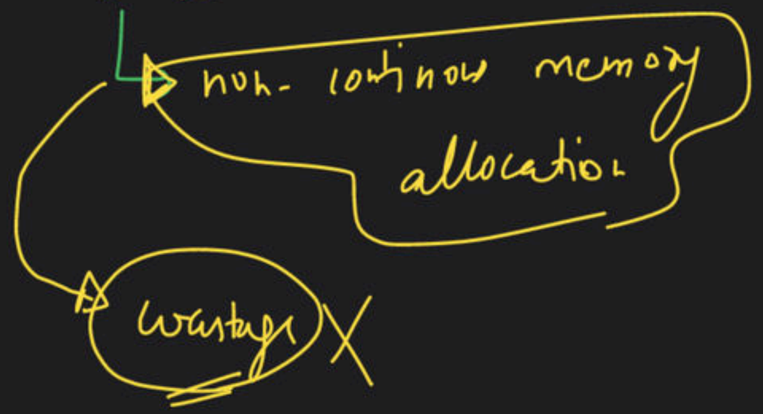
Linked List - Class 1

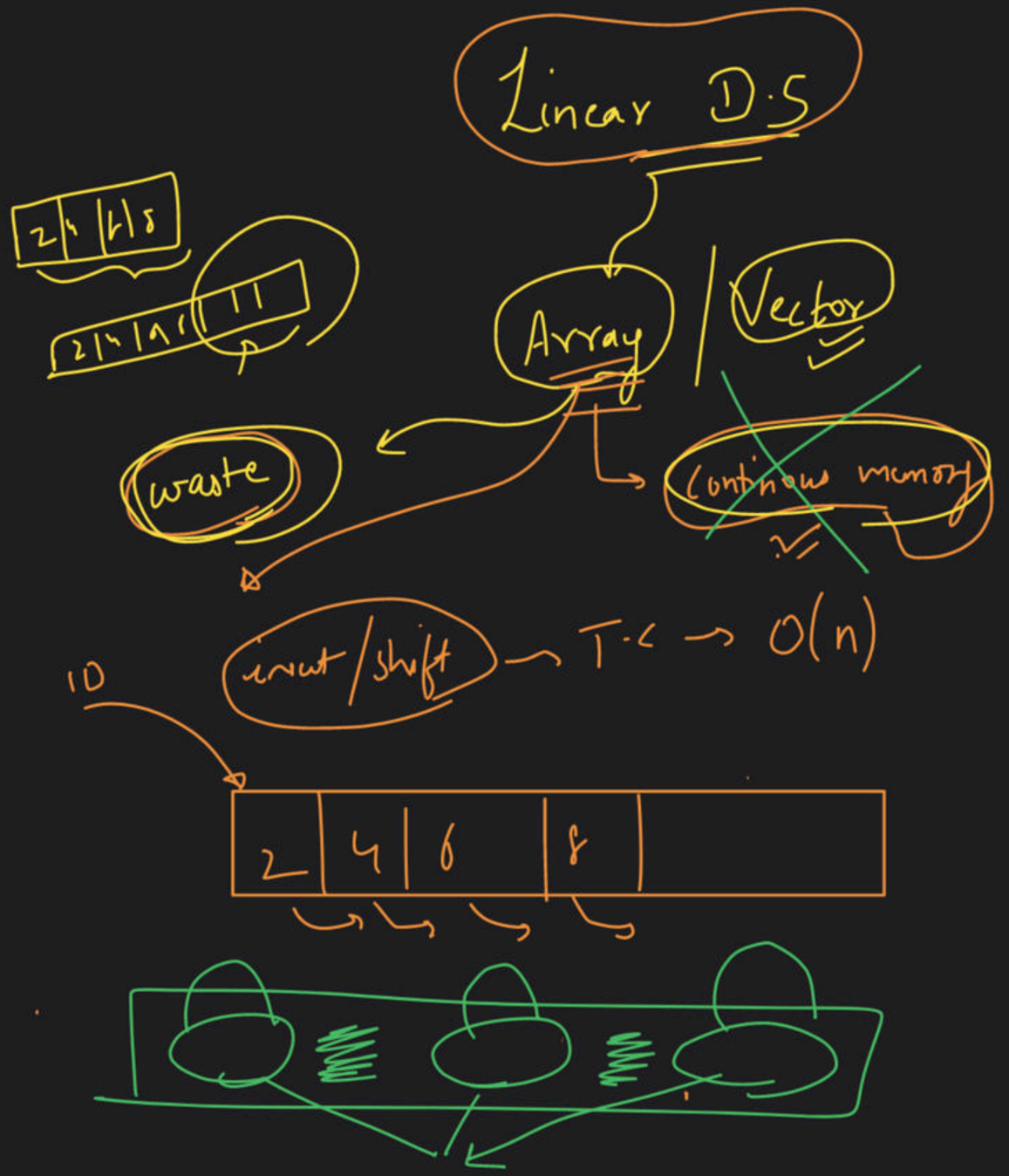
Special class

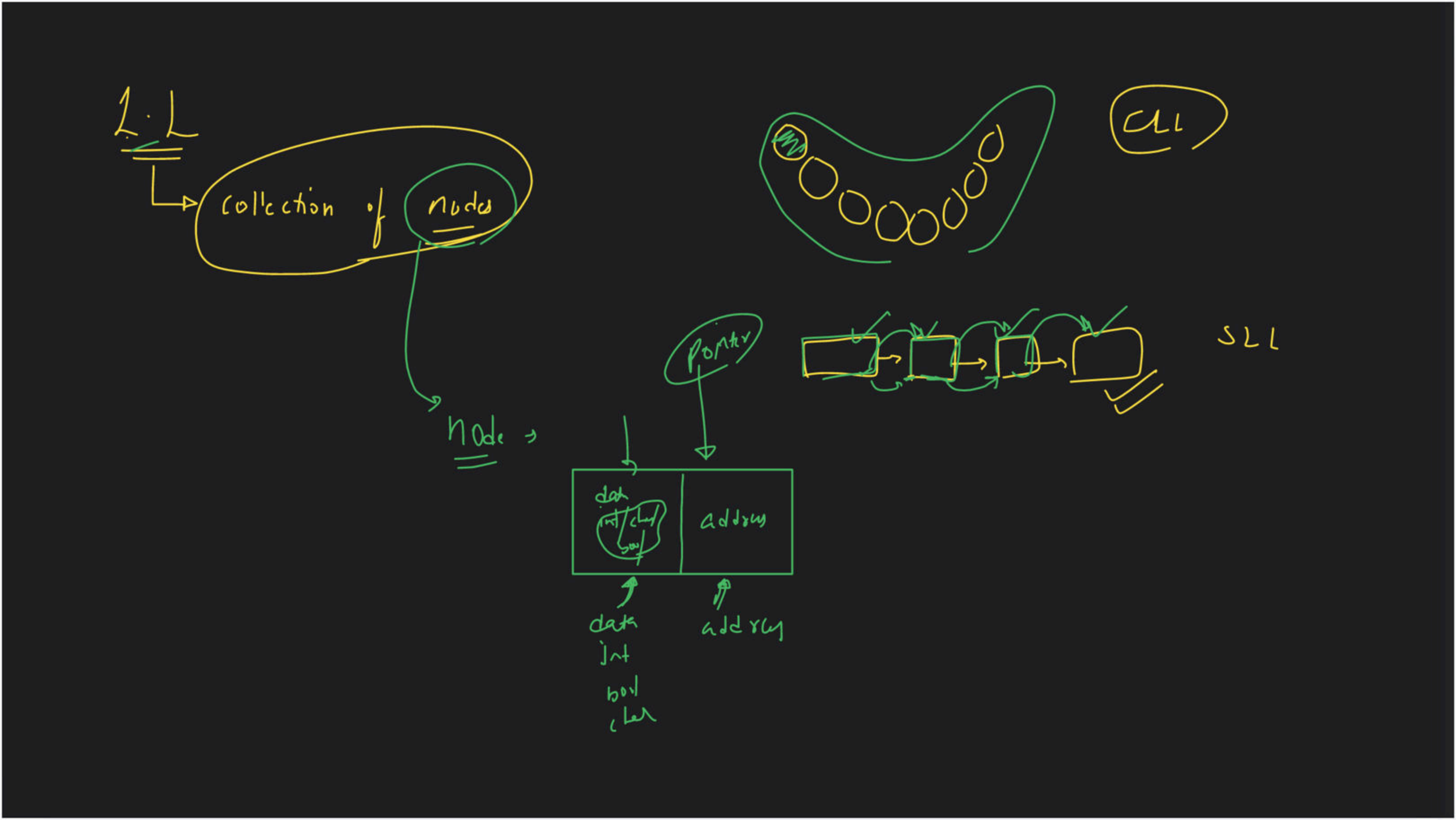
2 min
Break dedo

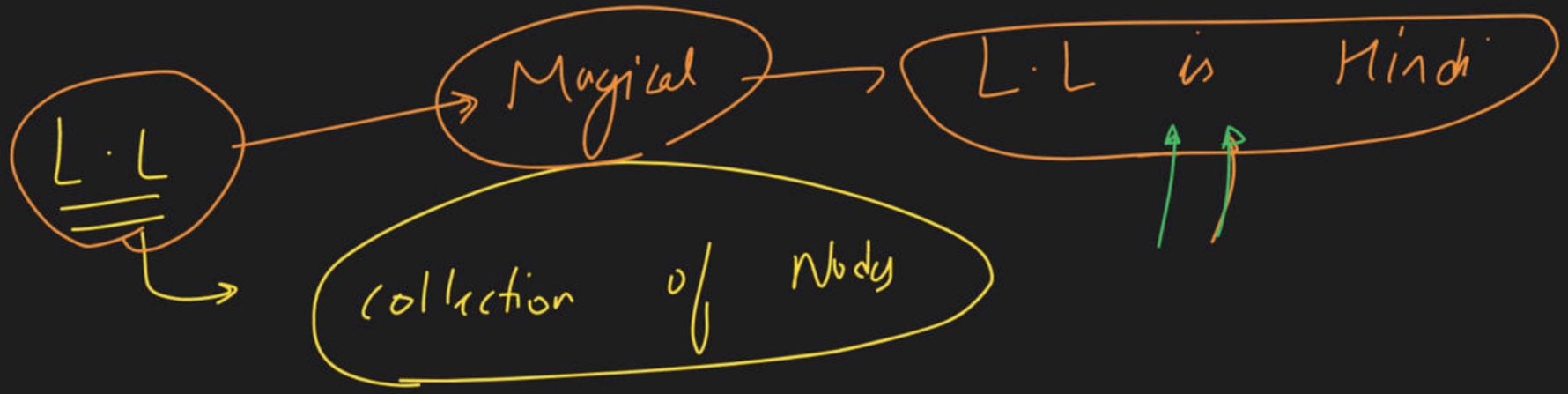
Love Babbar • Nov 3, 2023

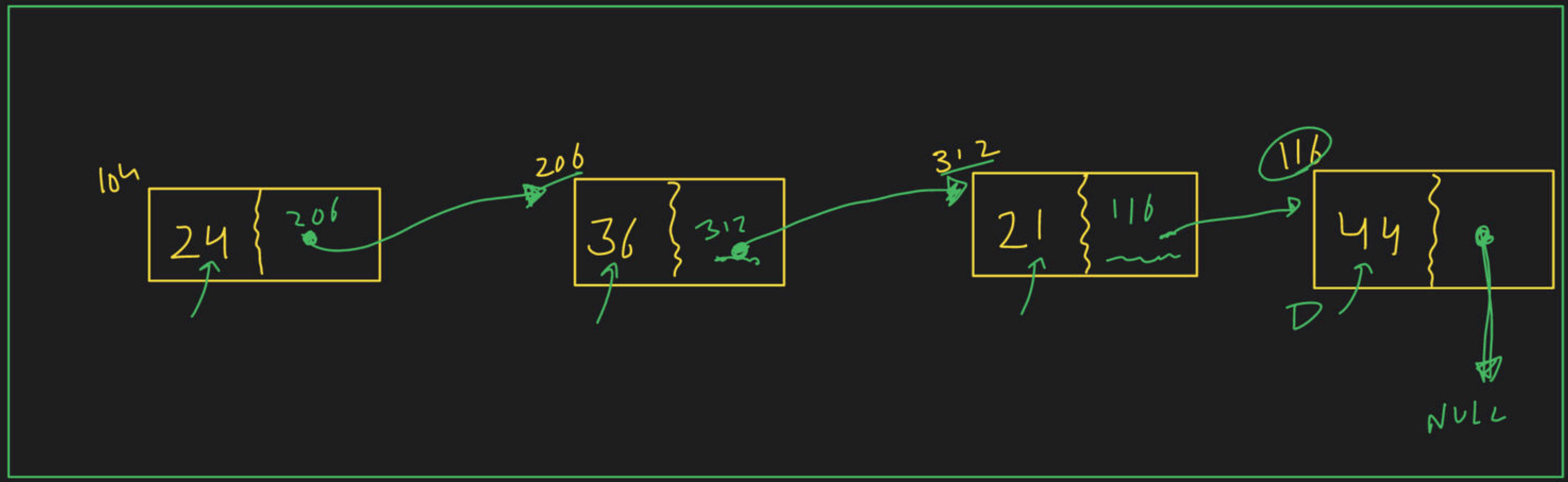
Linked List

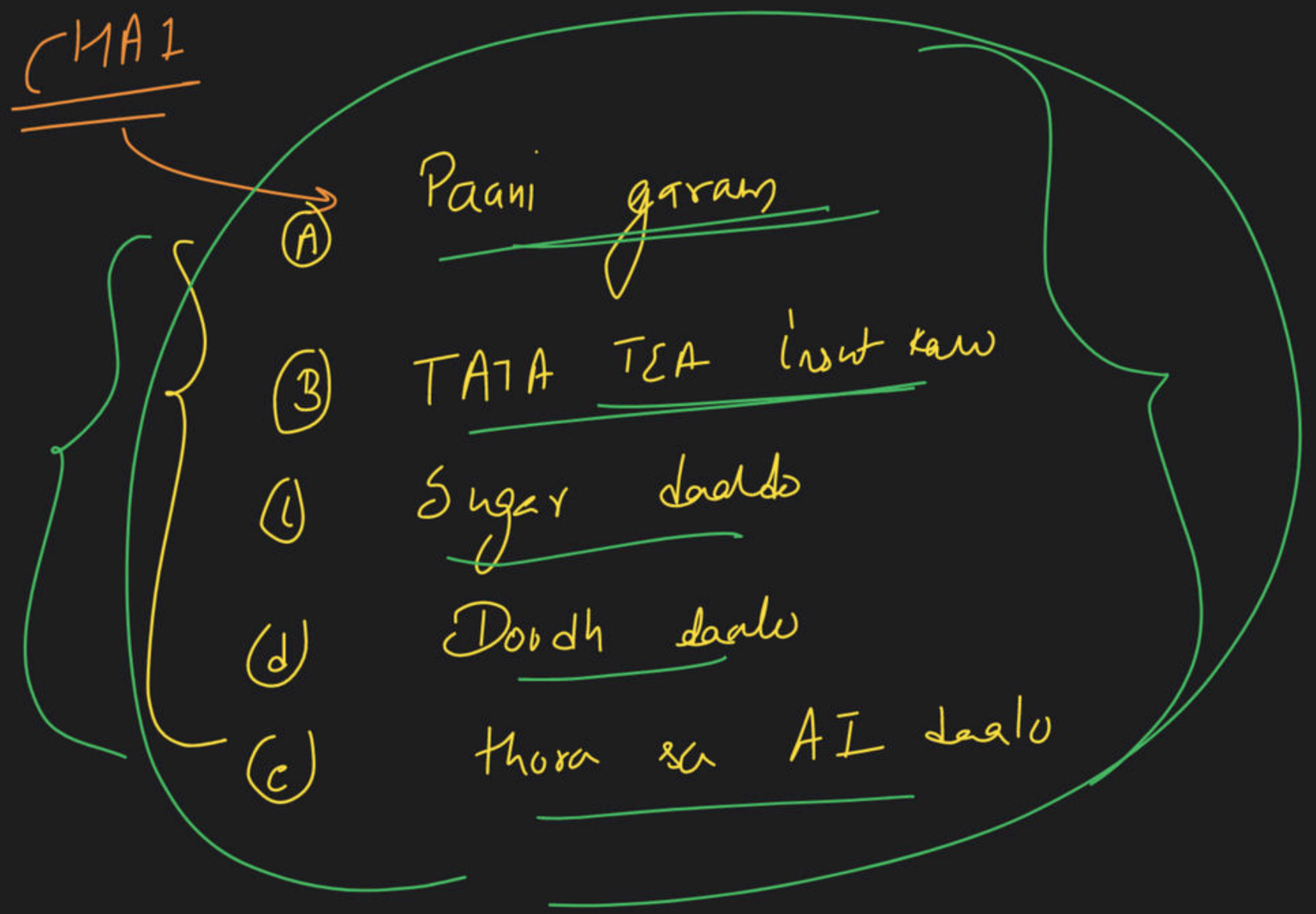

non- continous memory
allocation
wastage

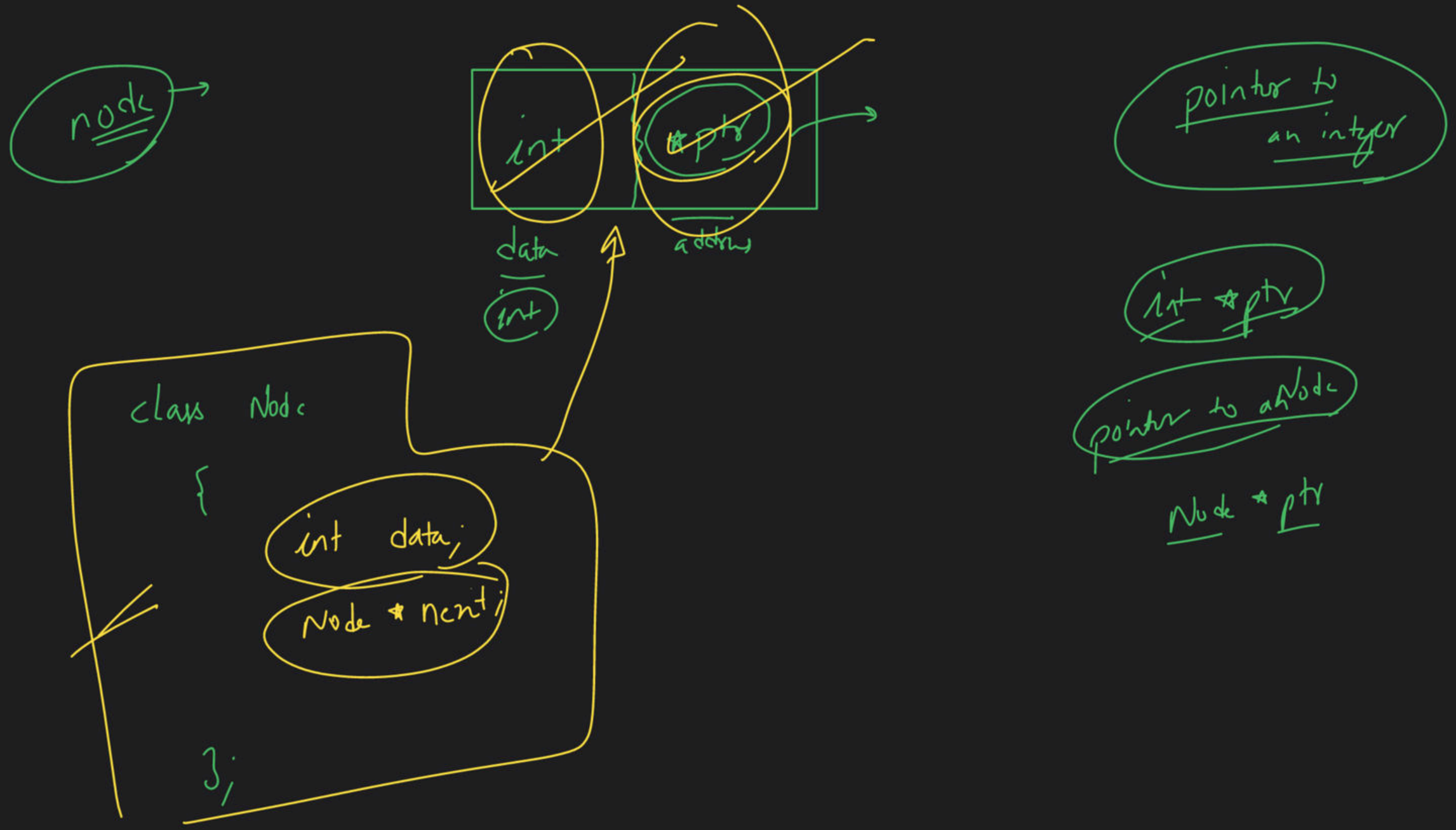

Linear D.S
Array
Vector
waste
continous memory
insert/shift
T.C → O(n)
ID
2
4
6
8

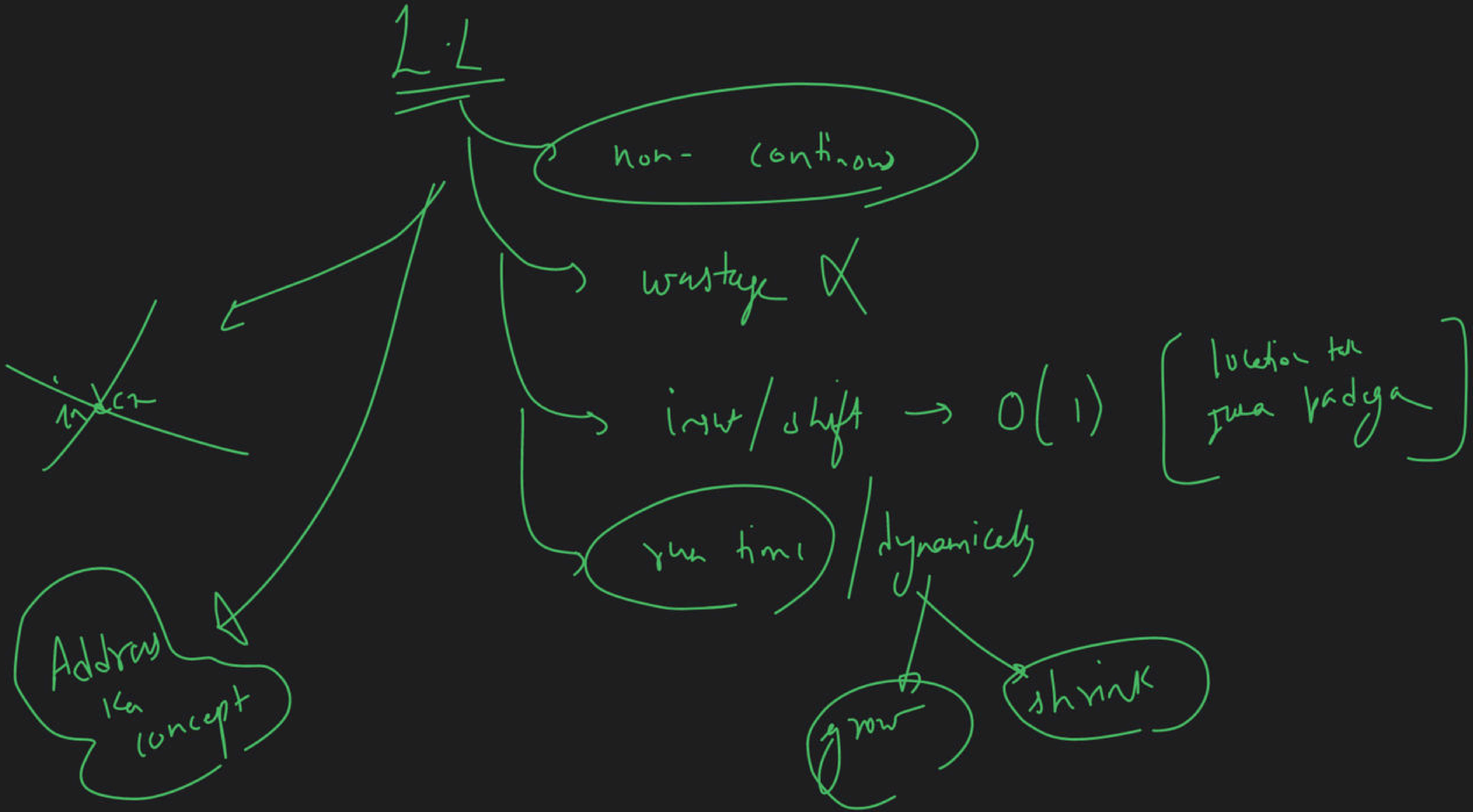

L.L
collection of nodes
node =
CLL
pointer
SLL
data
int
bool
char
address
data
int
bool
char
address


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
thoda sa AI daalo
D
Doodh daalo
E
Sugar daalo


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

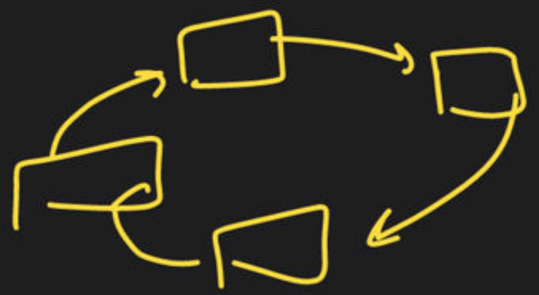

L.L
non- continous
wastage X
insert/shift → O(1)
location tak
jana padega
run time / dynamically
grow
shrink
index
Address ka concept

Types

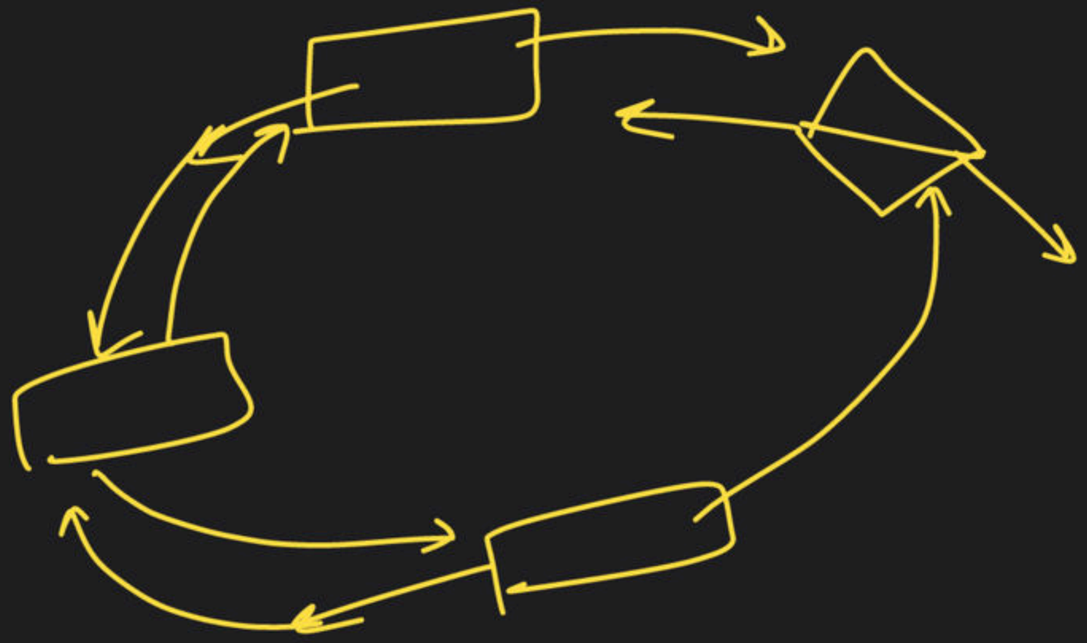
- Singly L.L

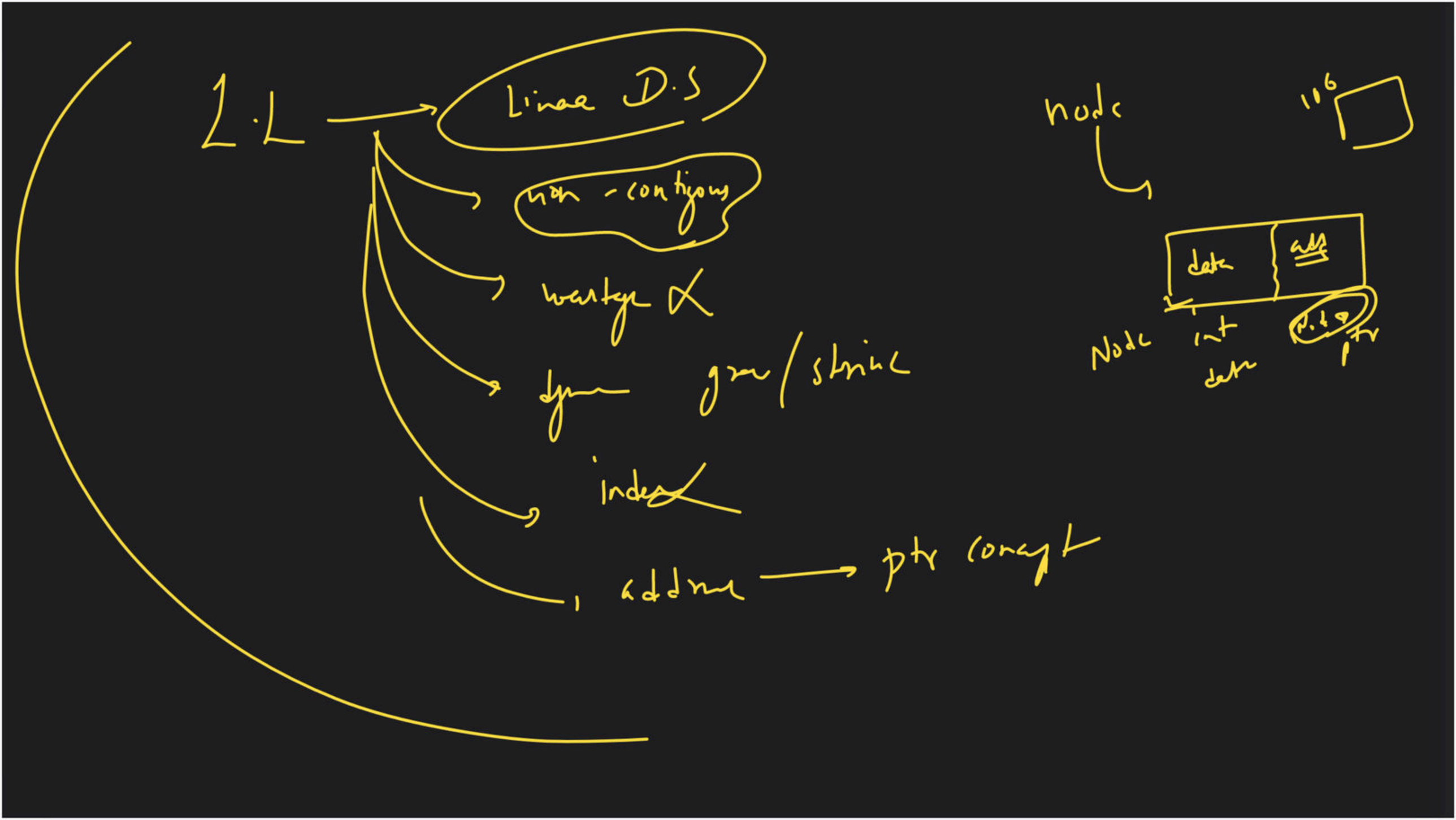
- Doubly L.L

- Circular Singly L.L

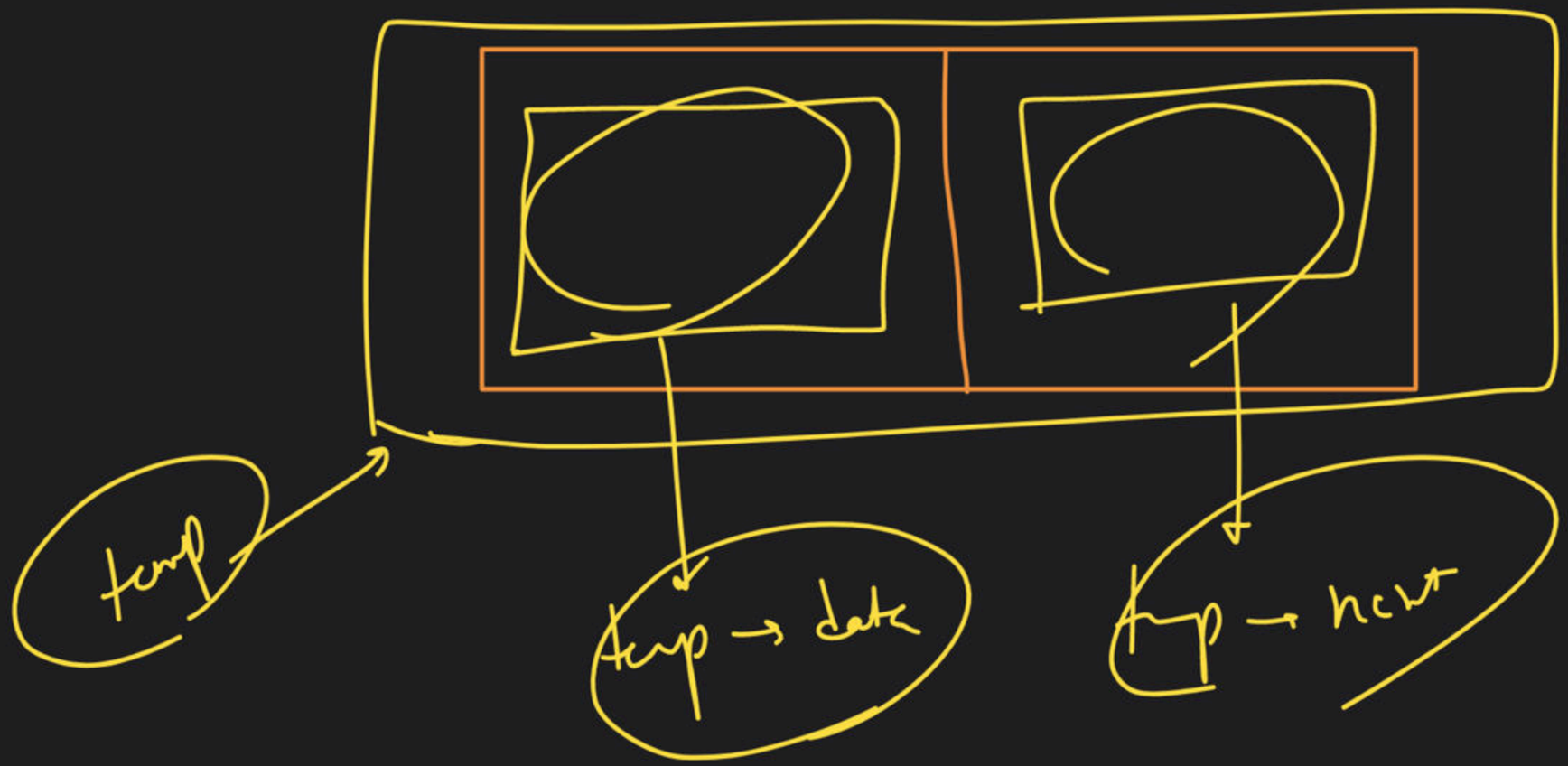


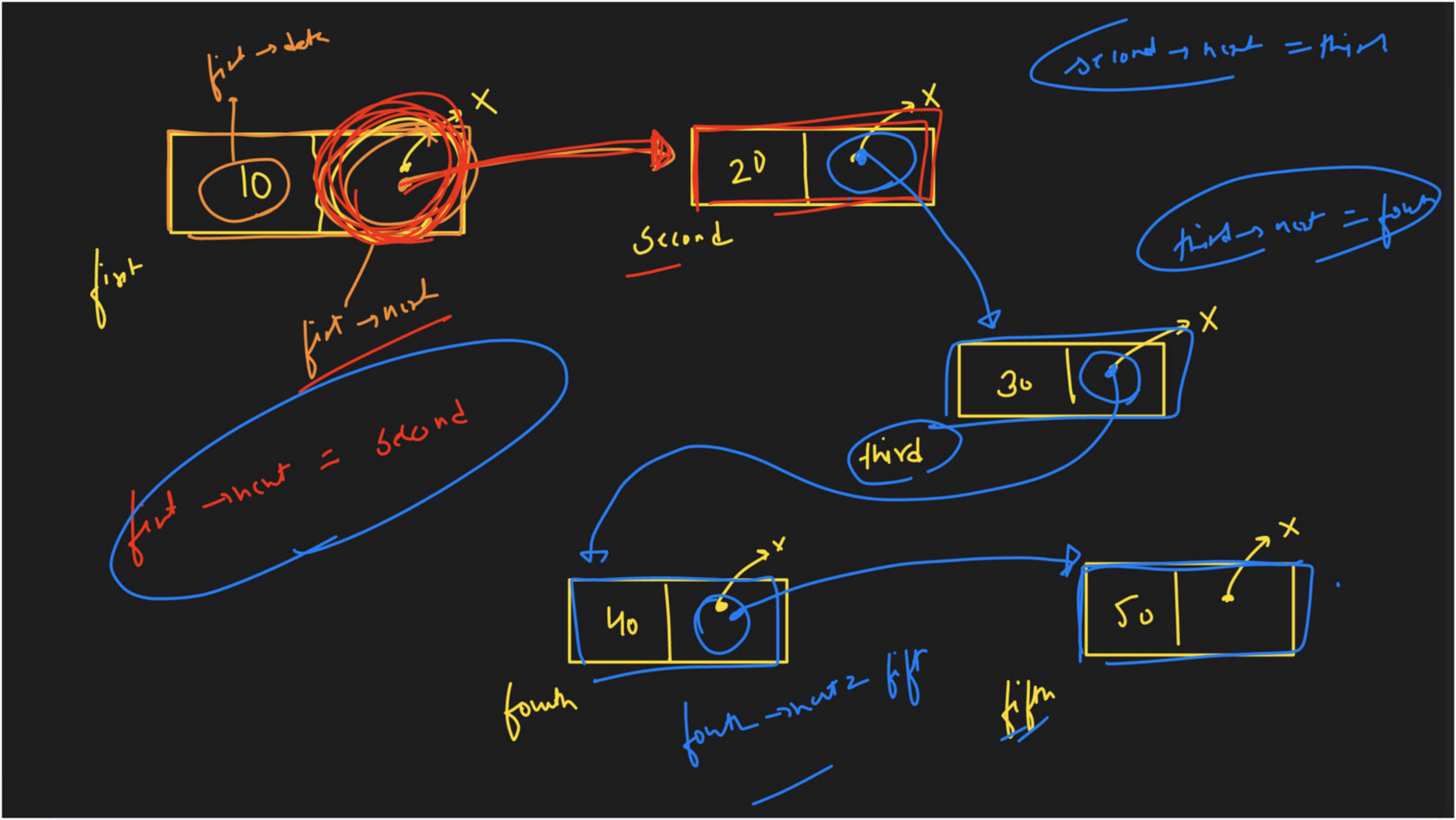
Circular Doubly L.L

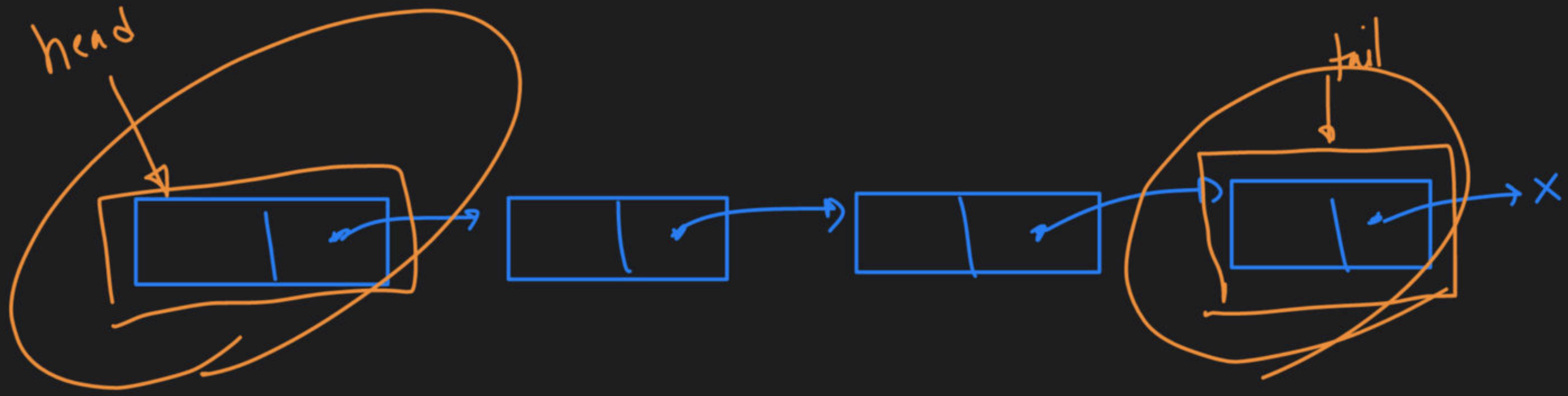


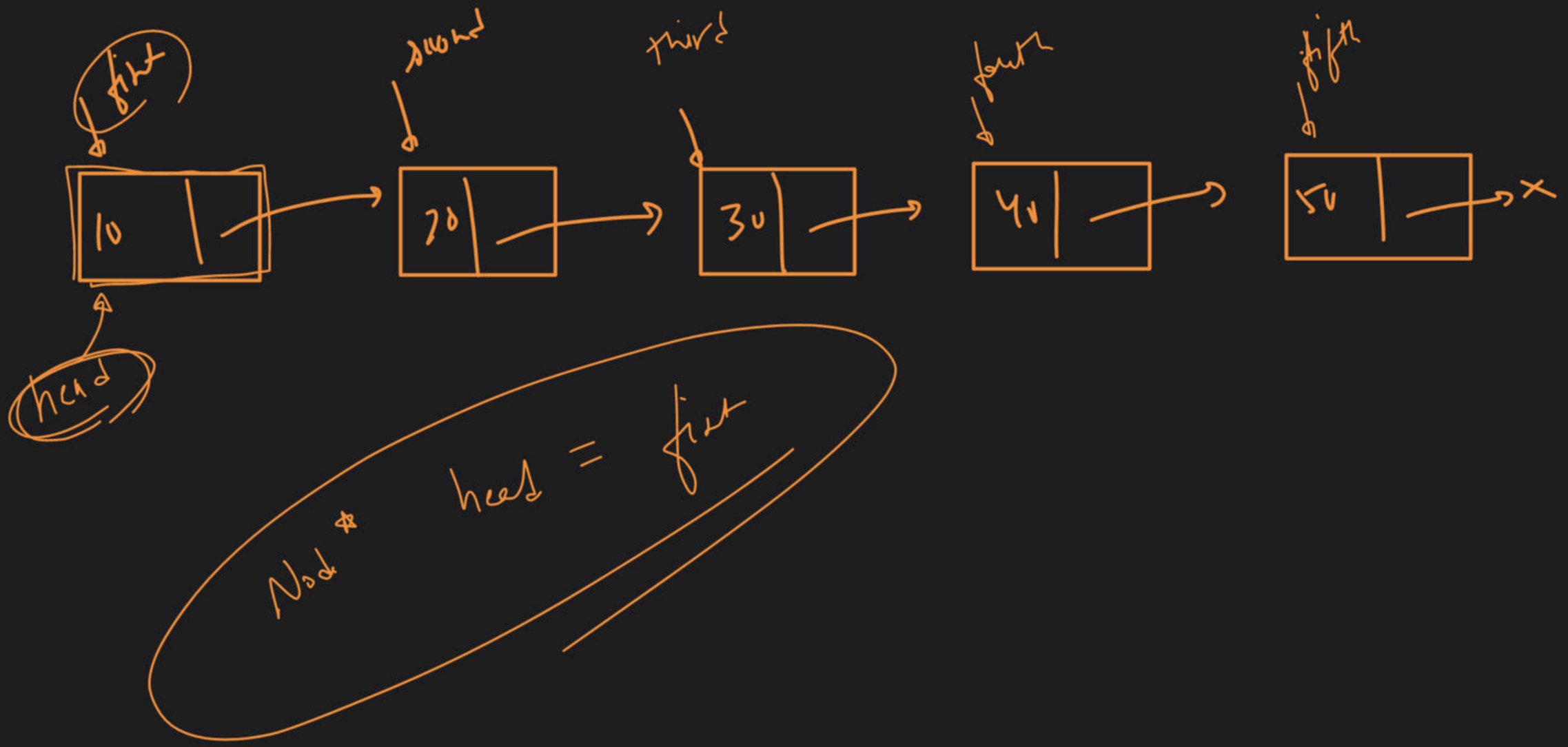

L.L
Linear D.S
non - contigous
node
data
Node
int
data

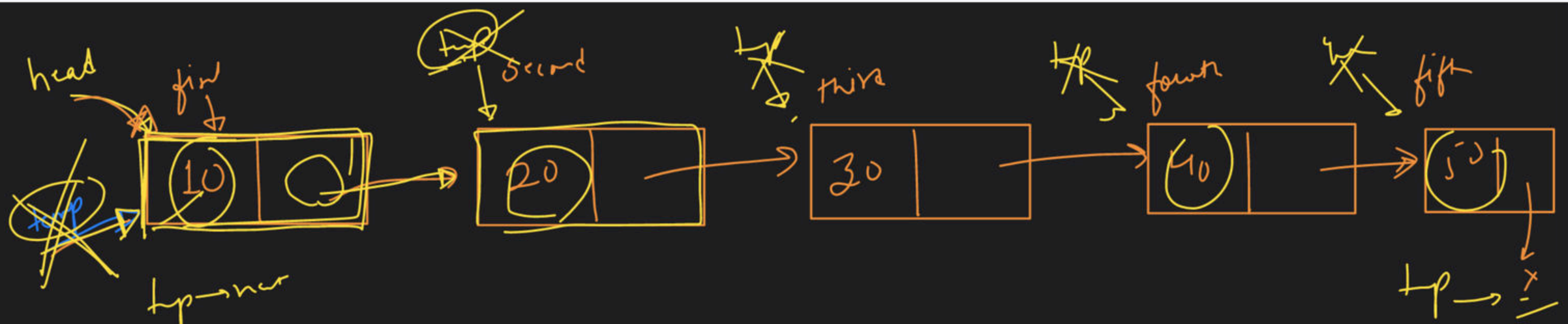
Node *temp = new Node()



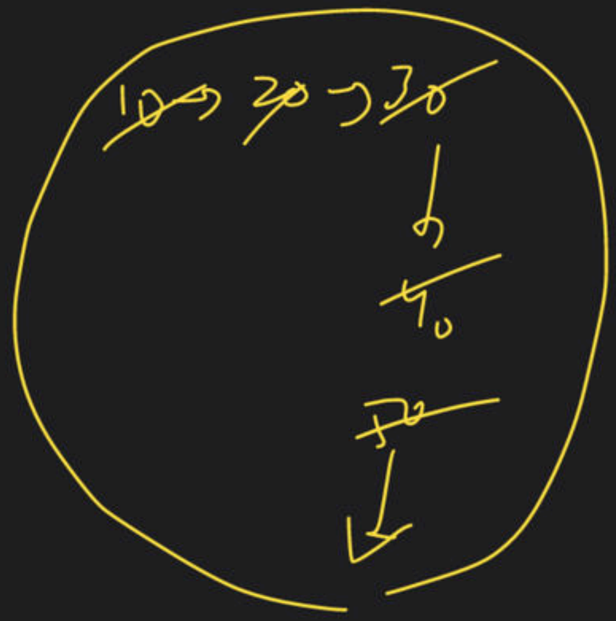

first -> data
10
X
first
first -> next
second -> next = third
20
X
second
third -> next = fourth
30
X
third
first -> next = second
40
X
fourth
fourth -> next = fifth
50
X
fifth

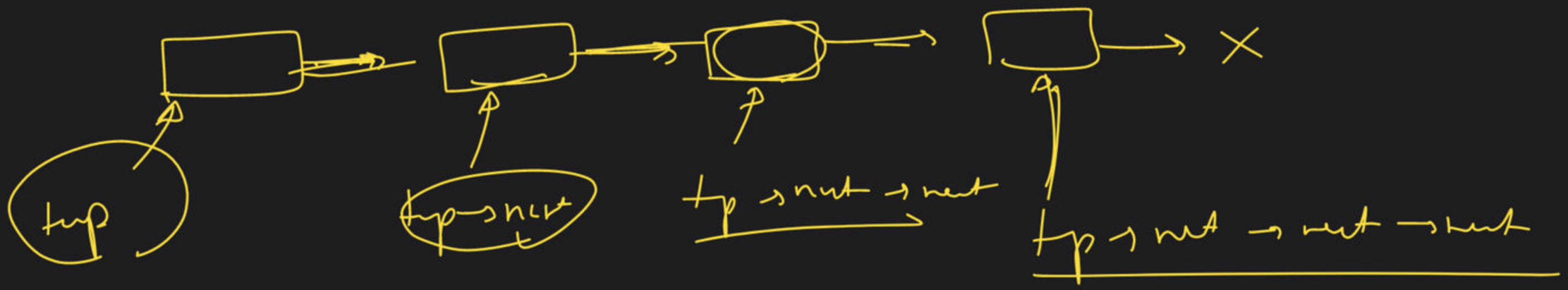

head
tail
X

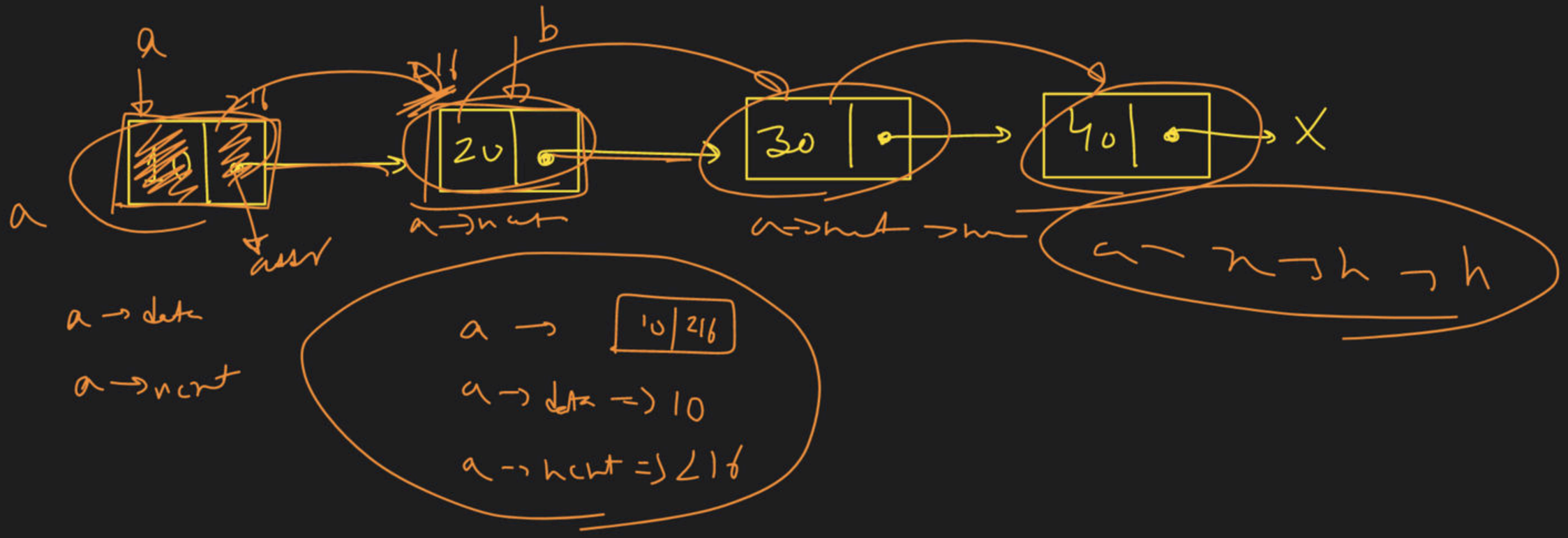

first
second
third
fourth
fifth
10
20
30
40
50
X
head
Node * head = first




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

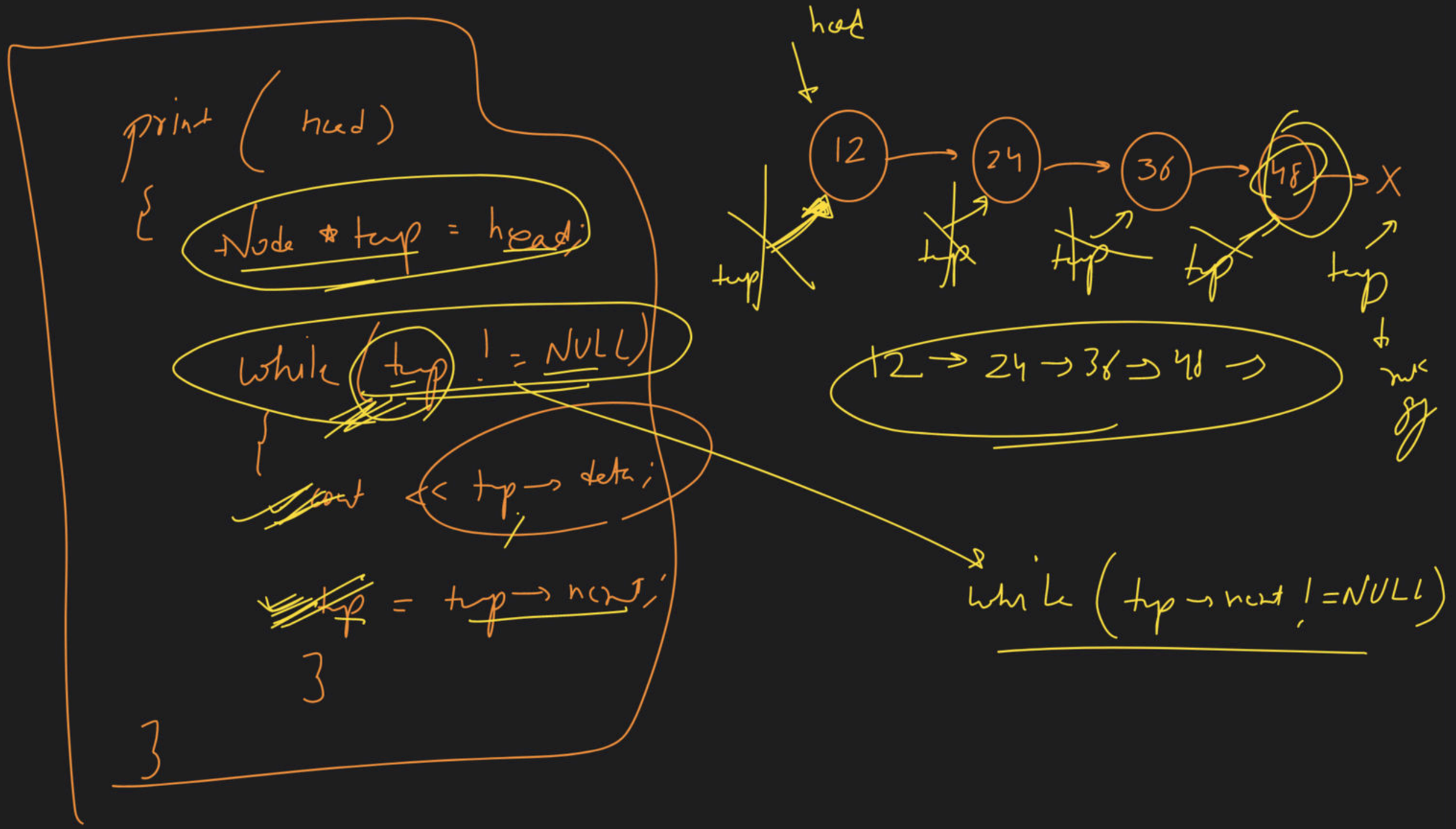



tmp
tmp->next
tmp->next->next
tmp->next->next->next
X


a
b
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

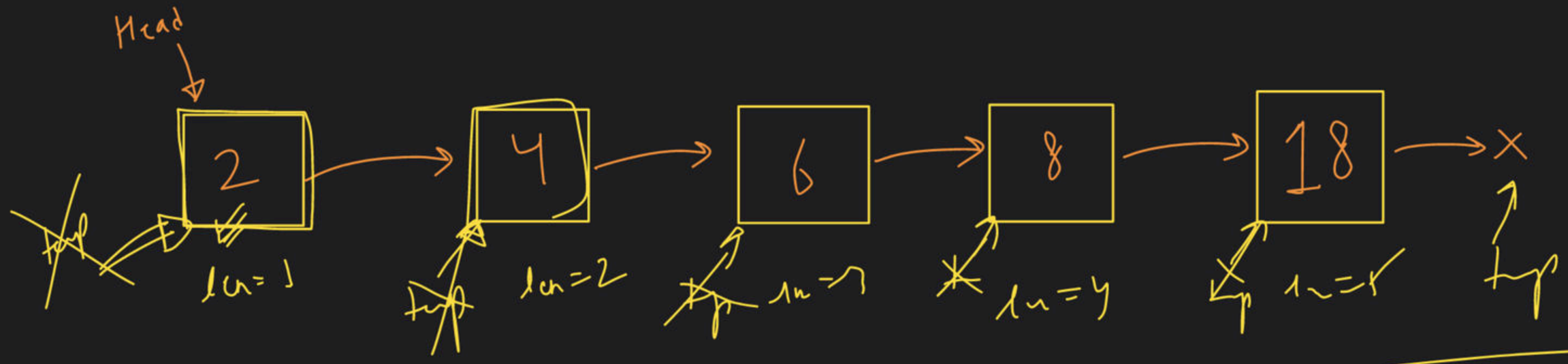
b = 216


print (head)
{
Node * tmp = head;
while (tmp != NULL)
{
cout << tmp -> data;
tmp = tmp -> next;
}
}
head
12
24
36
48
X
tmp
12 → 24 → 36 → 48 →
while (tmp -> next != NULL)

point Len = No. of Nodes

in a L.L

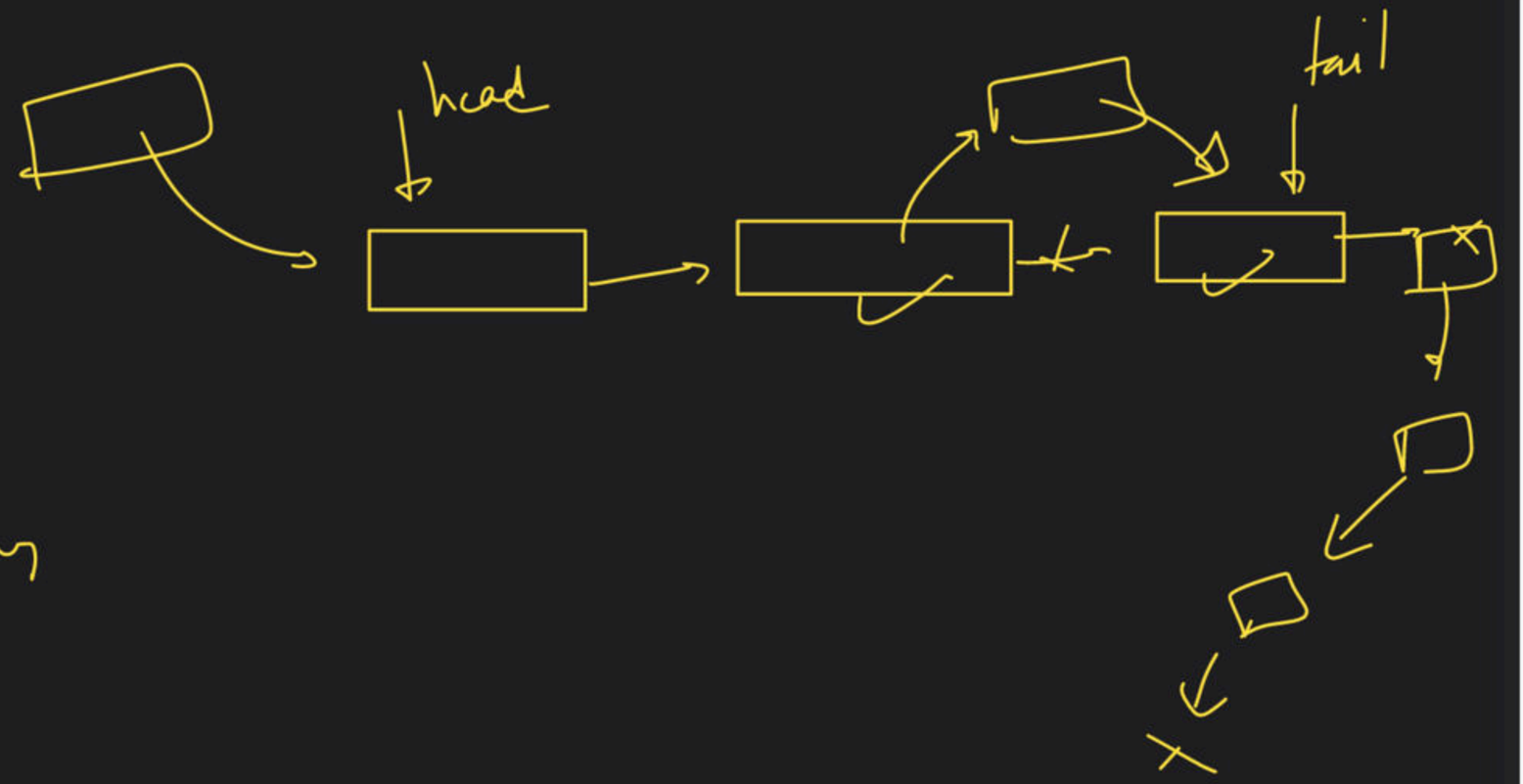
2 min



Len = 0

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

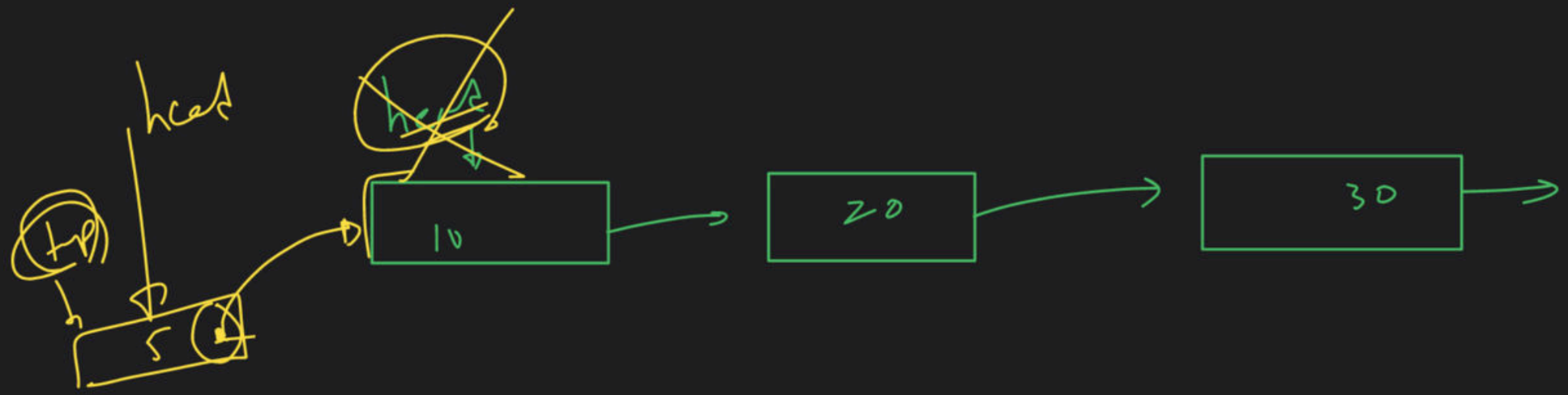
Insertion:-



- head → beginning
- tail
- insert at any point

Insertion

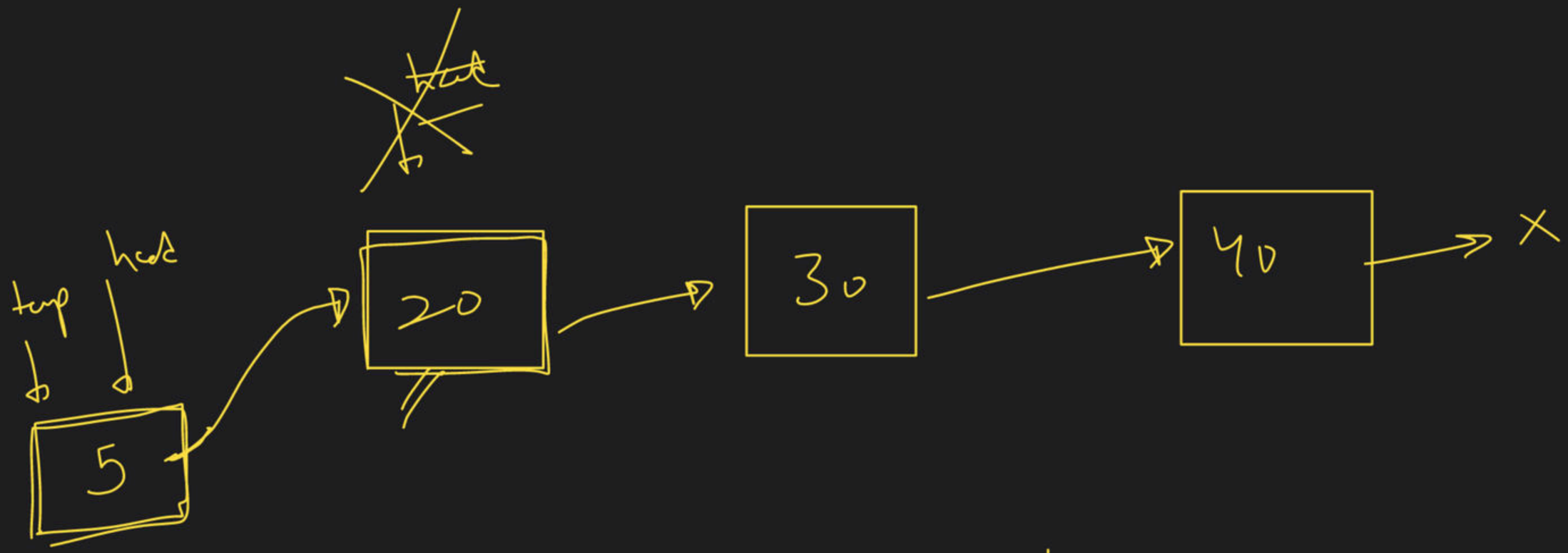
→ insertAtHead



(A) Create new Node

(B) temp → next = head

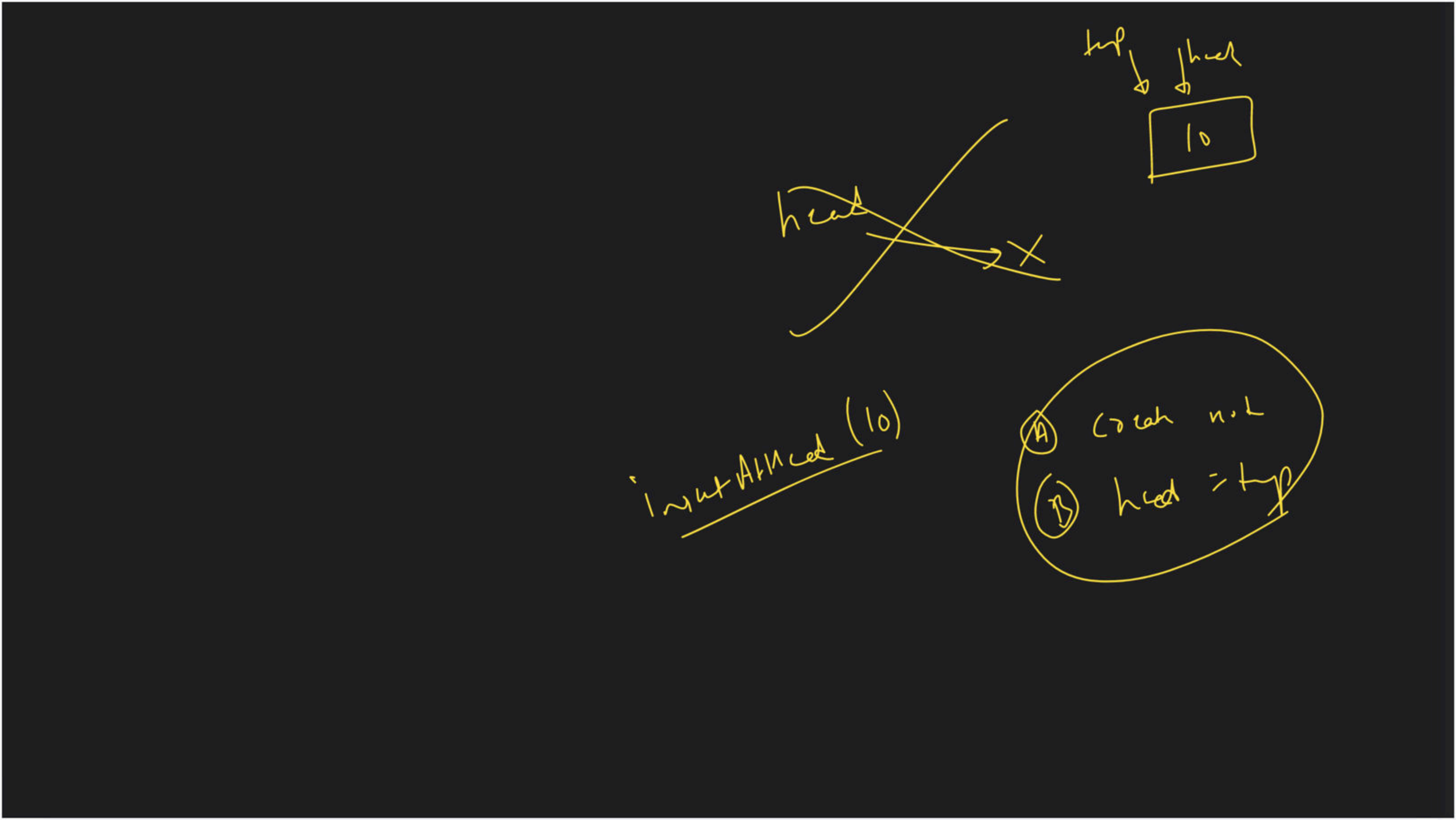
(C) head = temp



(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

insertAtTail (500)

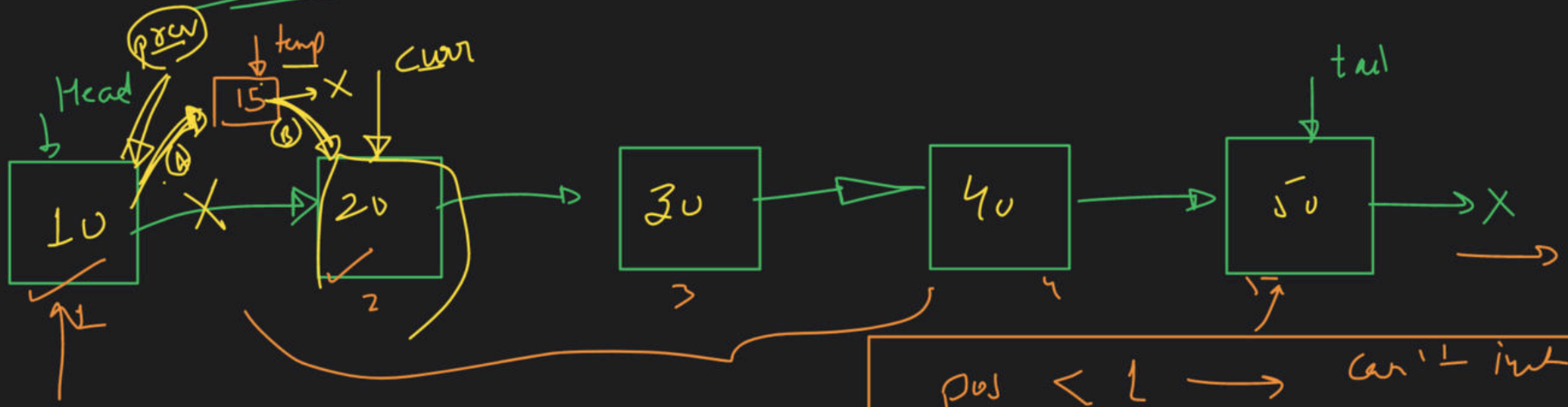
(A) Create New Node

(B) tail->next = temp

(C) tail = temp

Insert At Position

i/p → "n" position



position = 2

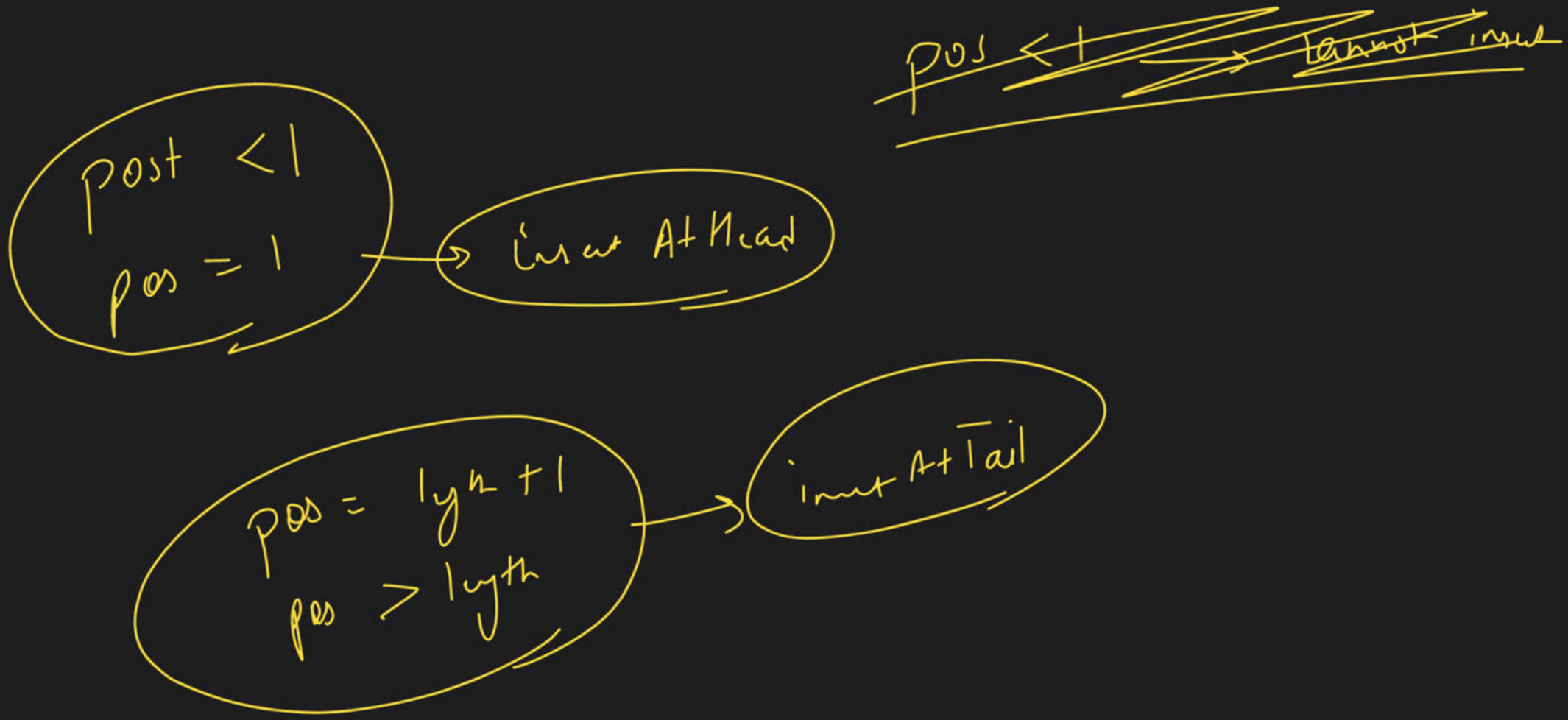
(A) create a node

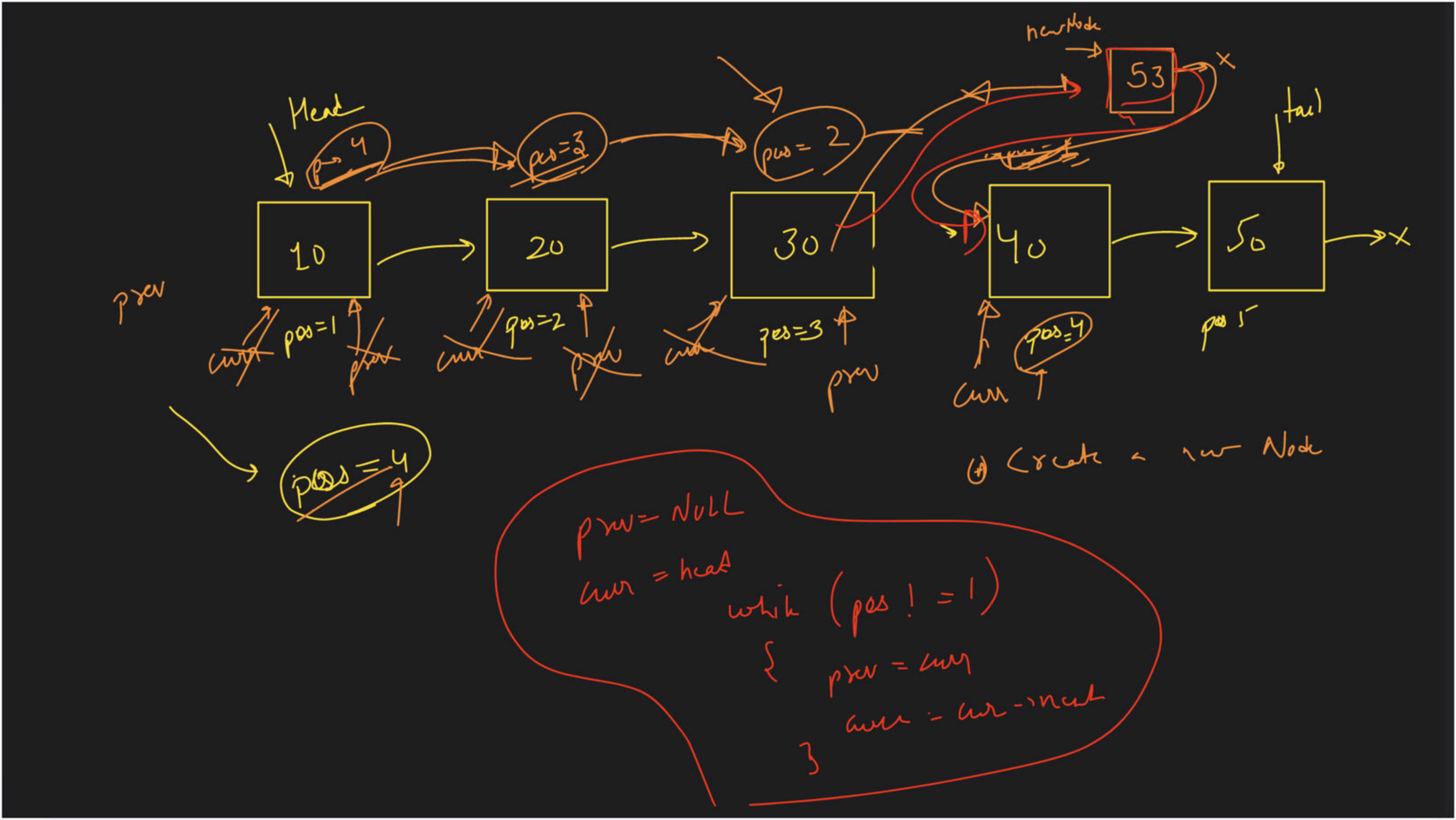
(B) traverse curr/prev to position

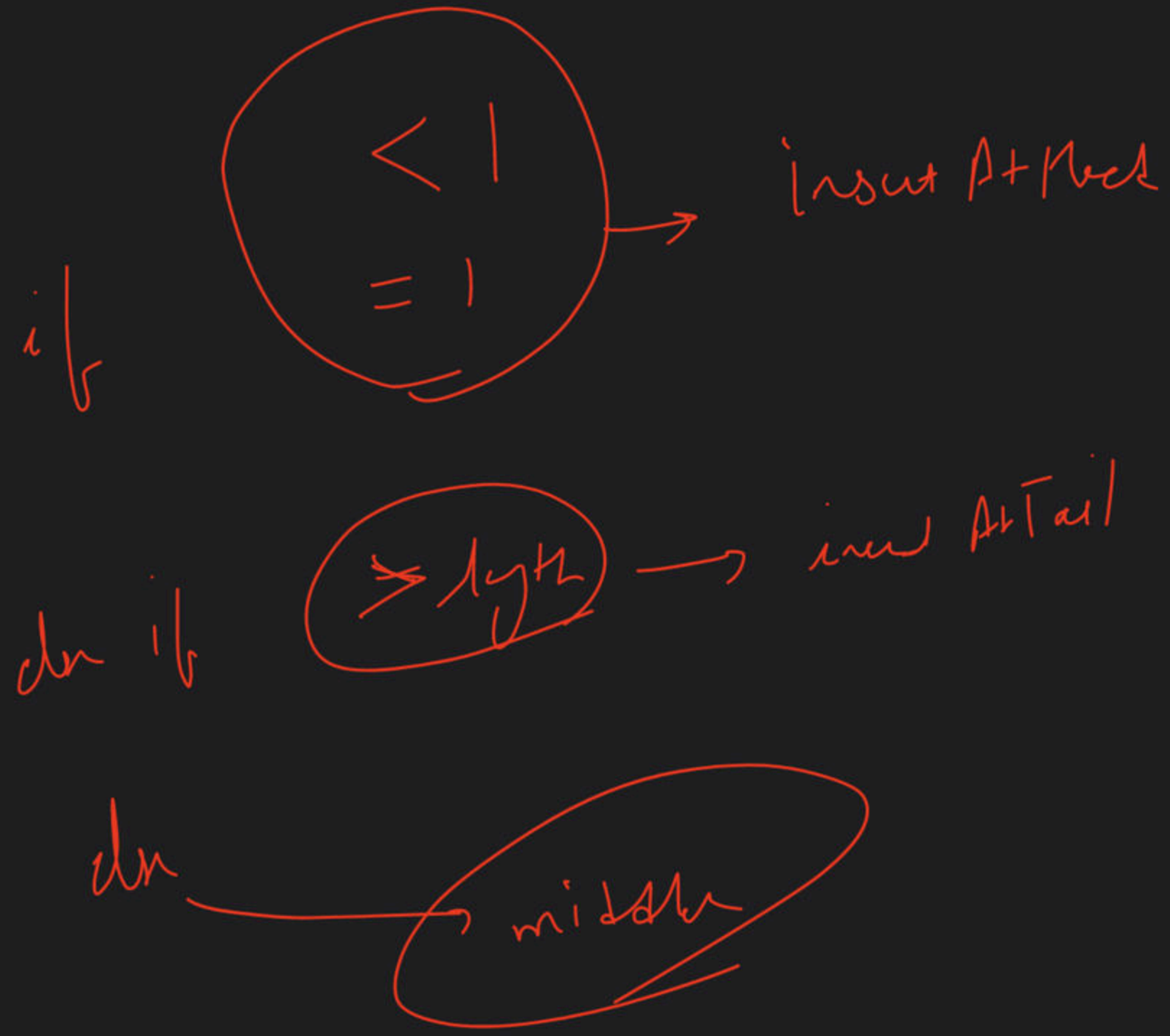
(C) prev → next = temp

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

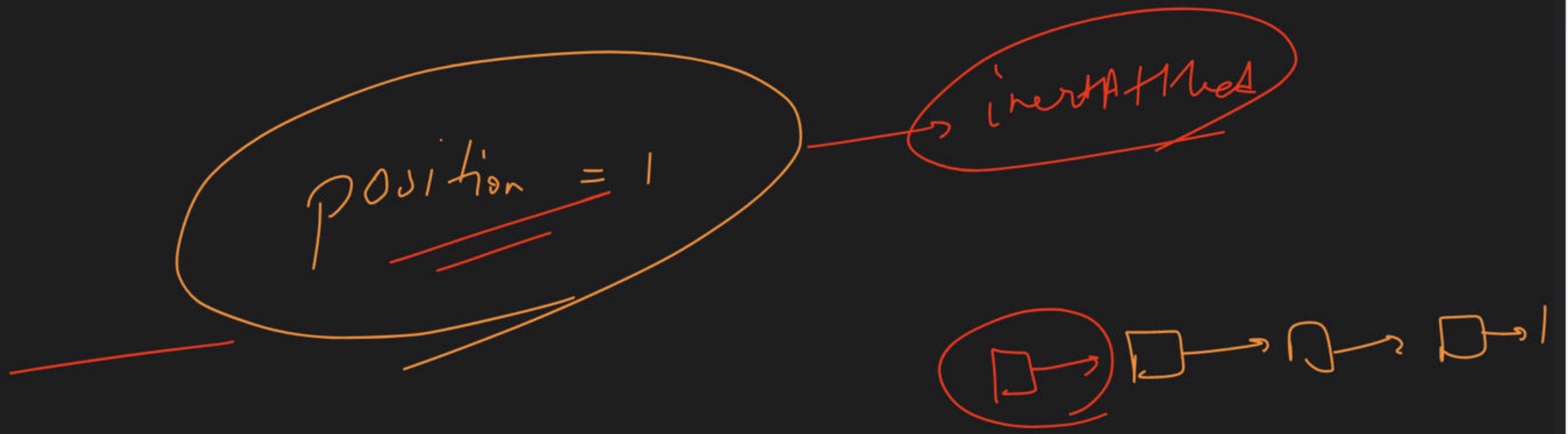

pos < 1
cannot insert
post < 1
pos = 1
insert At Head
pos = lyst + 1
pos > lyst
insert At Tail


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

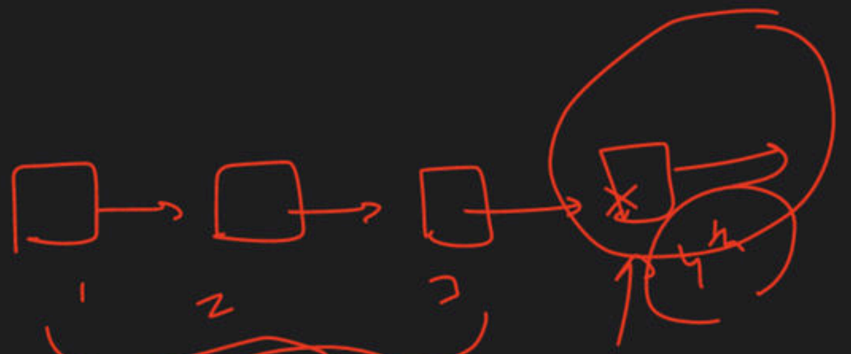

<1
=1
insert AtHead
>length
insert AtTail
middle

len = 4

pos → [1 ⟷ len + 1]


position = 1

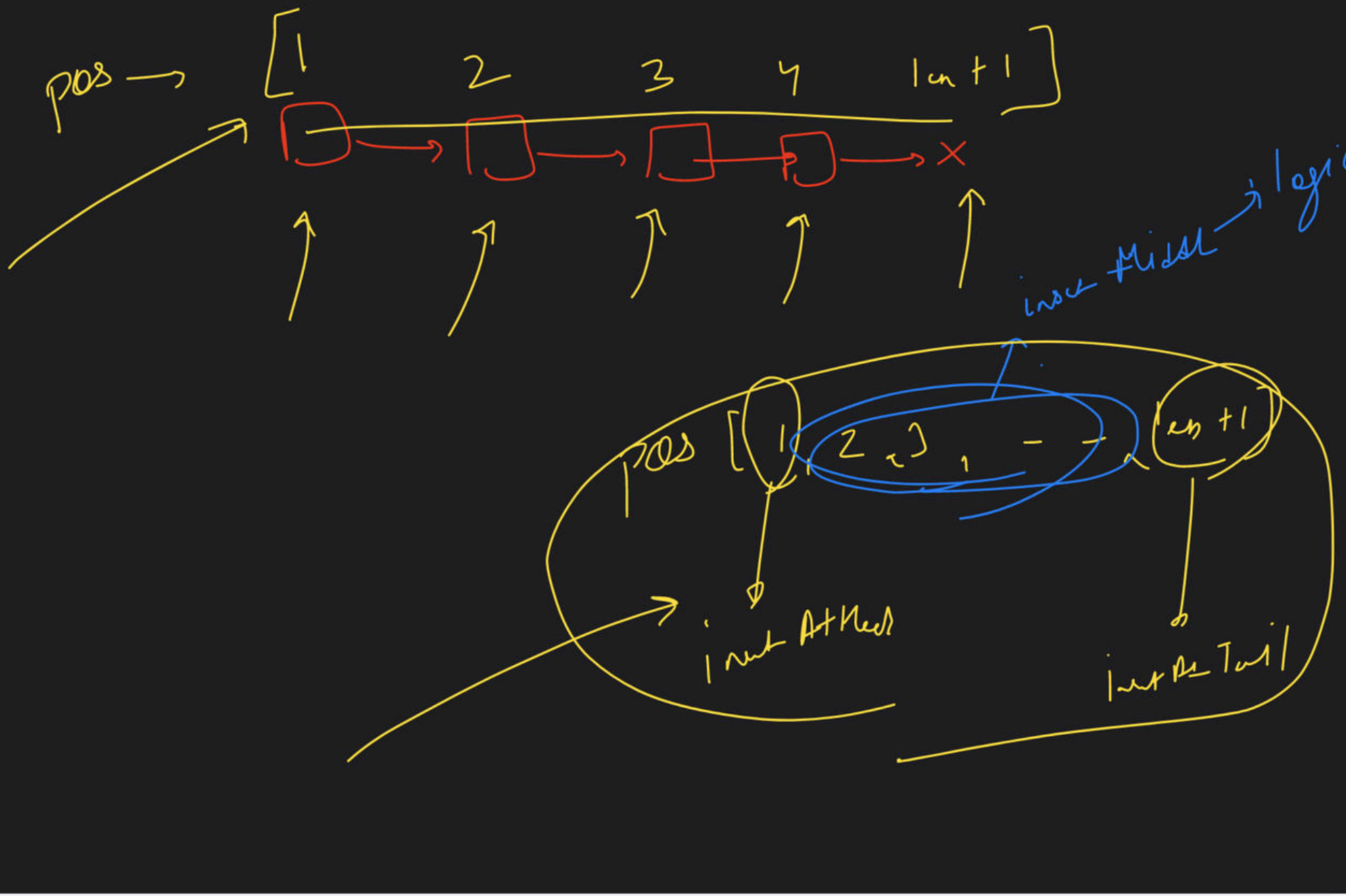
if (pos == len + 1) → insertAtTail()

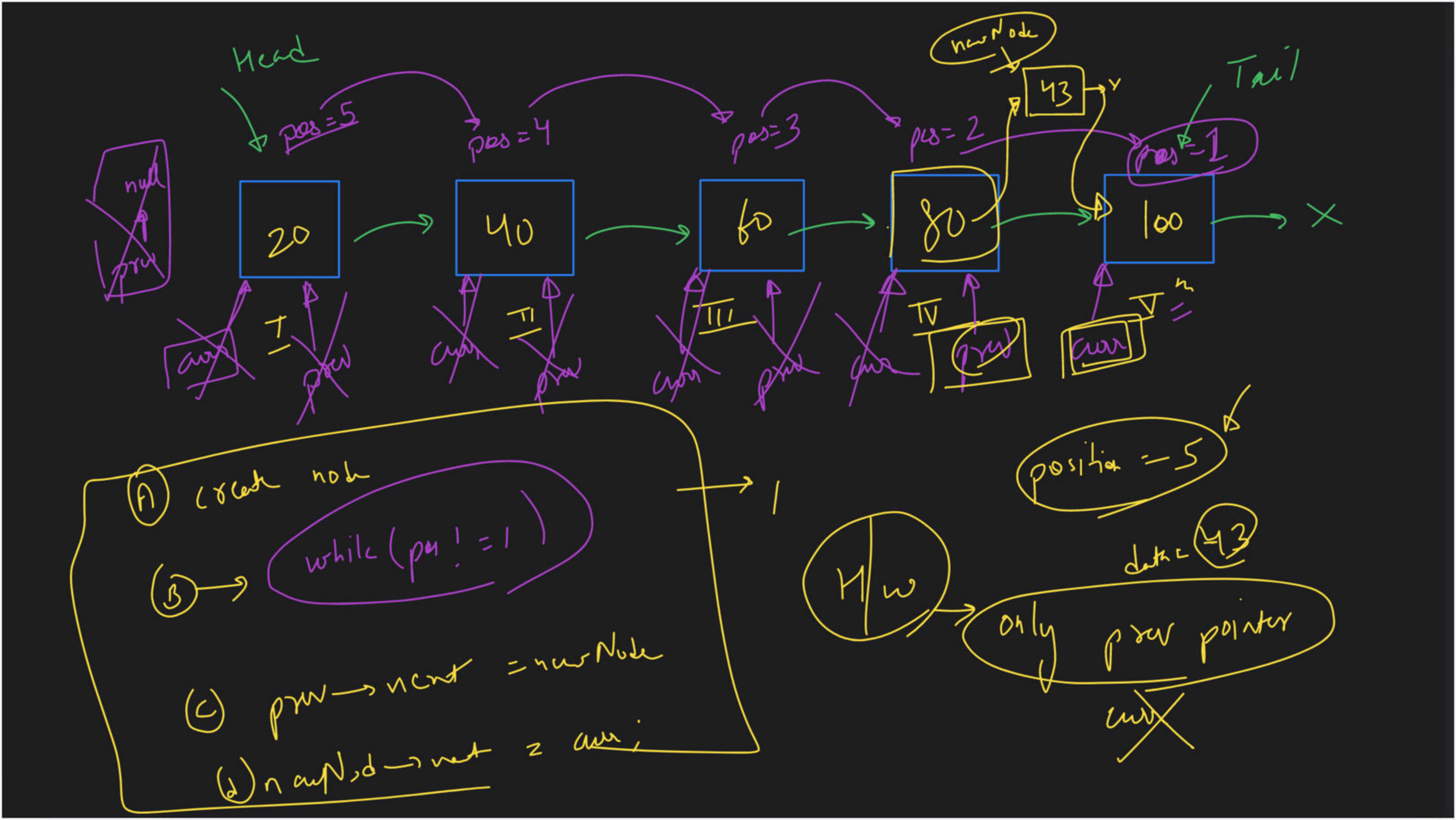


len = 3

pos = 4th

pos = len + 1


pos →
1
2
3
4
len+1
x
insertMiddle → logic
pos
1
2
3
len+1
insertAtHead
insertAtTail


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