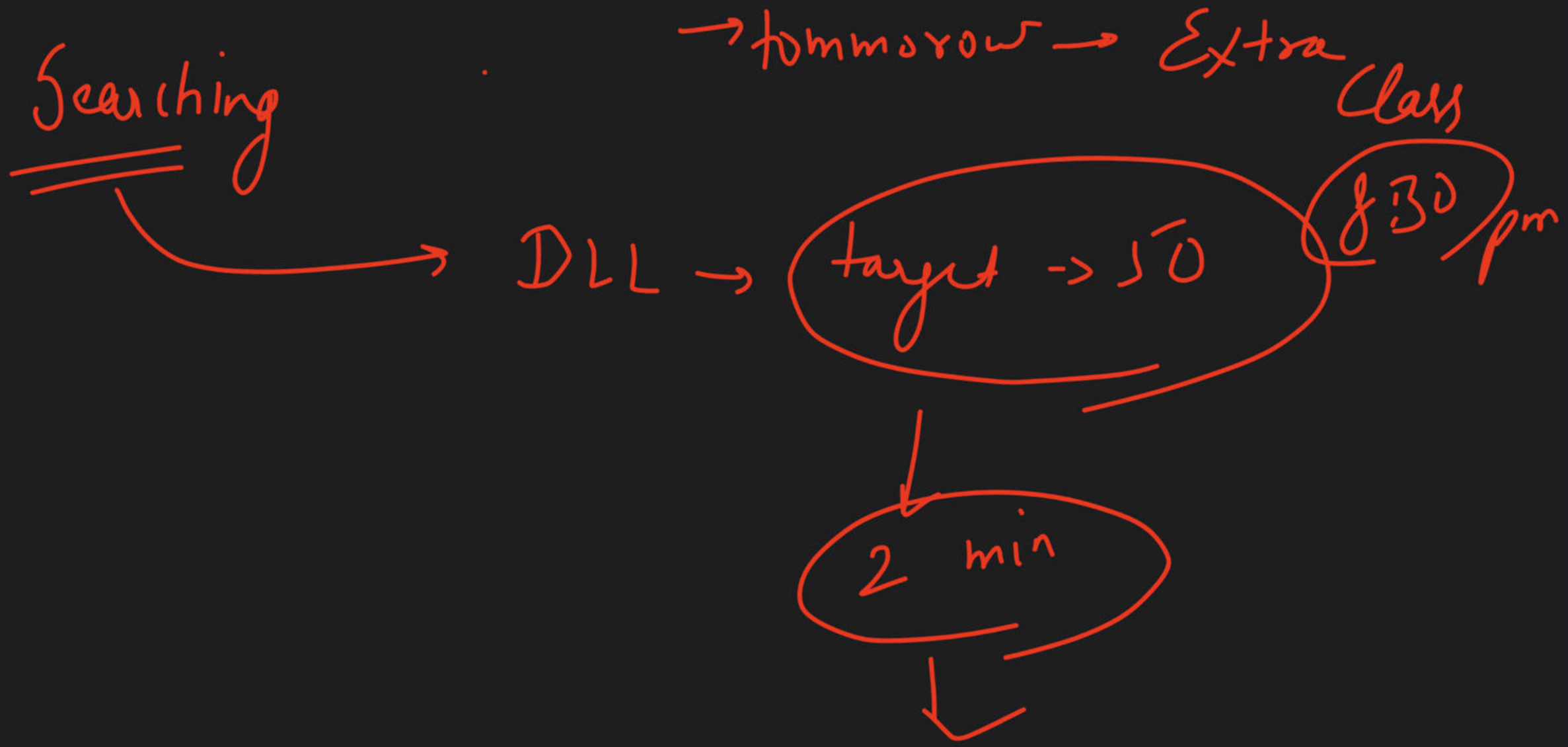

head
newNode
head
newNode
temp
10
50
X
head = newNode
newNode = head

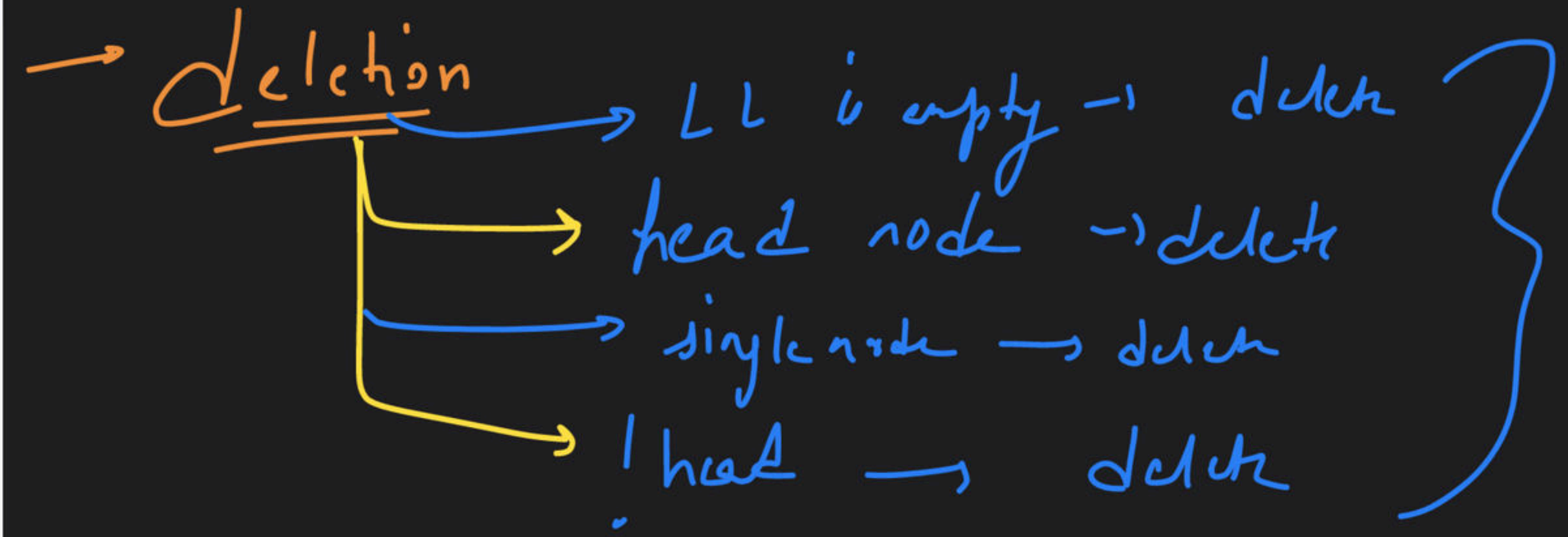


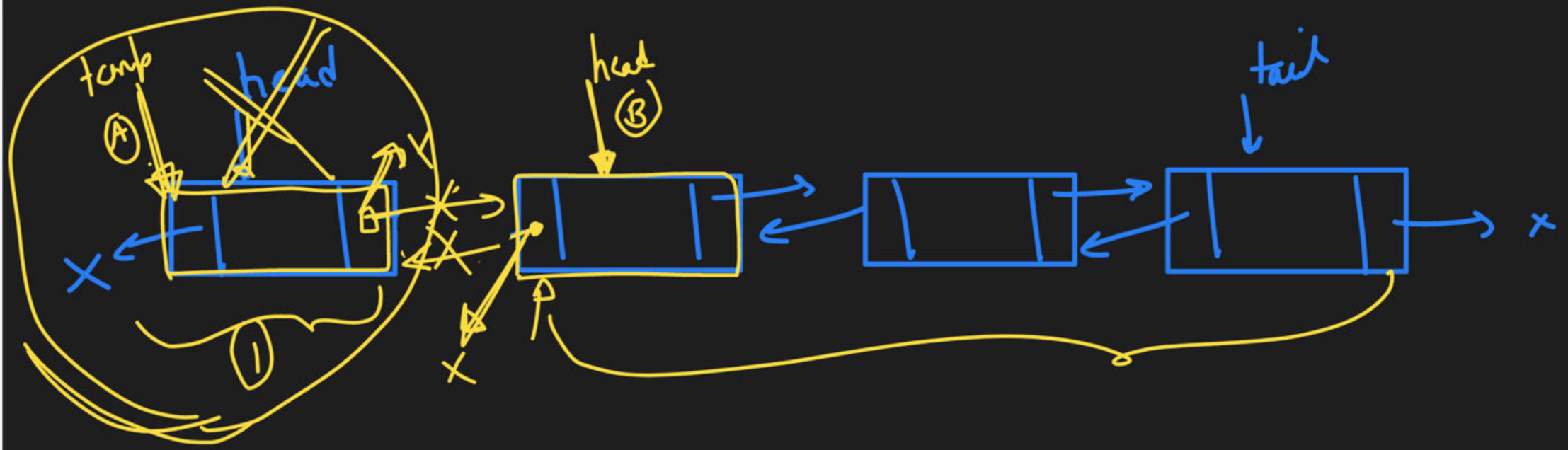
newNode -> prev = temp

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

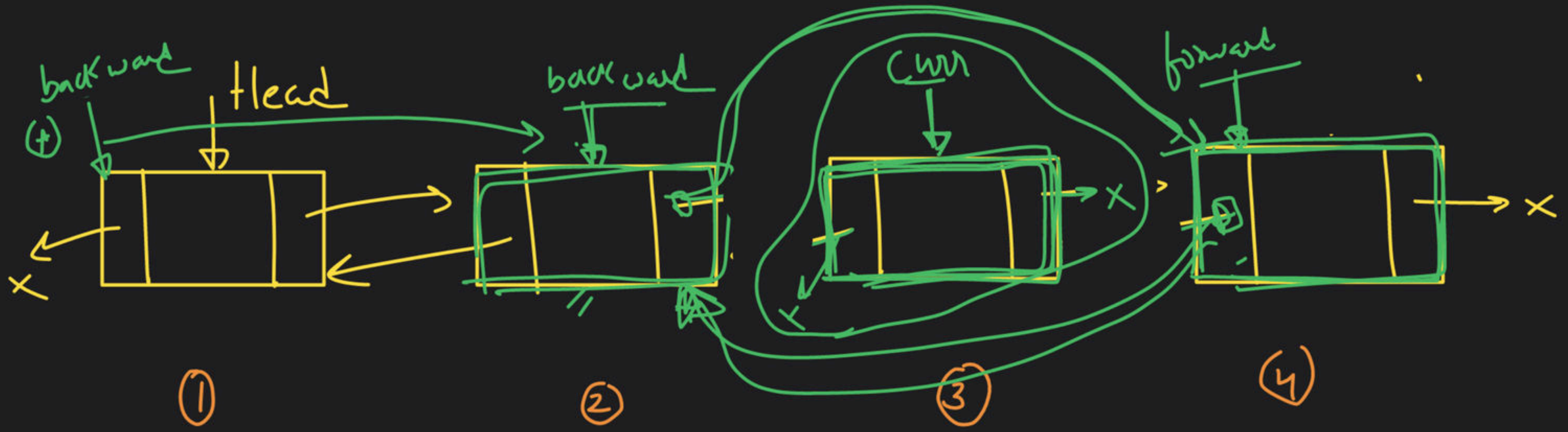
(B) head = head -> next

(C) head -> prev = NULL

(D) temp -> next = NULL

(E) delete temp

delete from list (1)



(A) backward = head

(B) backward -> pos - 2 -> step

(C) curr = backward -> next

(D) forward = curr -> next

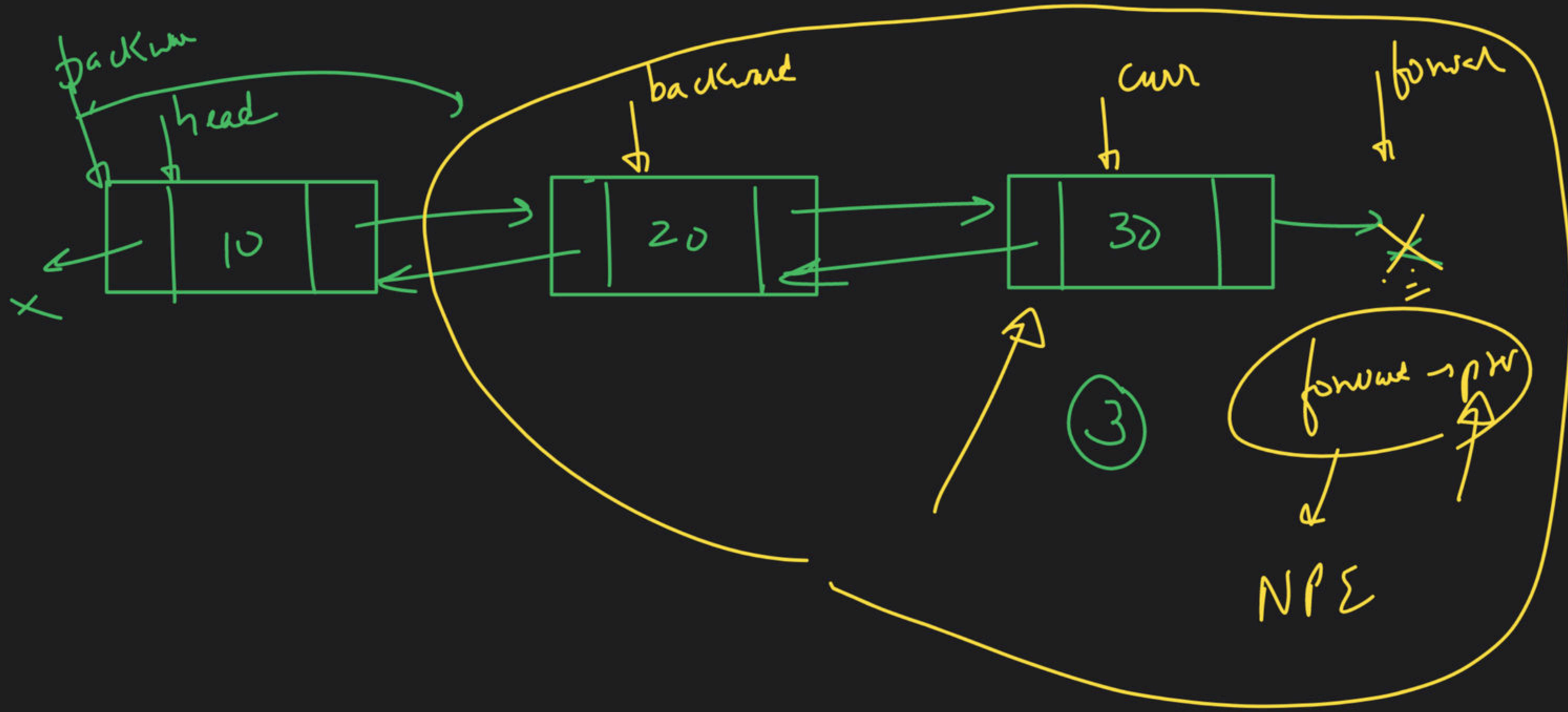
backward -> next = forward

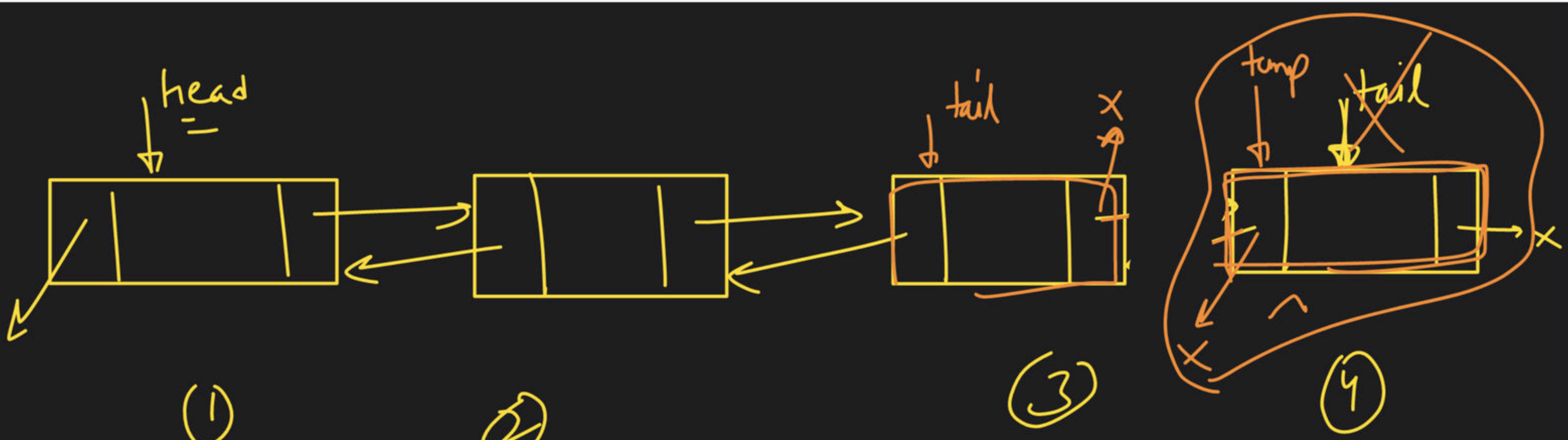
forward -> prev = backward

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

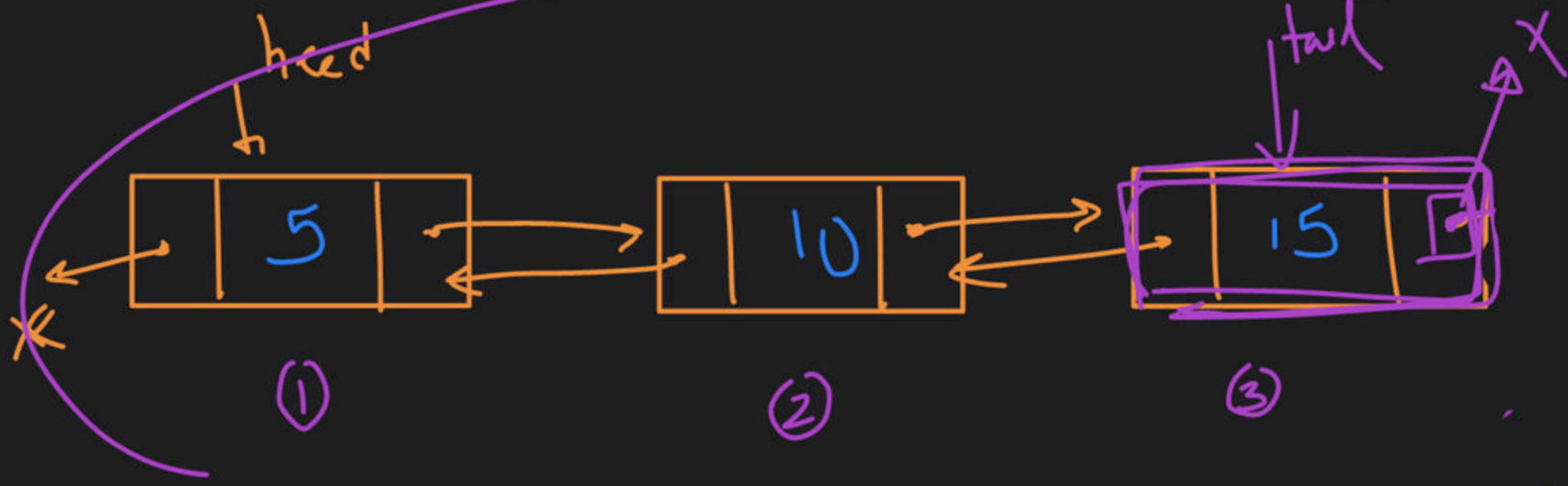
(D) temp → prev = NULL

(E) delete temp

deleteFromPos (4)

position == length

tail delete



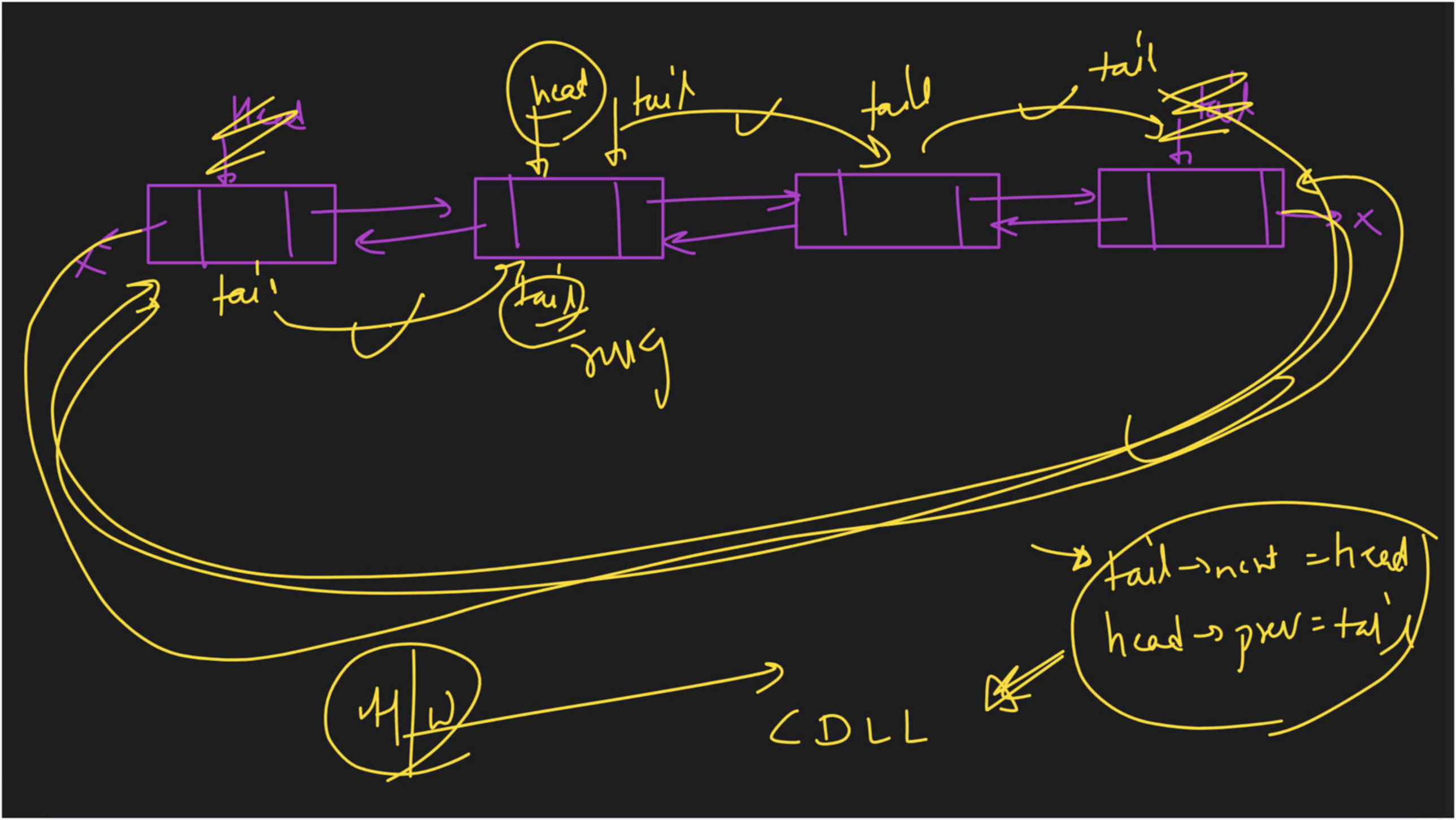
SLL

DLL

CDLL

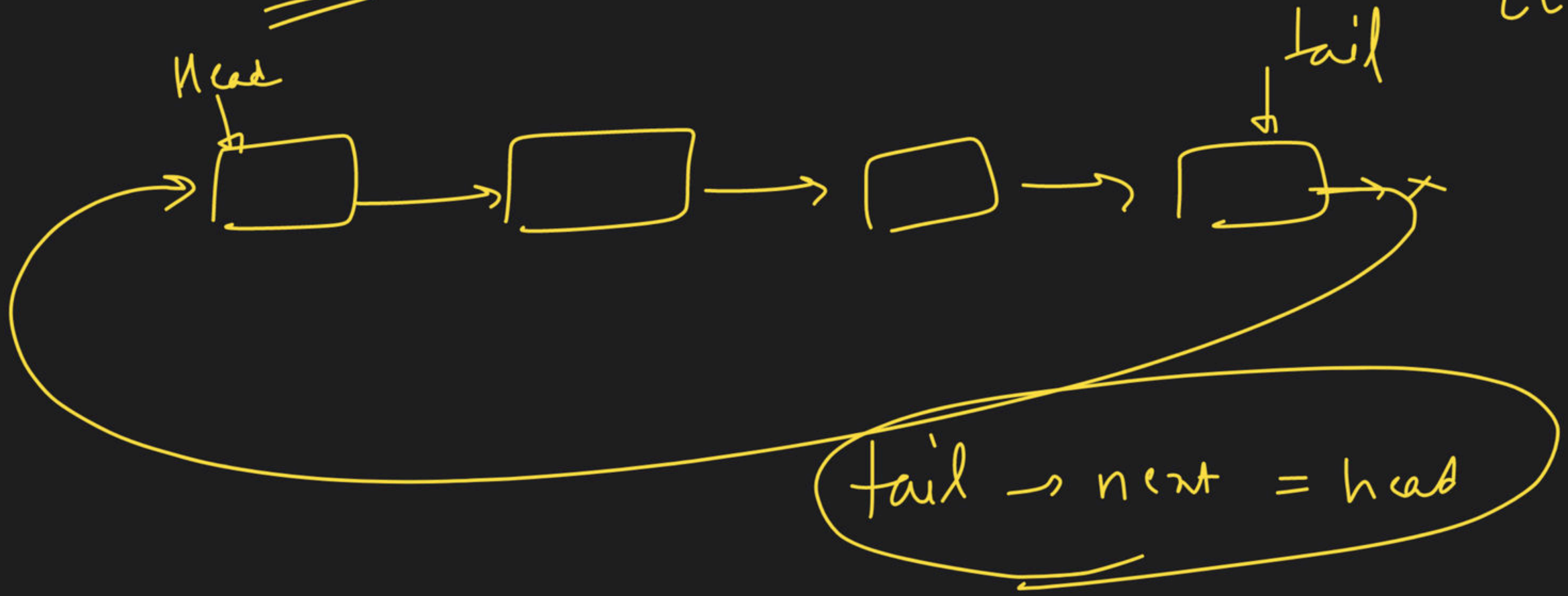
```
Node * temp
tail = temp -> prev
temp -> prev = NULL
tail -> next = NULL
delete temp
```

deleteFirst(4)


head
tail
tail
tail
tail
tail
tail
x
x
tail->next = head
head->prev = tail
CDLL

→ C·L·L

KHUD → length of CLL



S·L·L → C·L·L