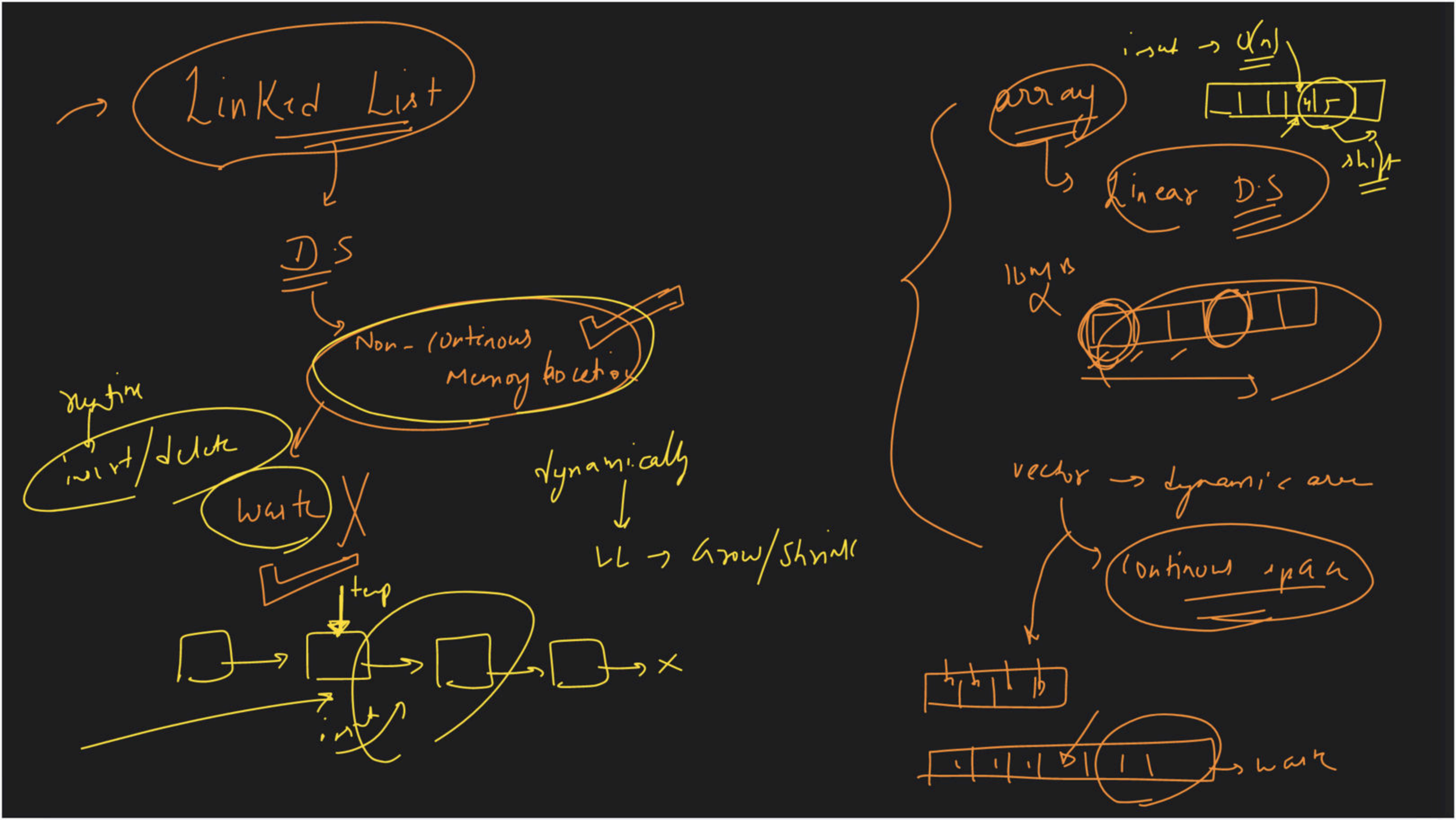
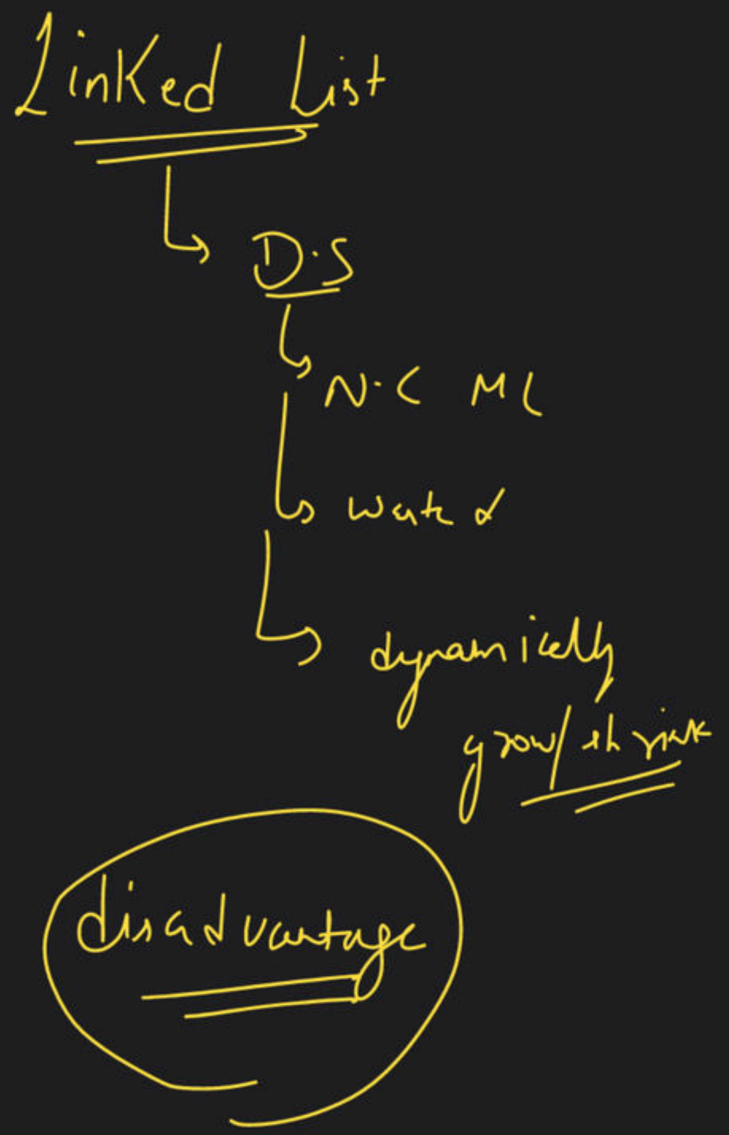
LL Class - 1

Special class

Love Babbar • Apr 8, 2023


LinKed List
D:S
Non-continous
Memory location
insert/delete
waste
dynamically
LL → Grow/Shrink
temp
insert
array
insert → O(n)
shift
Linear D:S
Vector → dynamic array
continous space
waste

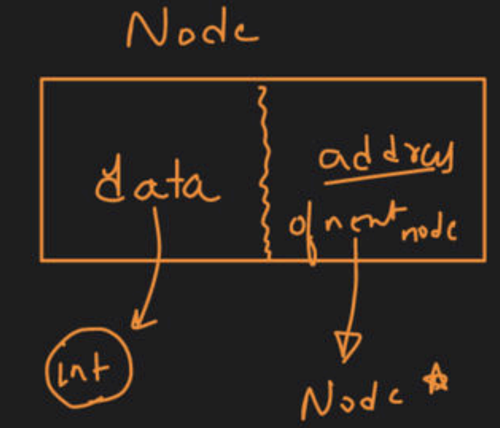

Linked List
D.S
N.C ML
waste of
dynamically
grow/shrink
disadvantage

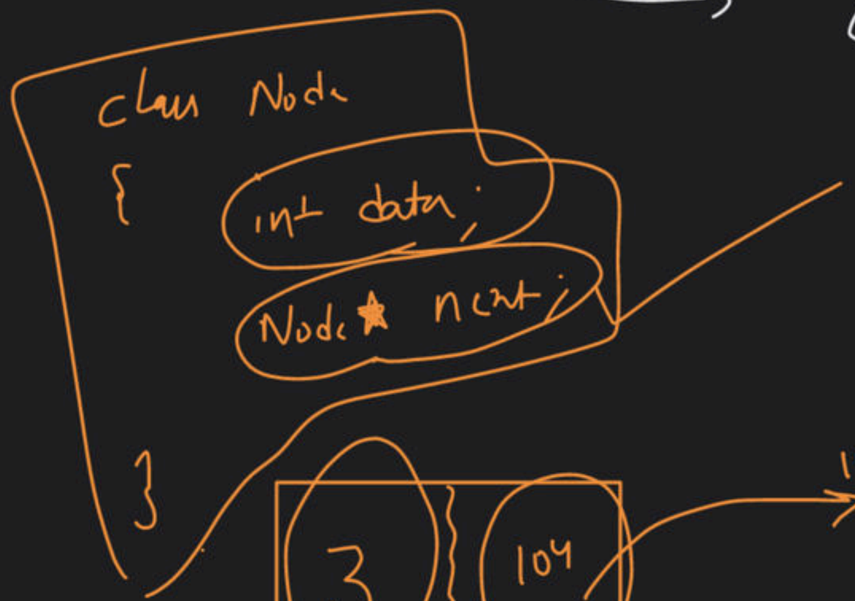
Linked List

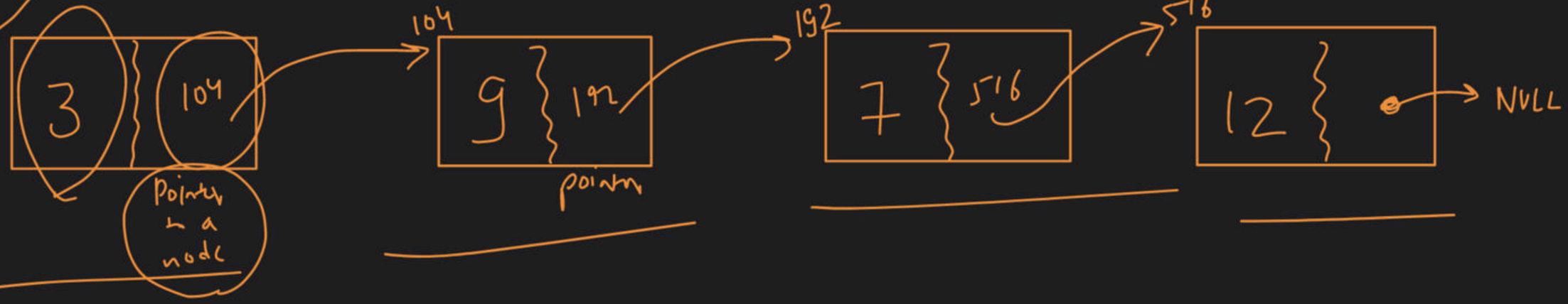
pointer to integer → int*

pointer to Node → Node*

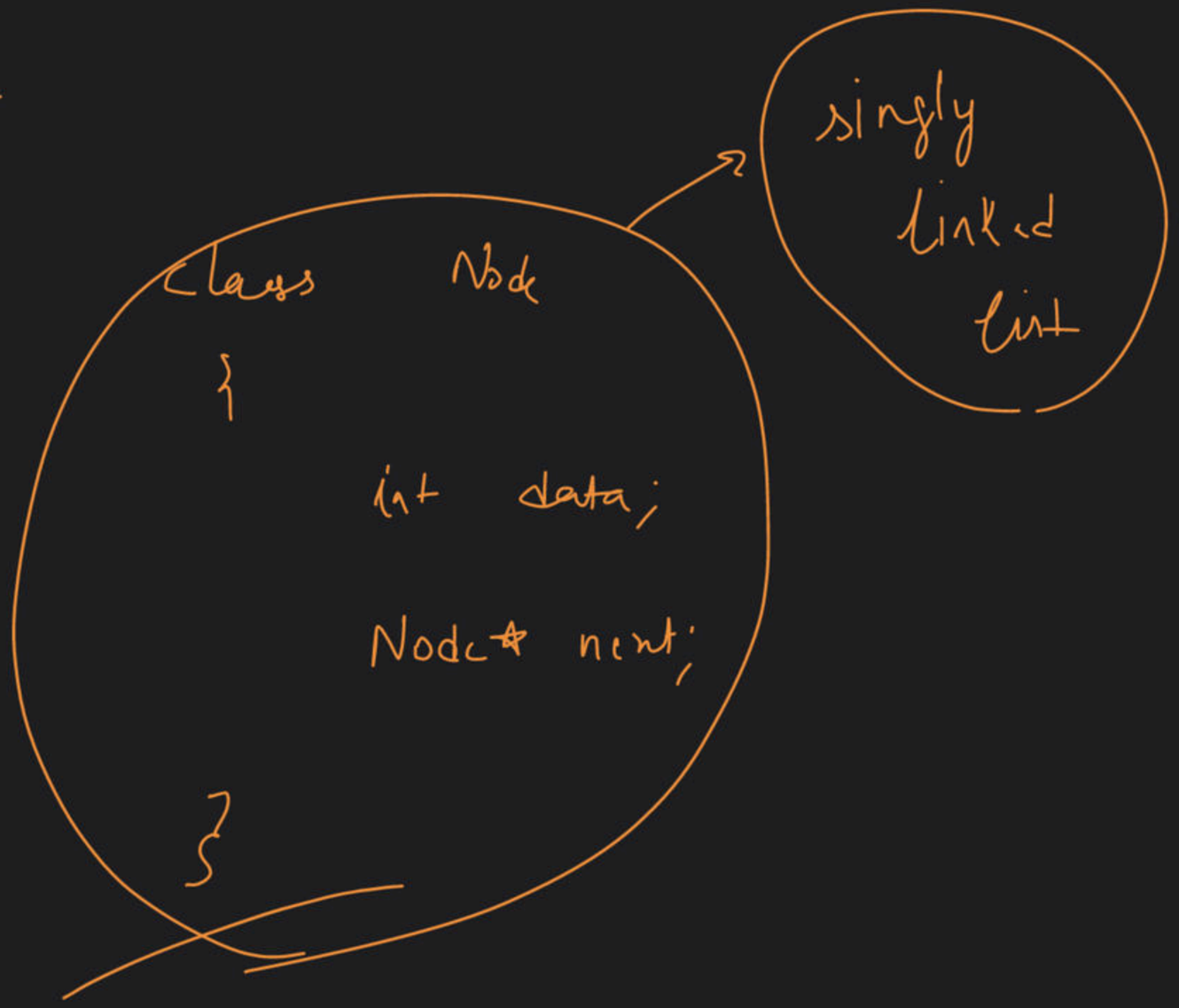
Collection of Nodes →

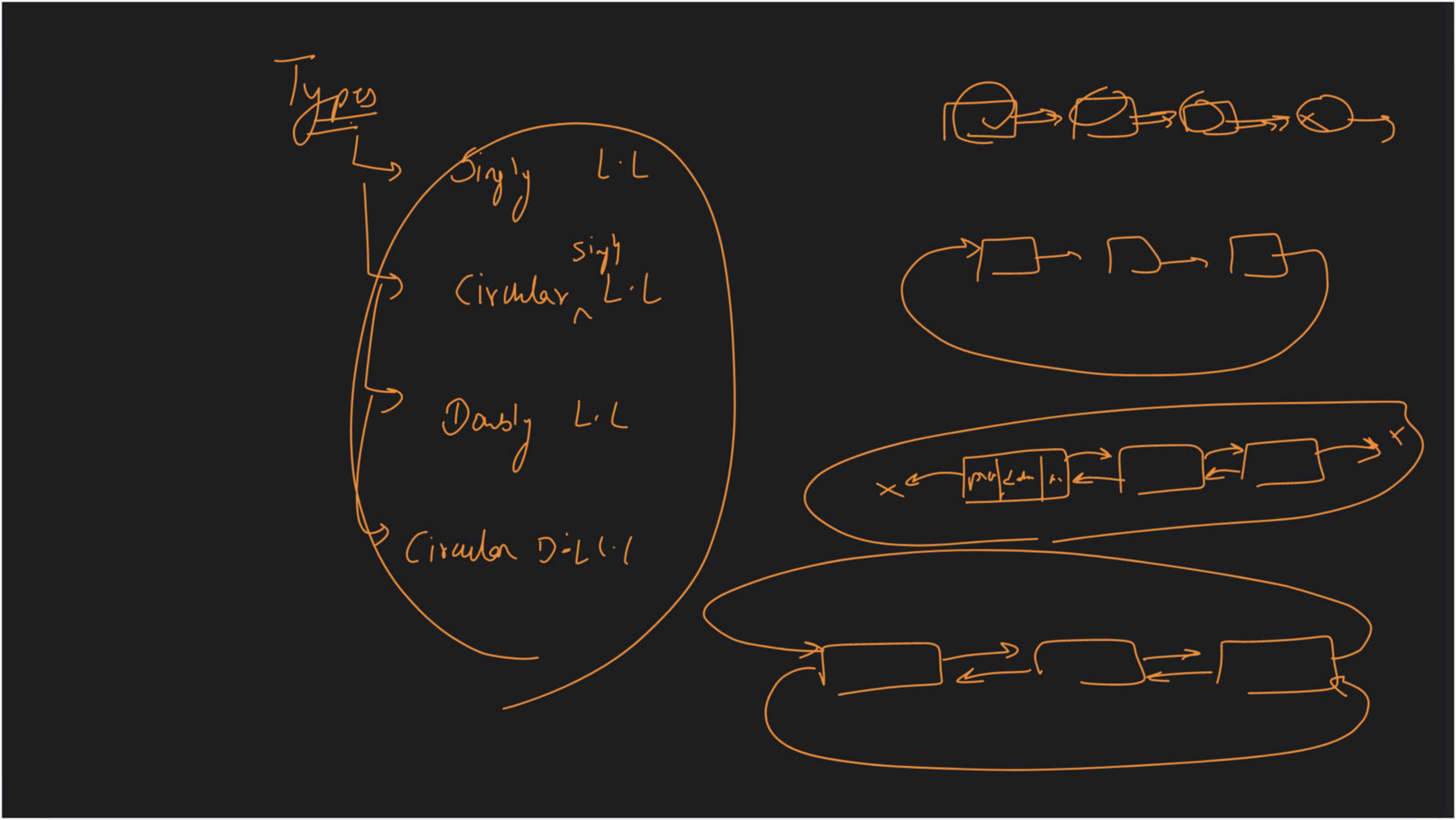


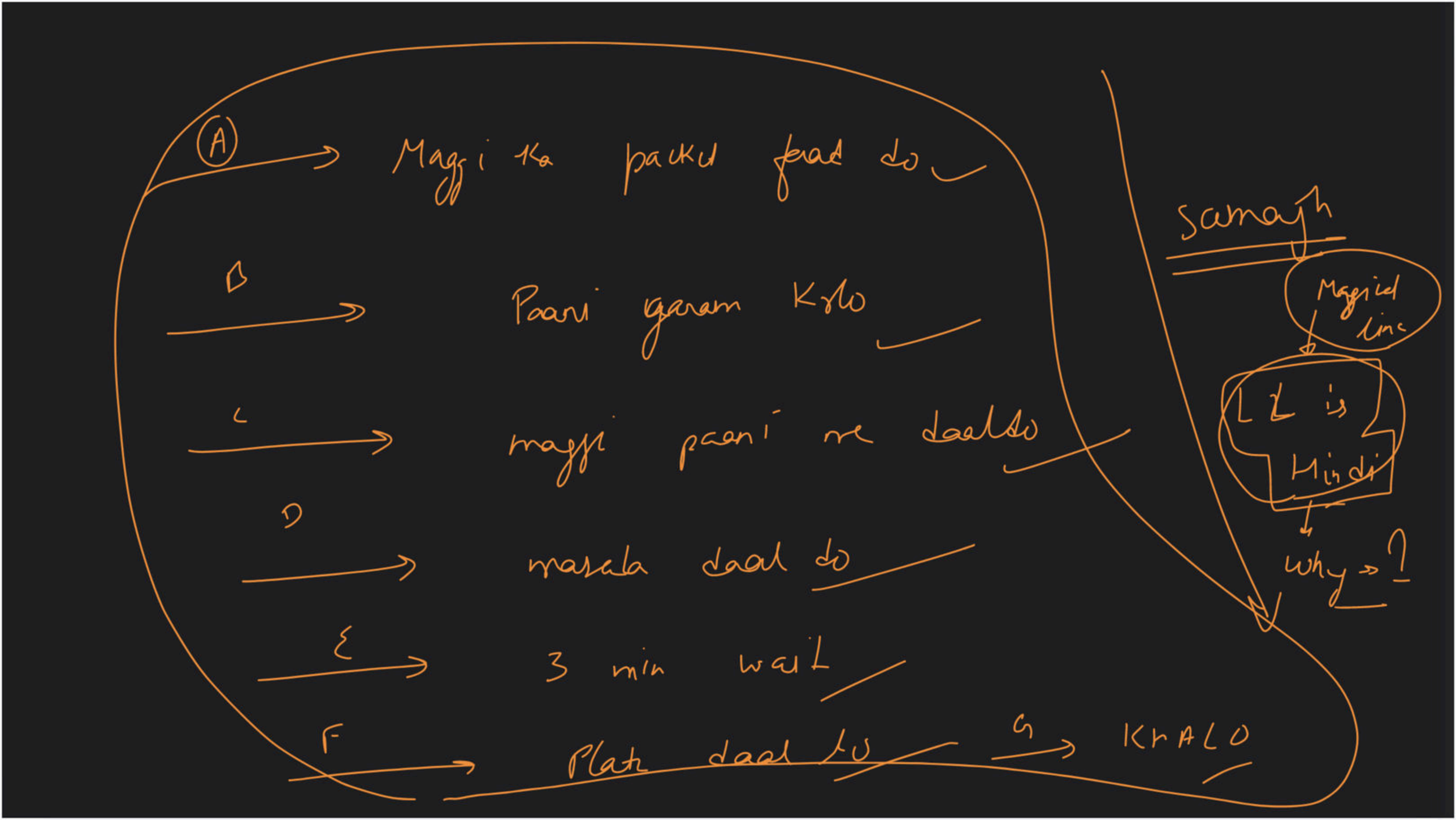


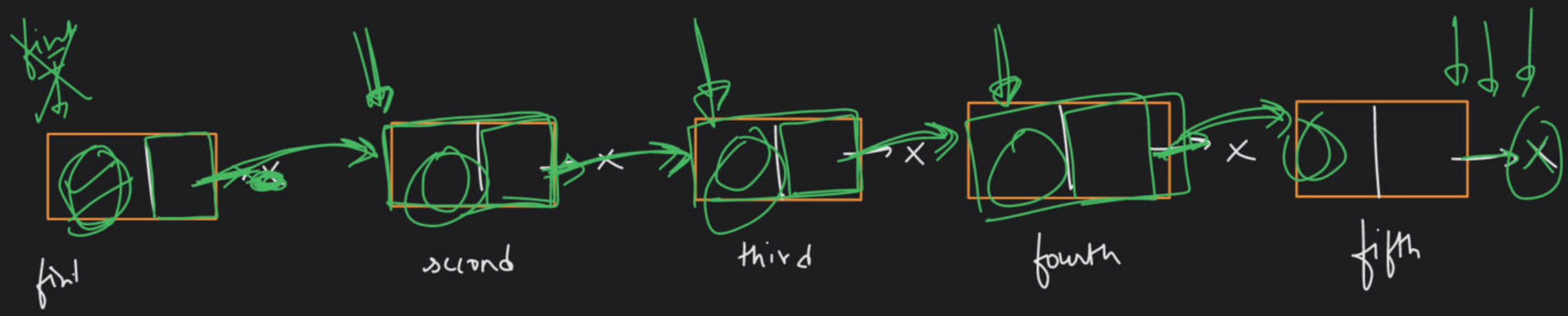


Node




Types
Singly L.L
Circular L.L
Doubly L.L
Circular D.L.L


A
Maggi ka packet phaad do
B
Paani garam karlo
C
maggi paani me daalo
D
masala daal do
E
3 min wait
F
Plate daal lo
G
KHALO
samajh
Magical line
L L is Hindi
Why ??



first → next = second

second → next = third

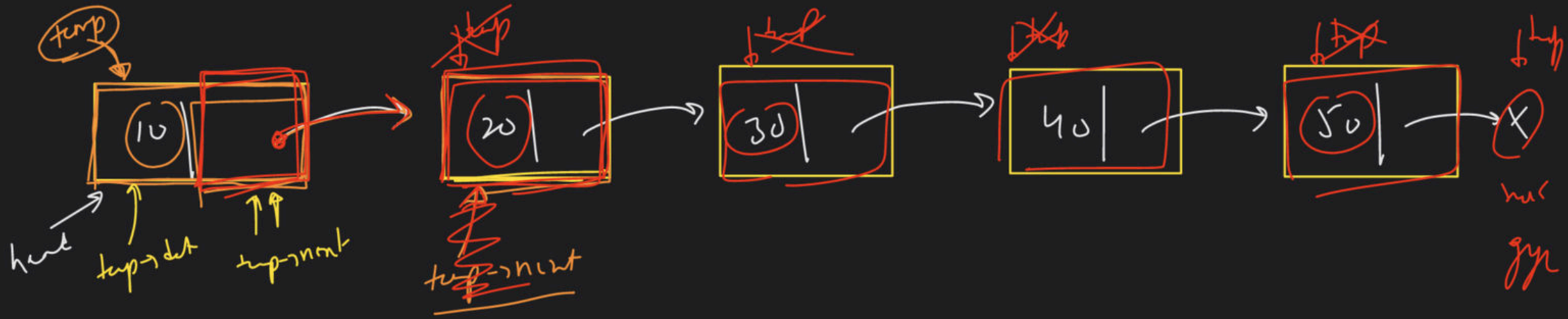
third → next = four

fourth → next = fifth

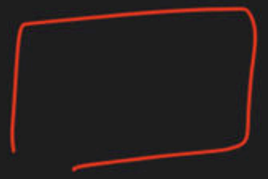
① print karlo data

② pointer ko aage lagao

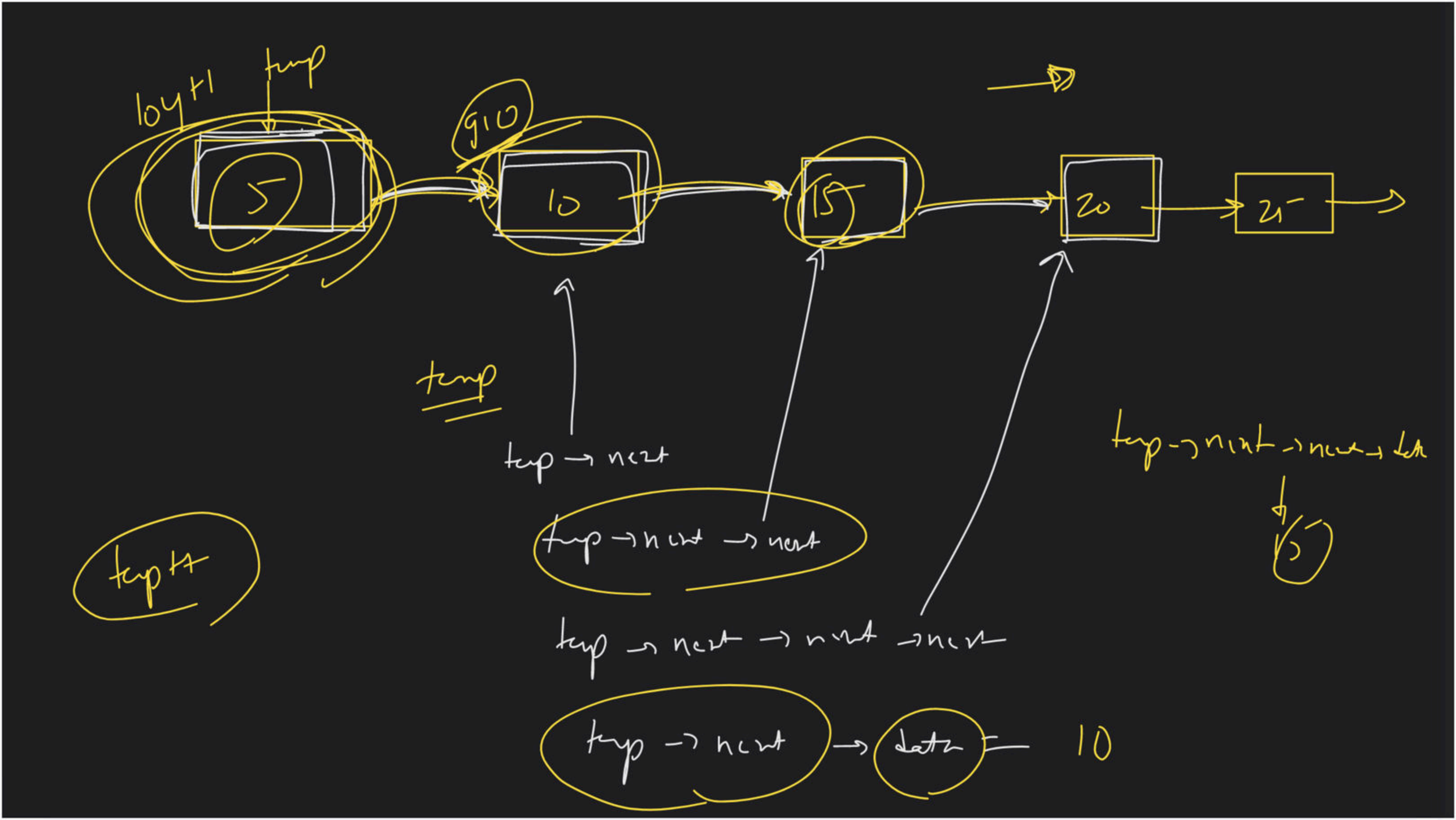
③ rukna kab h
↳ jb pointer null ho

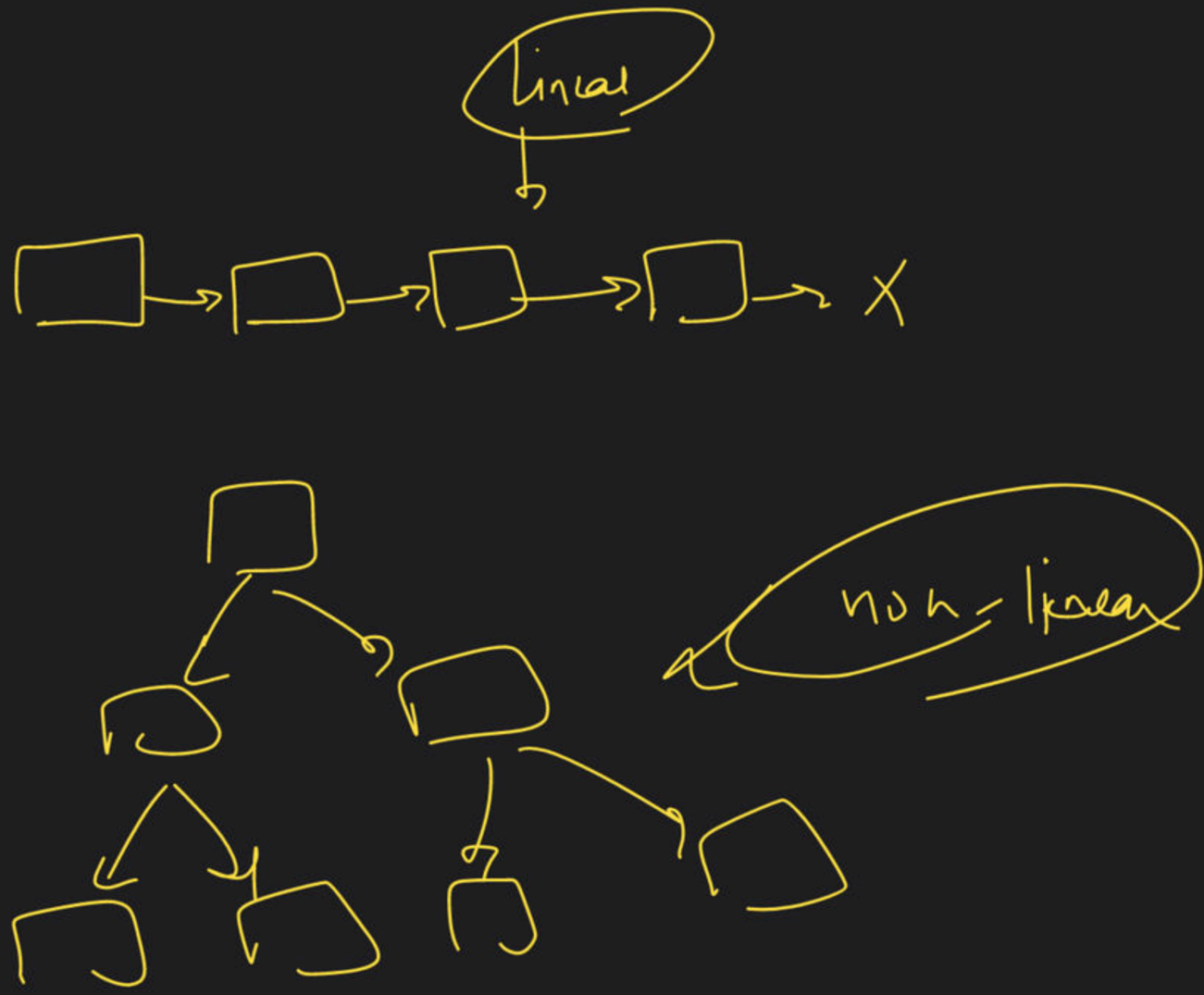


10 20 30 40 50



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


temp
5
10
15
20
25
temp
temp -> next
temp -> next -> next
temp -> next -> next -> next -> data
5
temp
temp -> next -> next -> next -> next
temp -> next
data
10

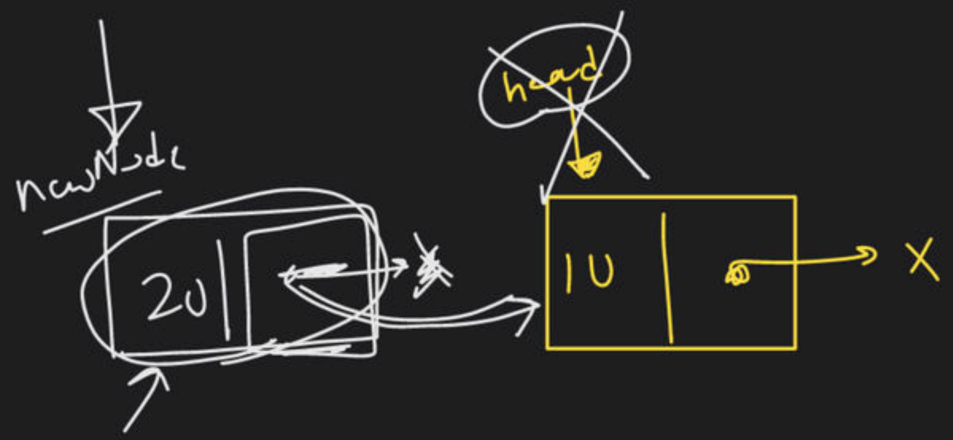

linear
X
non-linear

Node * head = new Node(10)

LL

Otherwise

Insert

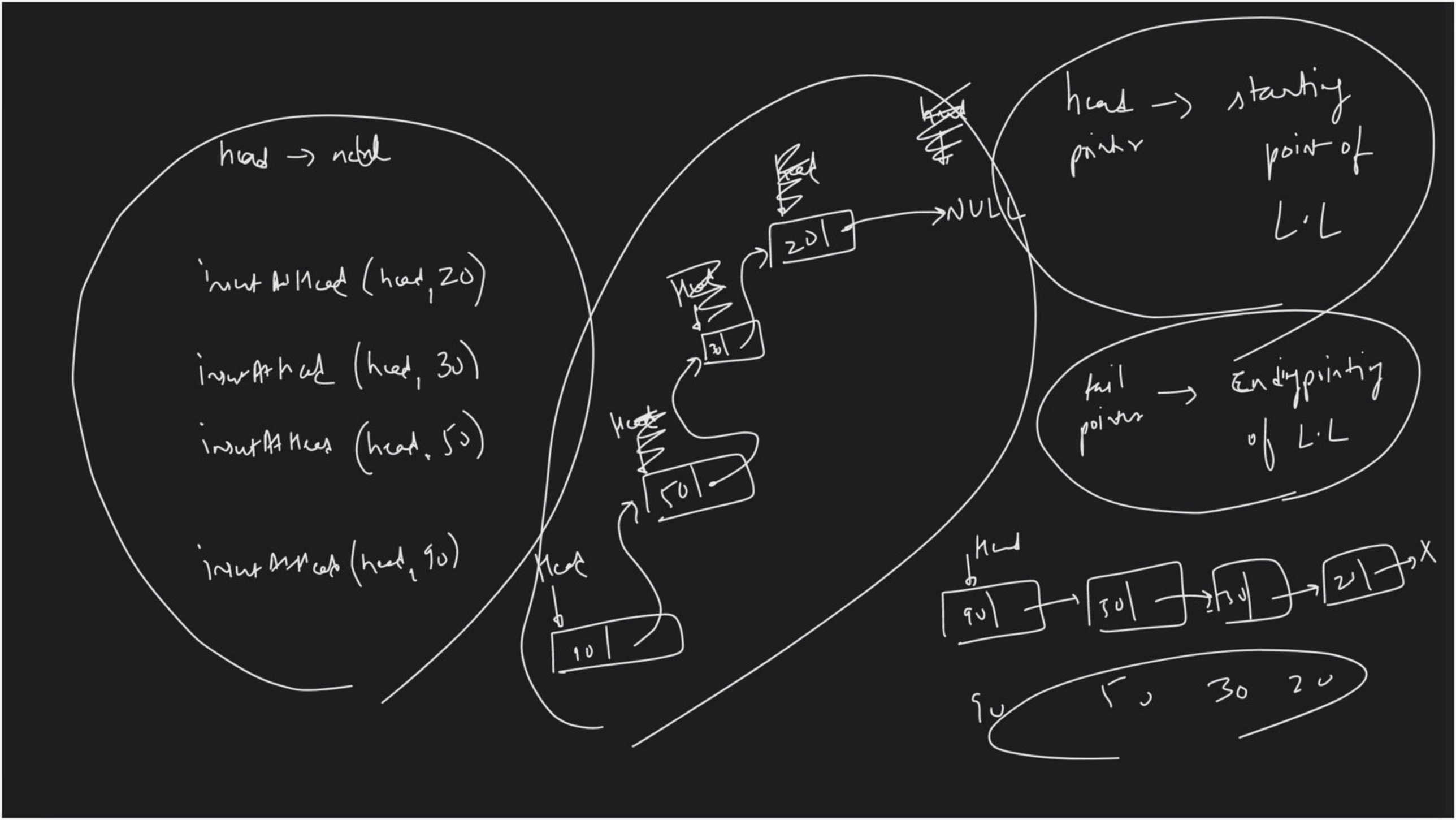


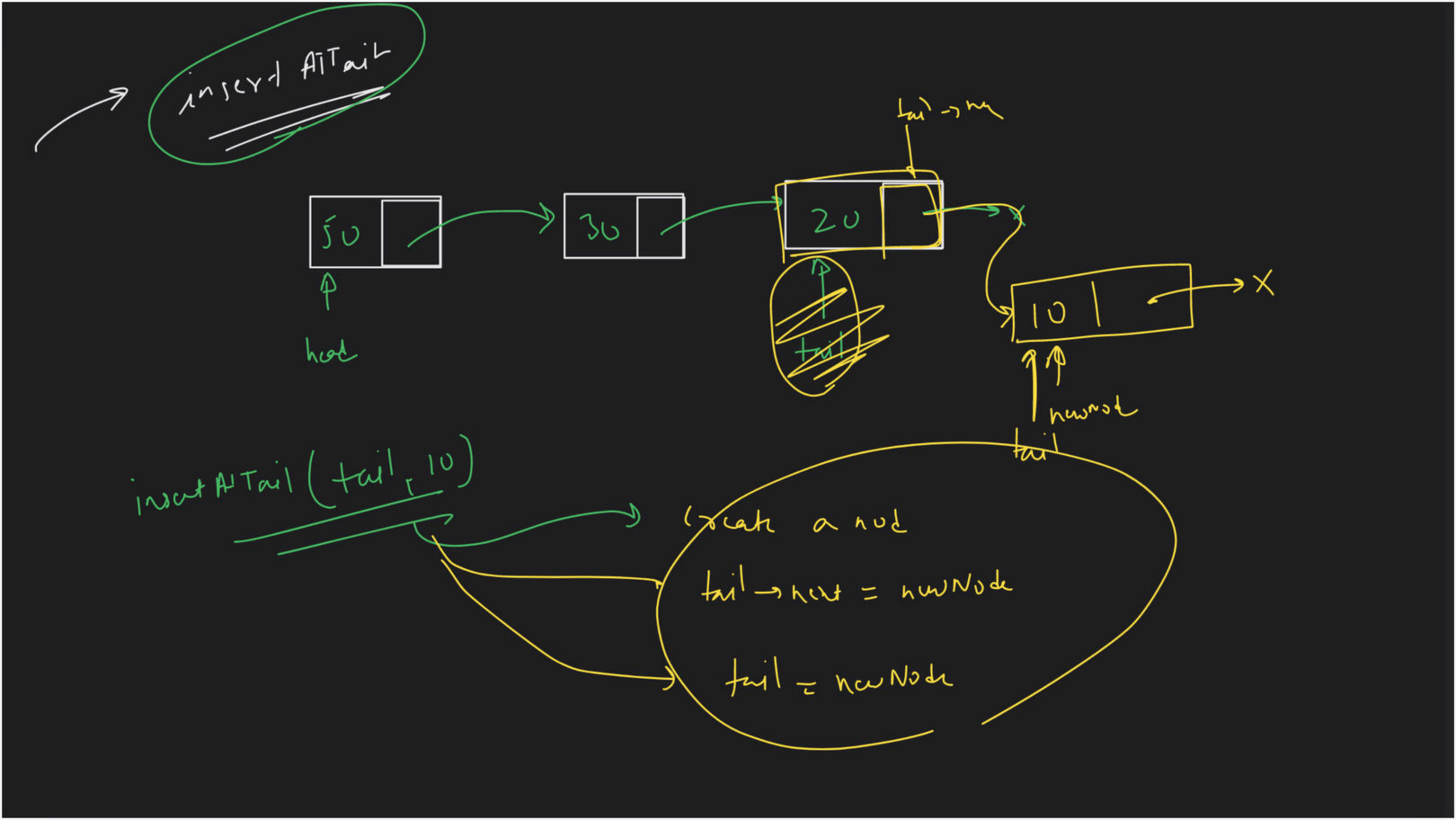
insertAtHead (head , 20)

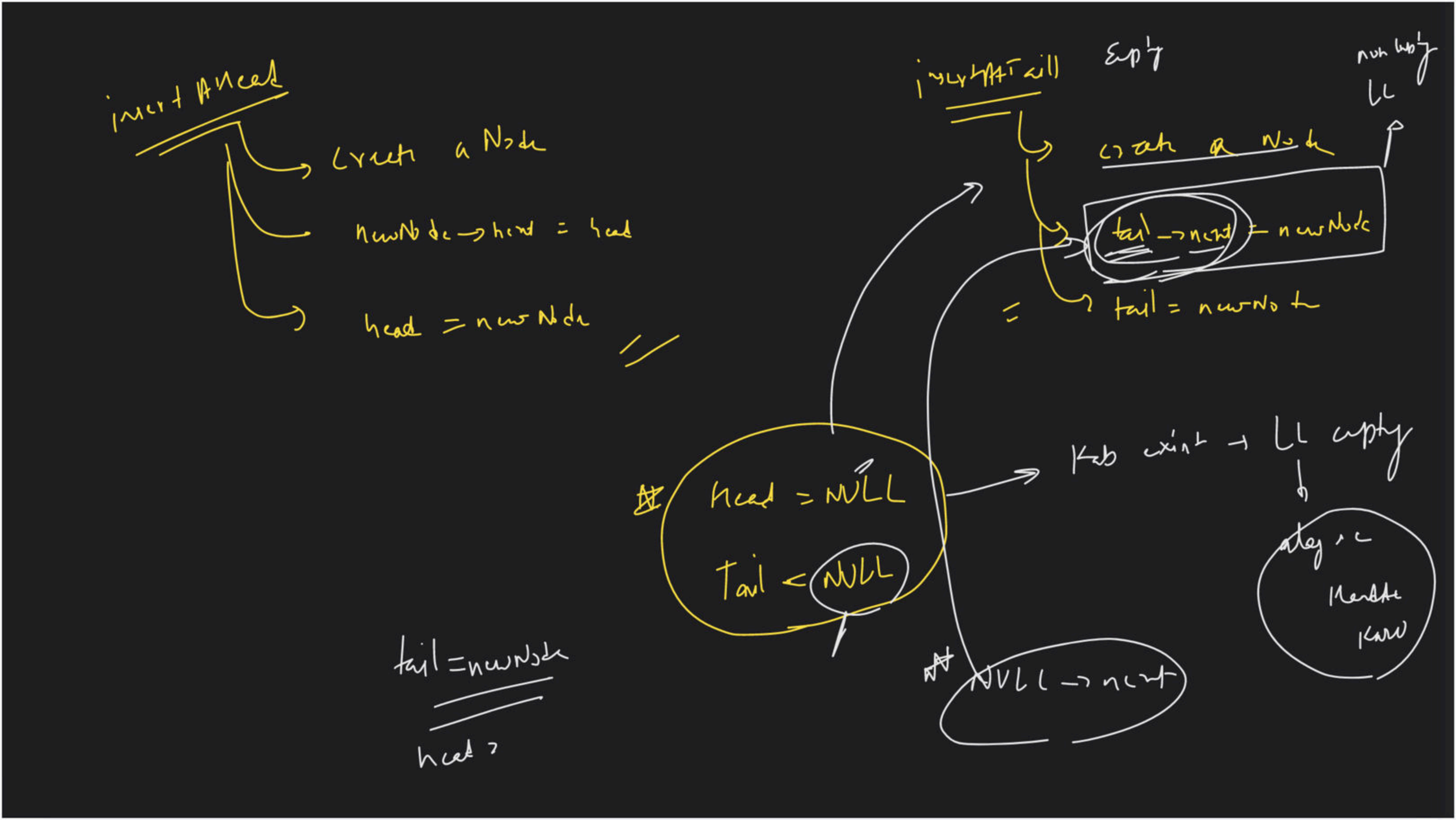
(A) Create a Node

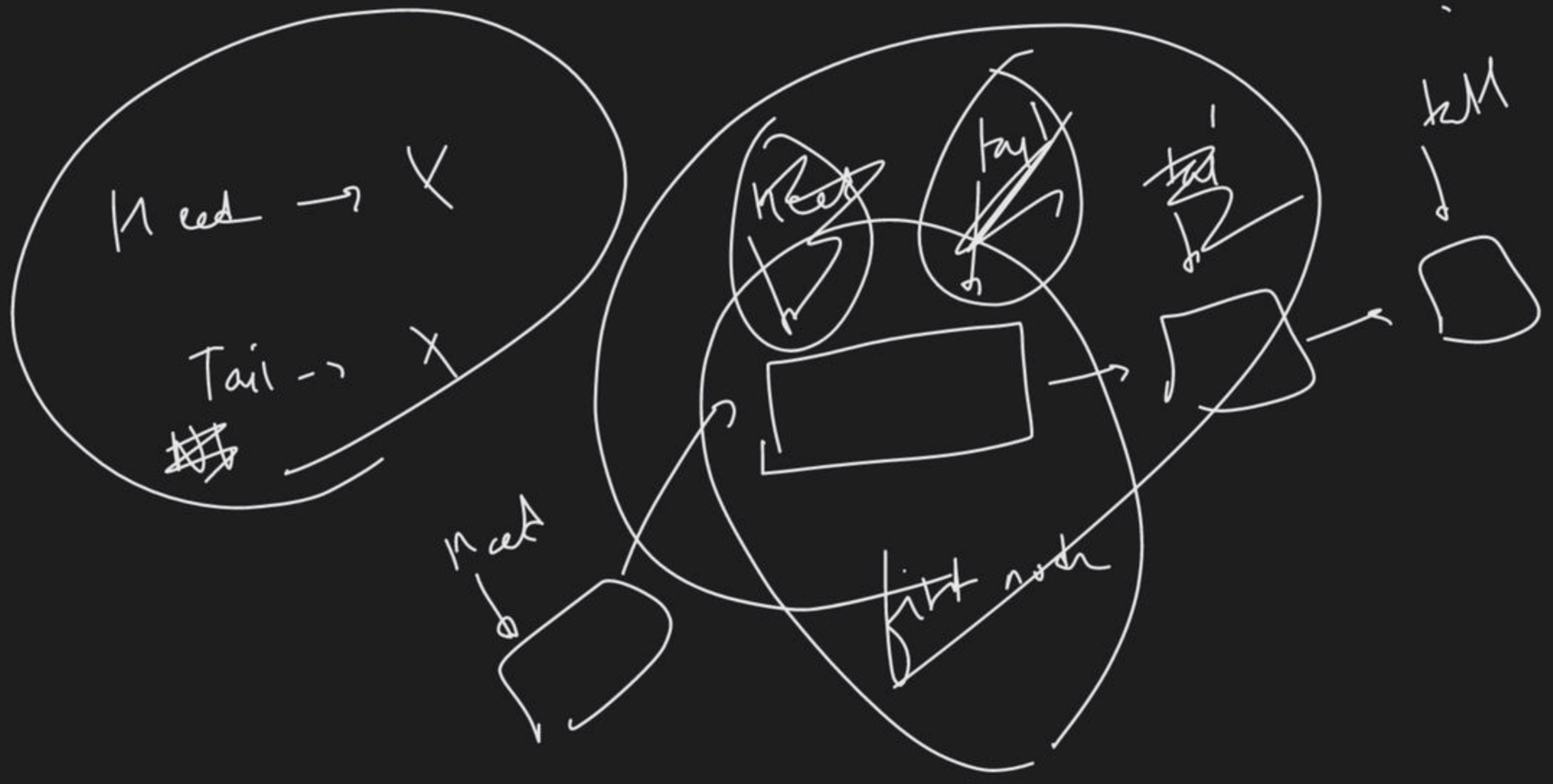
(B) newNode → next = head

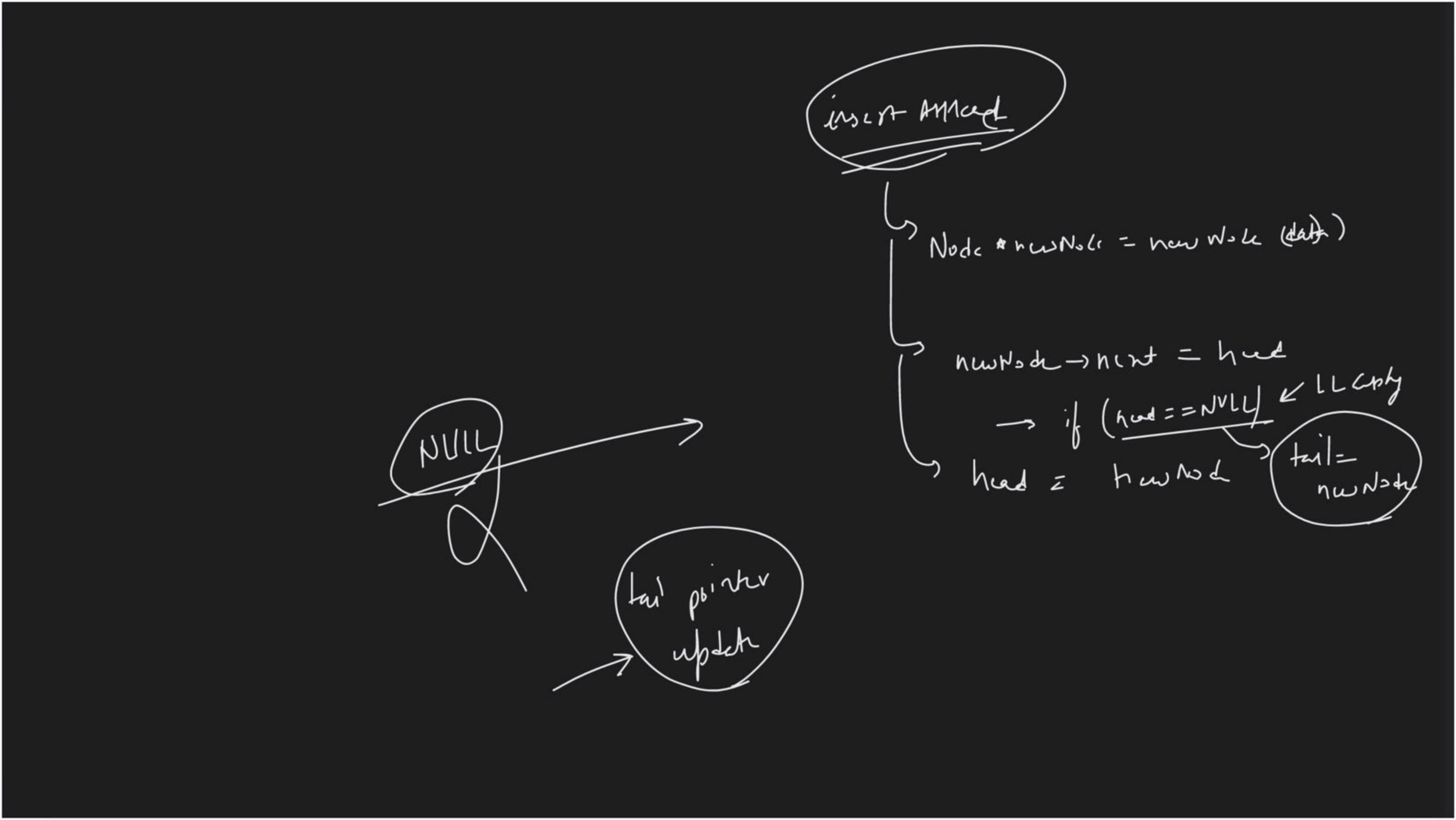
(C) head = newNode

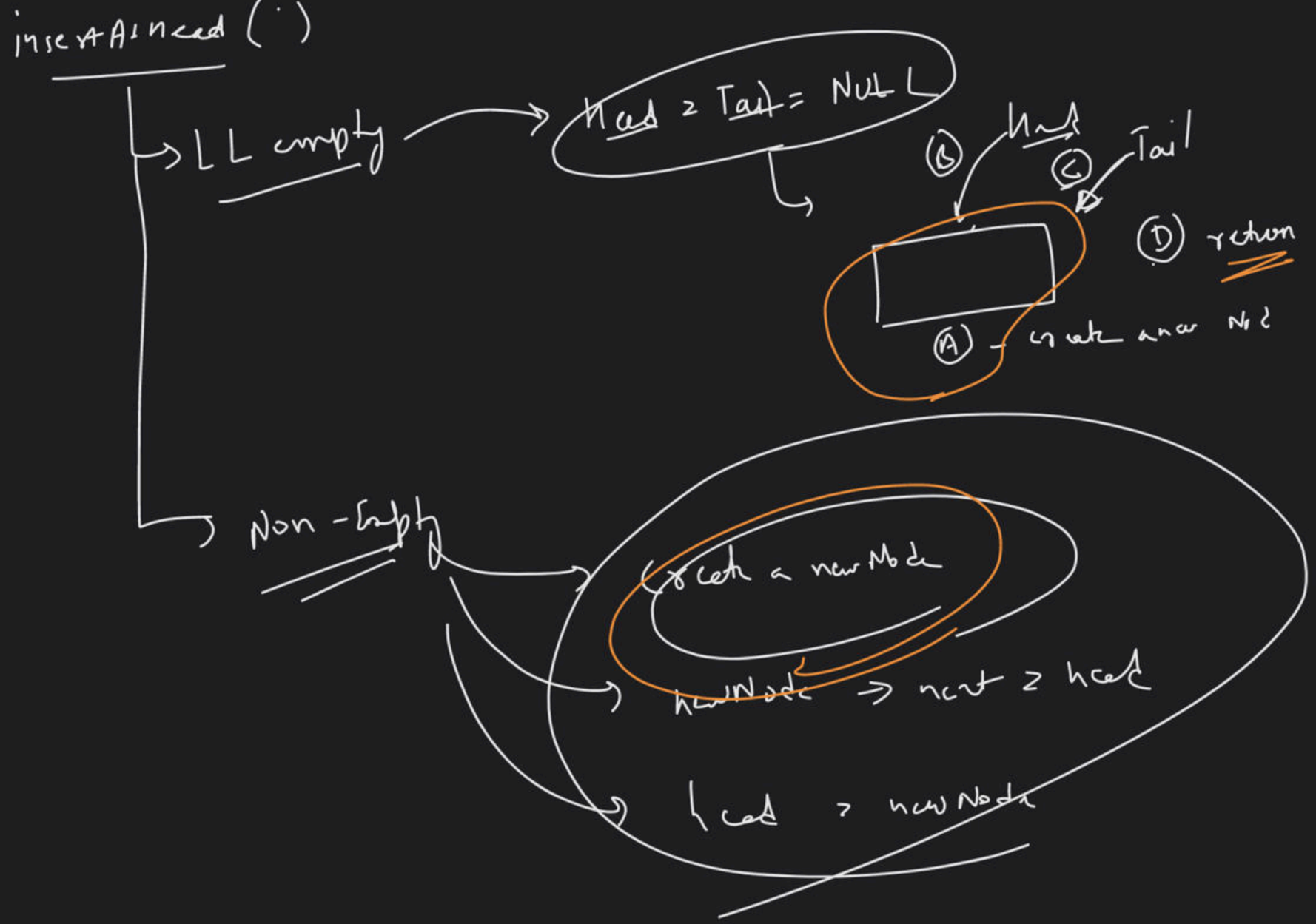

head → null
insertAtHead (head, 20)
insertAtHead (head, 30)
insertAtHead (head, 50)
insertAtHead (head, 90)
head
10
50
30
20
NULL
head → starting point of L.L
tail pointer → Ending pointing of L.L
head
90
50
30
20
X
90 50 30 20


insertAtTail
50
30
20
10
X
head
tail
tail->next
newNode
tail
insertAtTail(tail, 10)
Create a node
tail->next = newNode
tail = newNode


insertAtHead
Create a Node
newNode->next = head
head = newNode
insertAtTail will
Empty
non empty
LL
create a Node
tail->next = newNode
=
tail = newNode
head = NULL
tail = NULL
Key exist → LL empty
display
NULL->next
tail=newNode
head

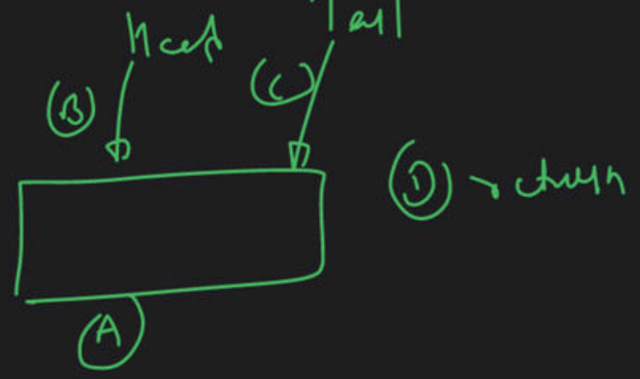

head → X
Tail → X
head
tail
tail
null
head
first node


insert At Head
Node * newNode = new Node (data)
newNode → next = head
→ if (head == NULL) ← LL empty
tail = newNode
head = newNode
NULL
tail pointer update


insertAtHead ( )
LL empty
Head = Tail = NULL
B
Head
C
Tail
D return
A create a new Node
Non-Empty
Create a new Node
newNode → next = head
head = newNode

insert At Tail

→ LL Empty → Head = Tail = NULL



→ LL non-Empty

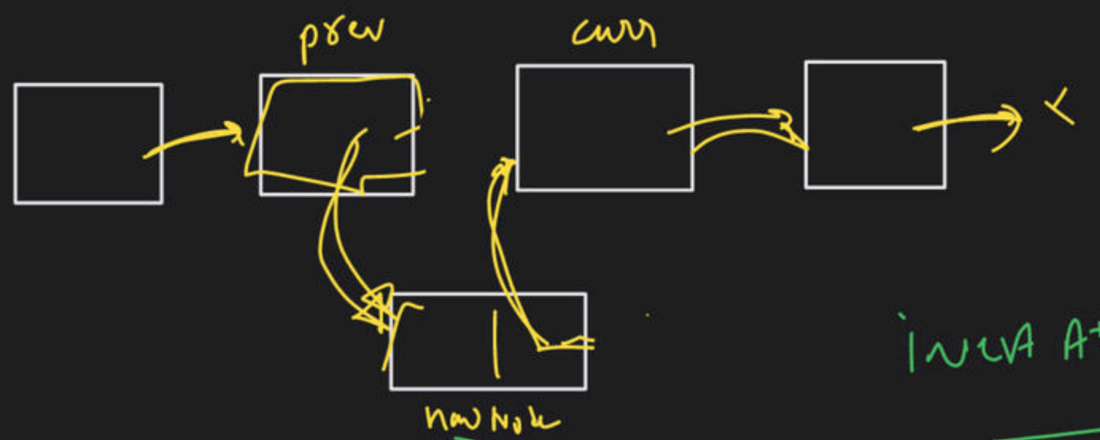
→ create a new Node

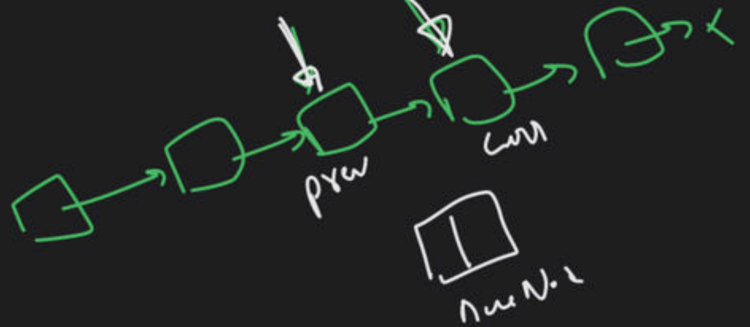
→ tail → next = newNode

→ tail = newNode

→ Head

→ Tail





→ insertAtPosition

→ position → 3rd position

→ value → value

insertAtPosition(head, tail, data)

→ LL Empty

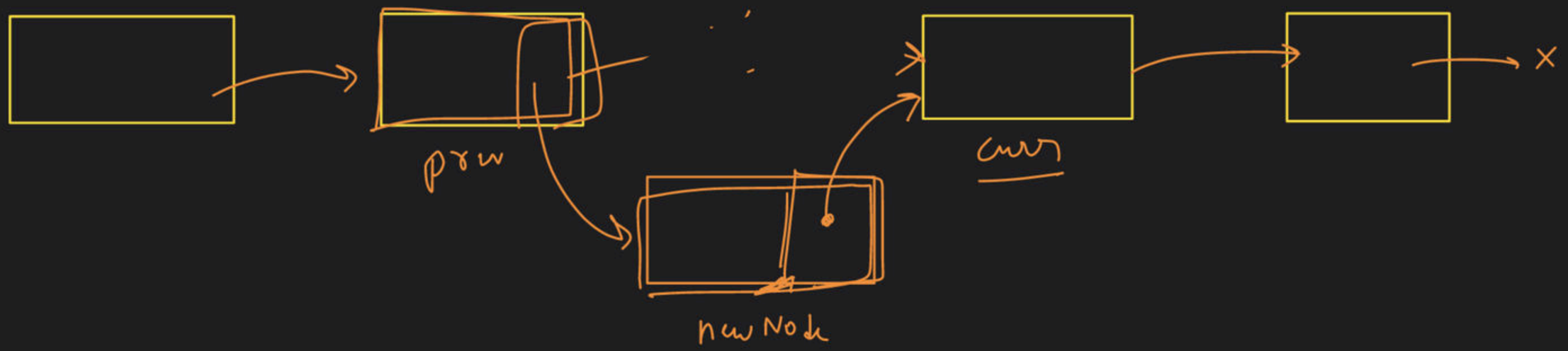
→ non-Empty →

(A) find that positi

(B) create a node

(C) newNode → next = curr

(D) prev → next = newNode

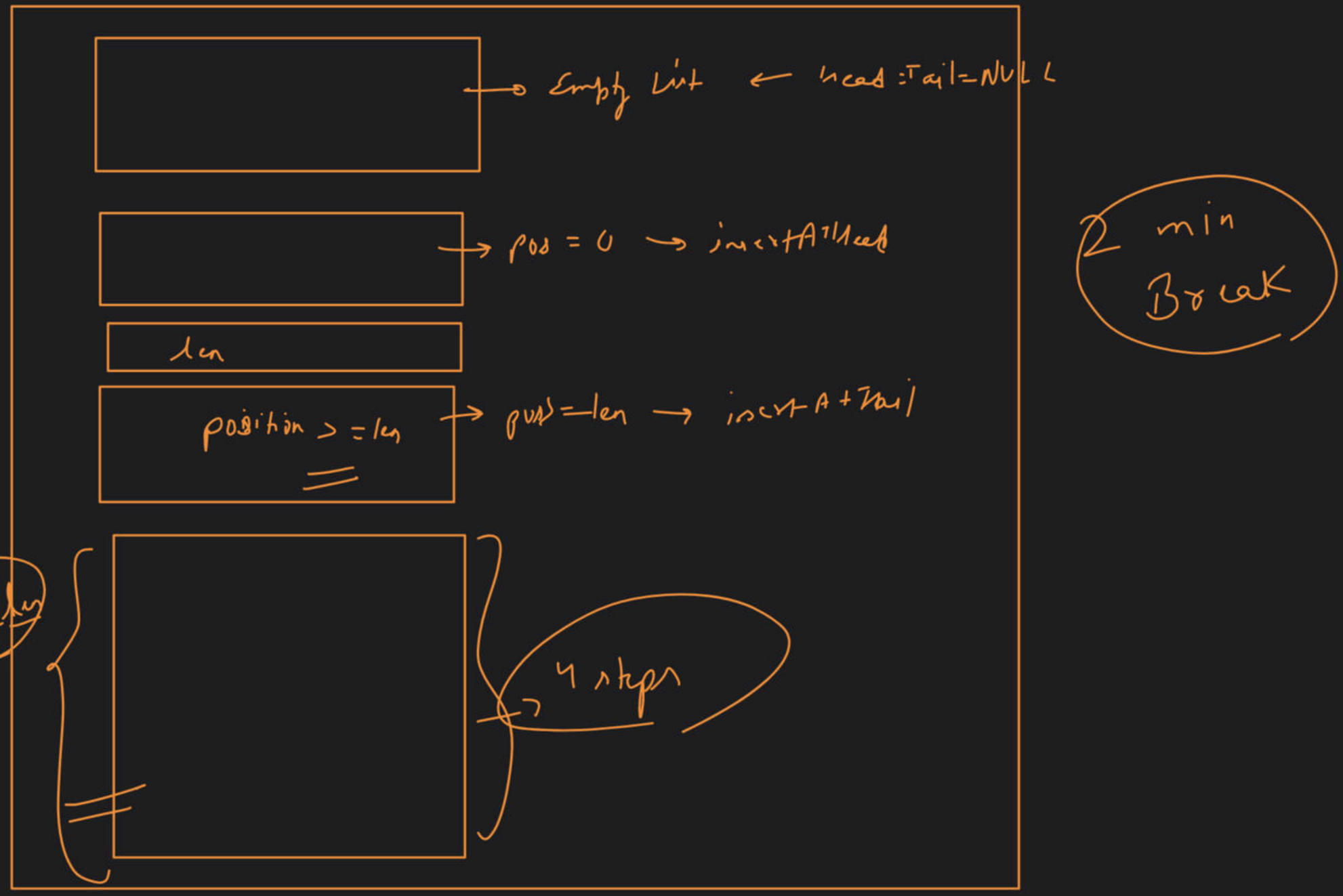


(A) find position -> prev curr

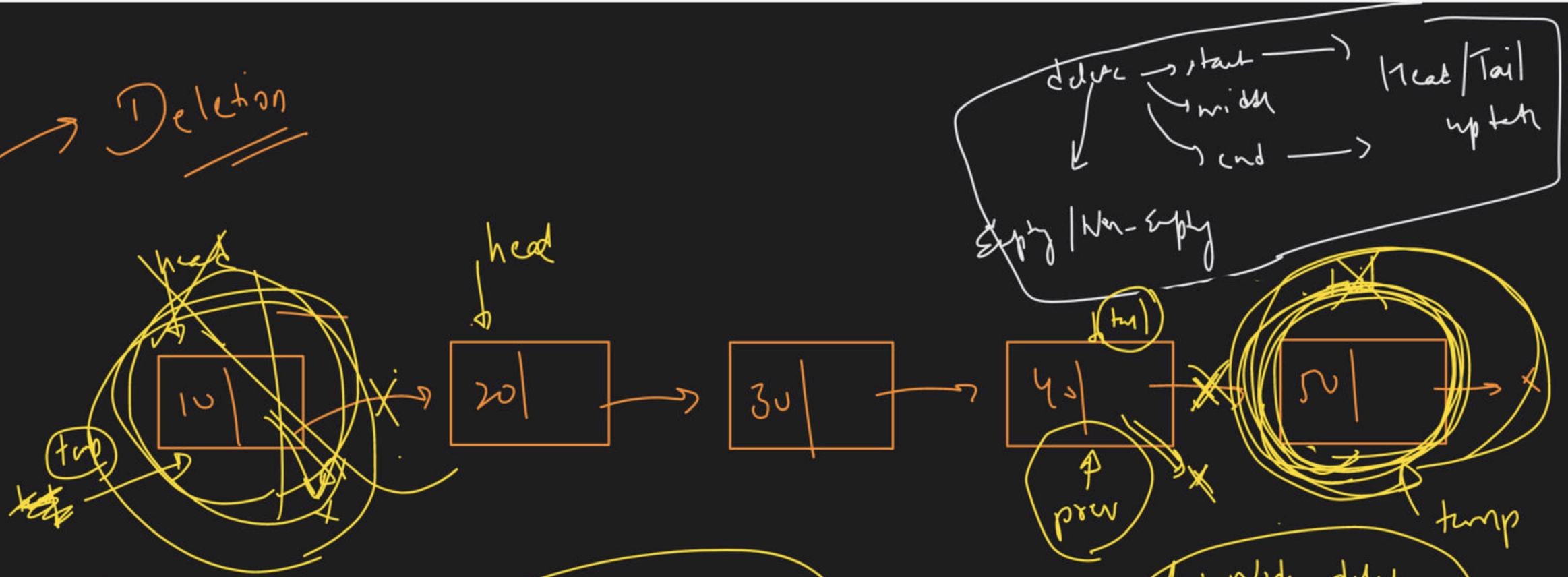
(B) create newNode

(C) newNode -> next = curr

(D) prev -> next = newNode


Empty List ← head :Tail=NULL
pos = 0 → insertAtHead
len
position >= len
pos=len → insertAtTail
position < len
4 steps
2 min Break

Deletion



deque → start → middle → end → Head / Tail update

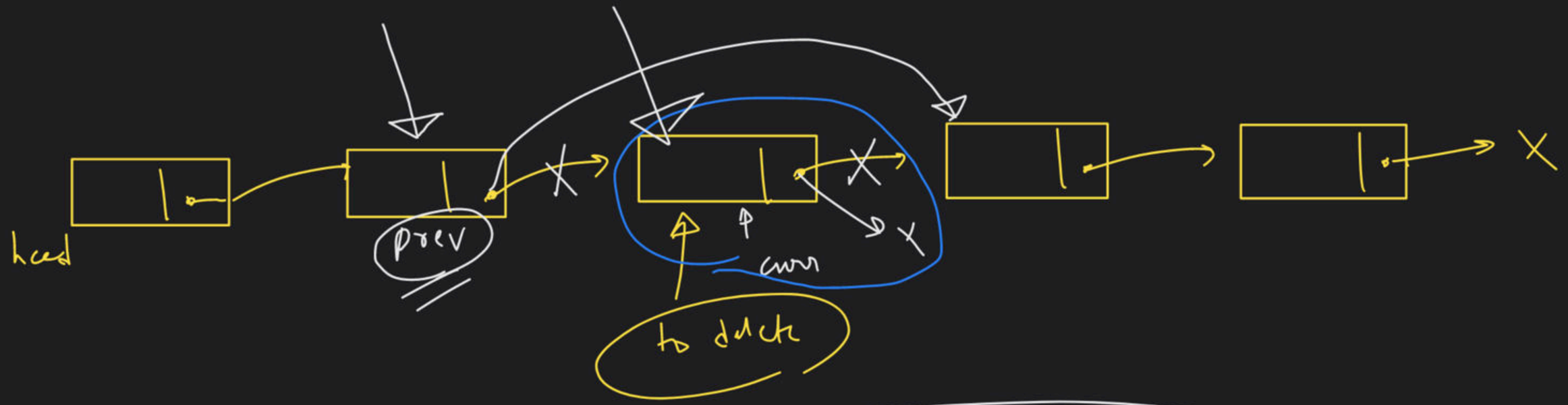
Empty / Non-Empty

First Node delete

```
head = head -> next ;
temp -> next = null
delete temp
```

Last Node delete

```
prev find
prev -> next = NULL
tail = prev
delete (temp)
```



Steps

find prev & curr

prev -> next = curr -> next

curr -> next = NULL

delete curr