



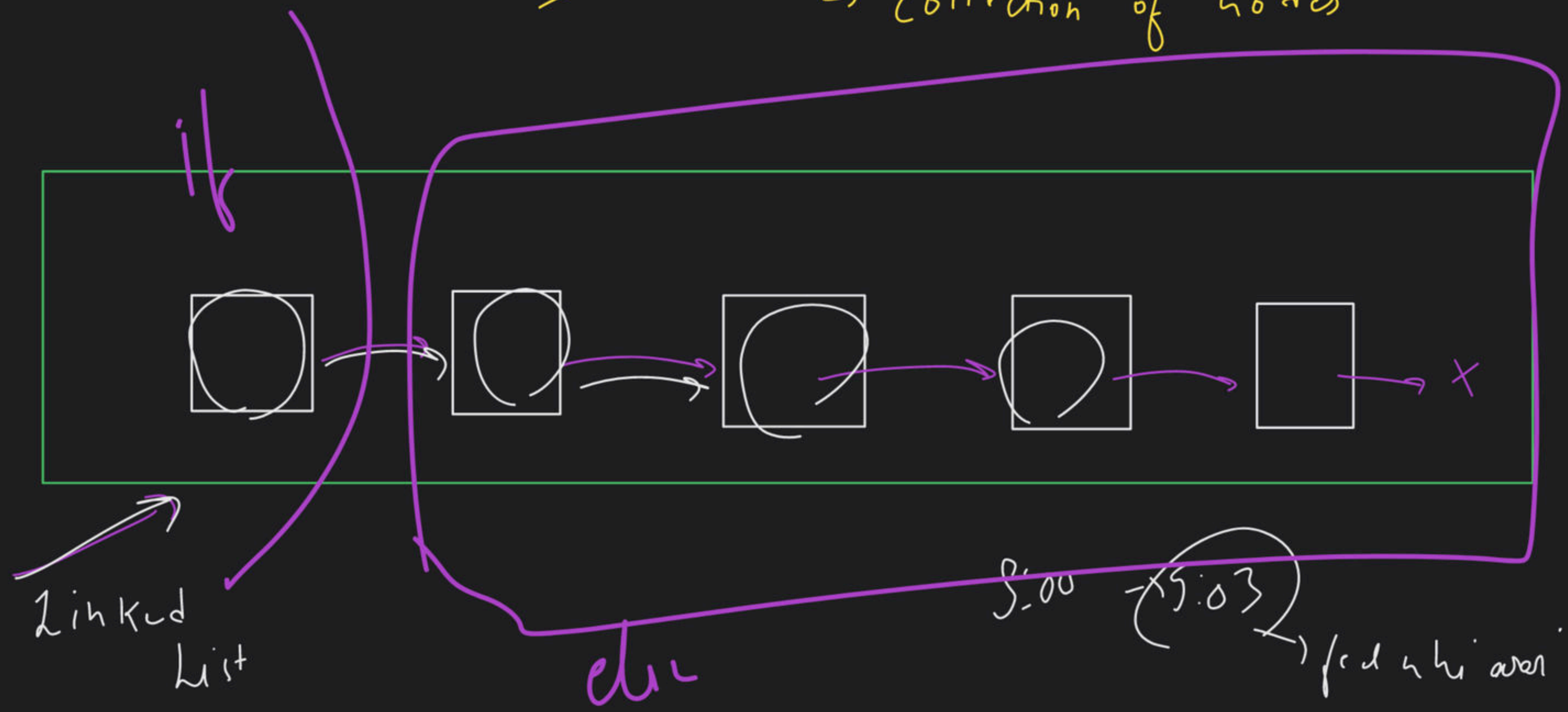
Linked List - I

Foundation Course on Data Structures & Algorithm - Part I

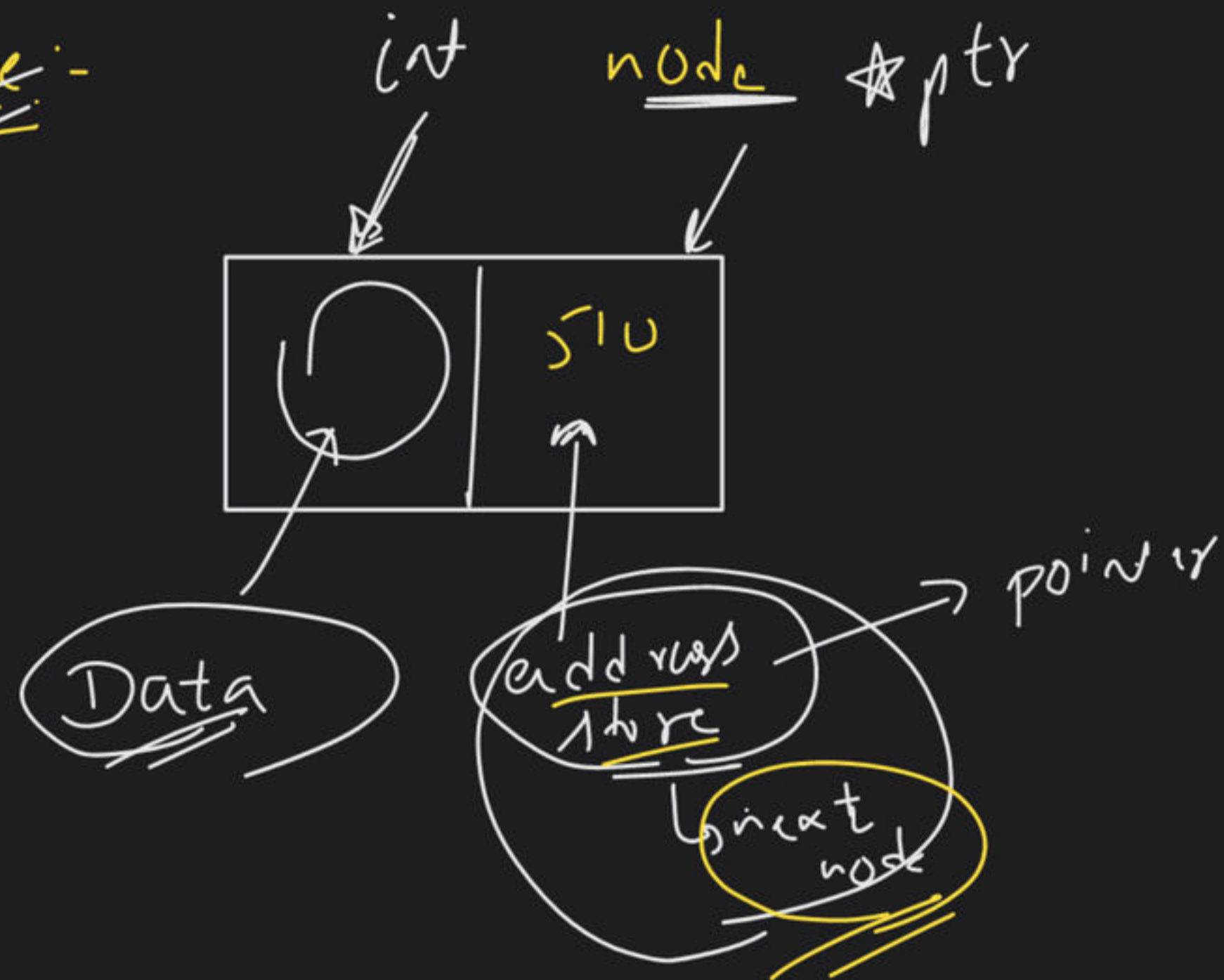
→ Linked List : Linear data structure

echo
↓
yay

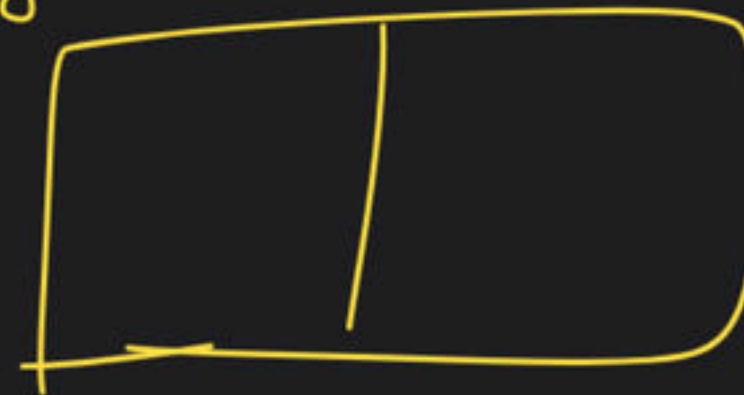
→ collection of nodes



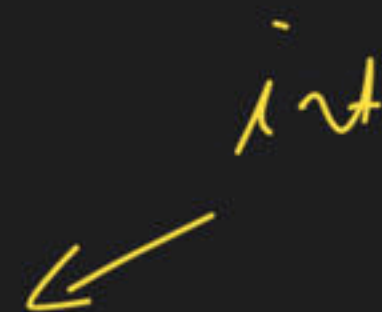
Node:-



double *t;

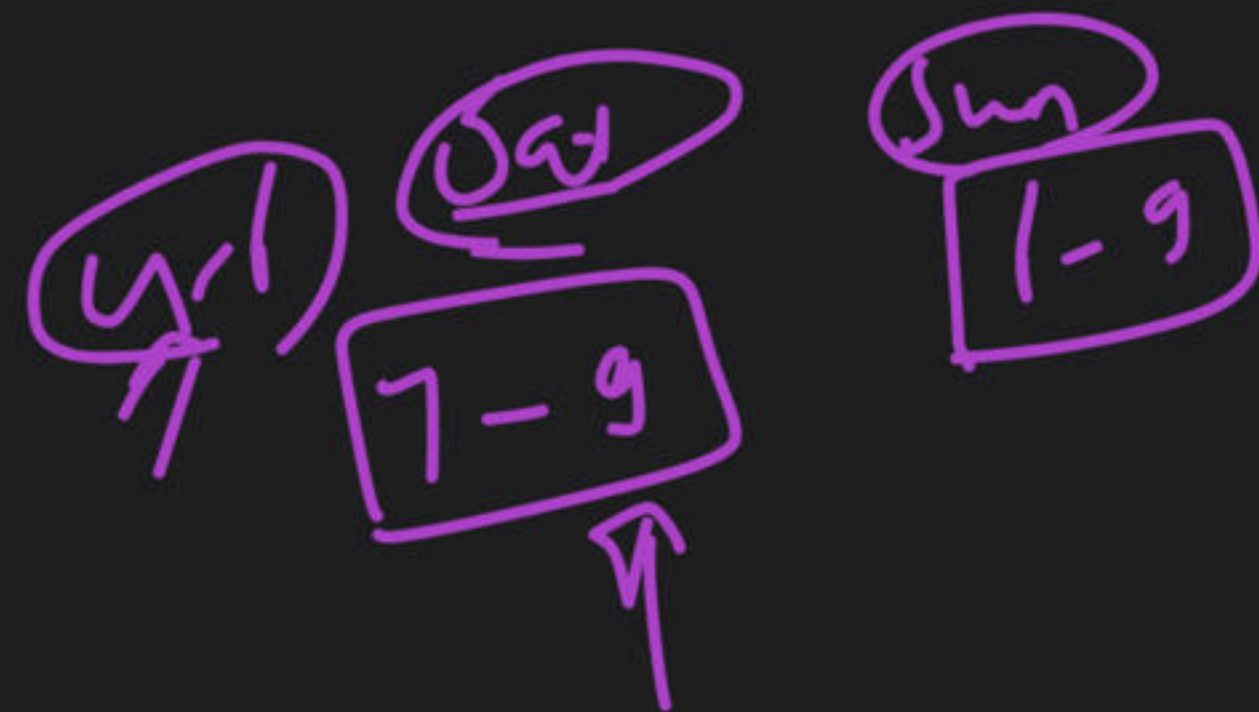


int *ptr

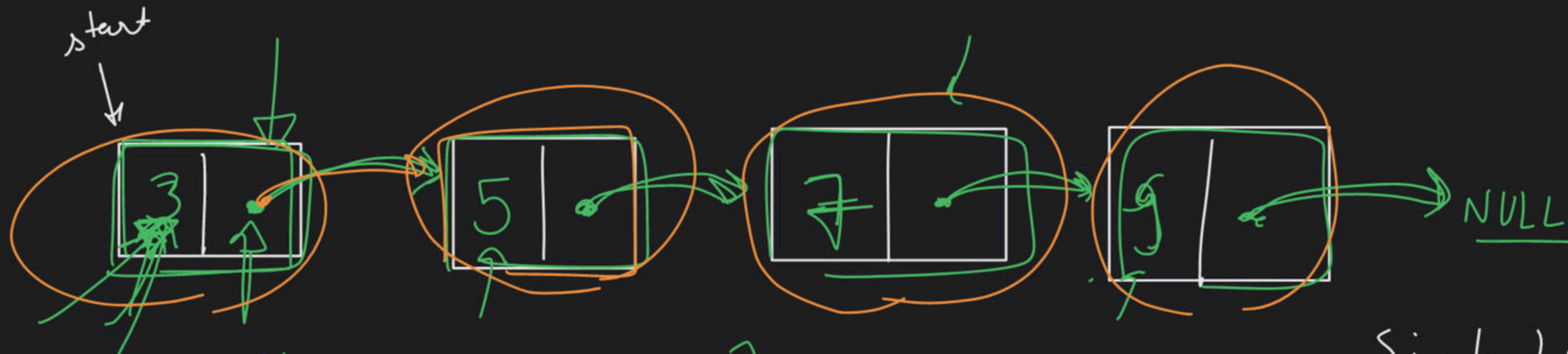



```
class Node
{
    int data;
    Node* next;
}
```

} → Data Member



H/W

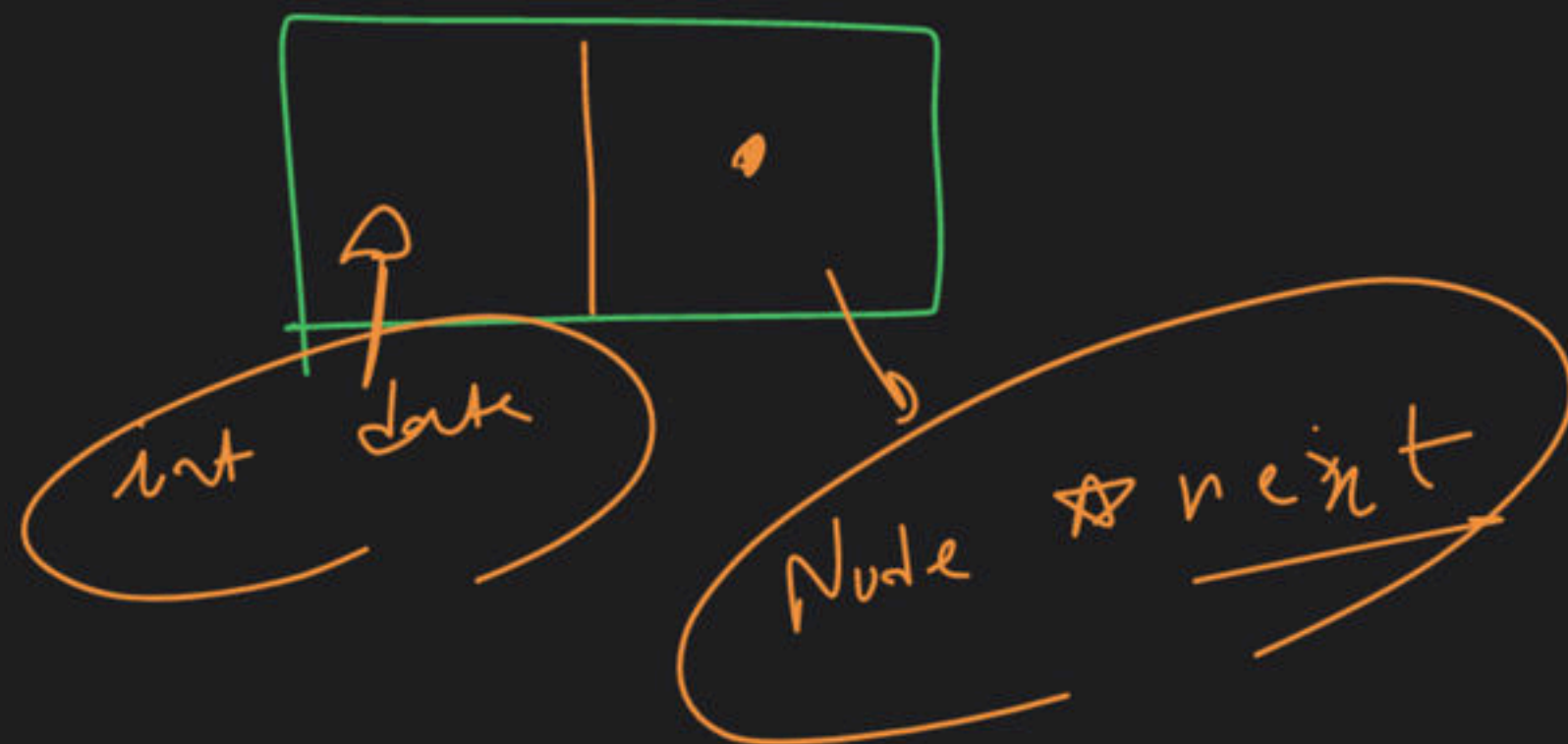


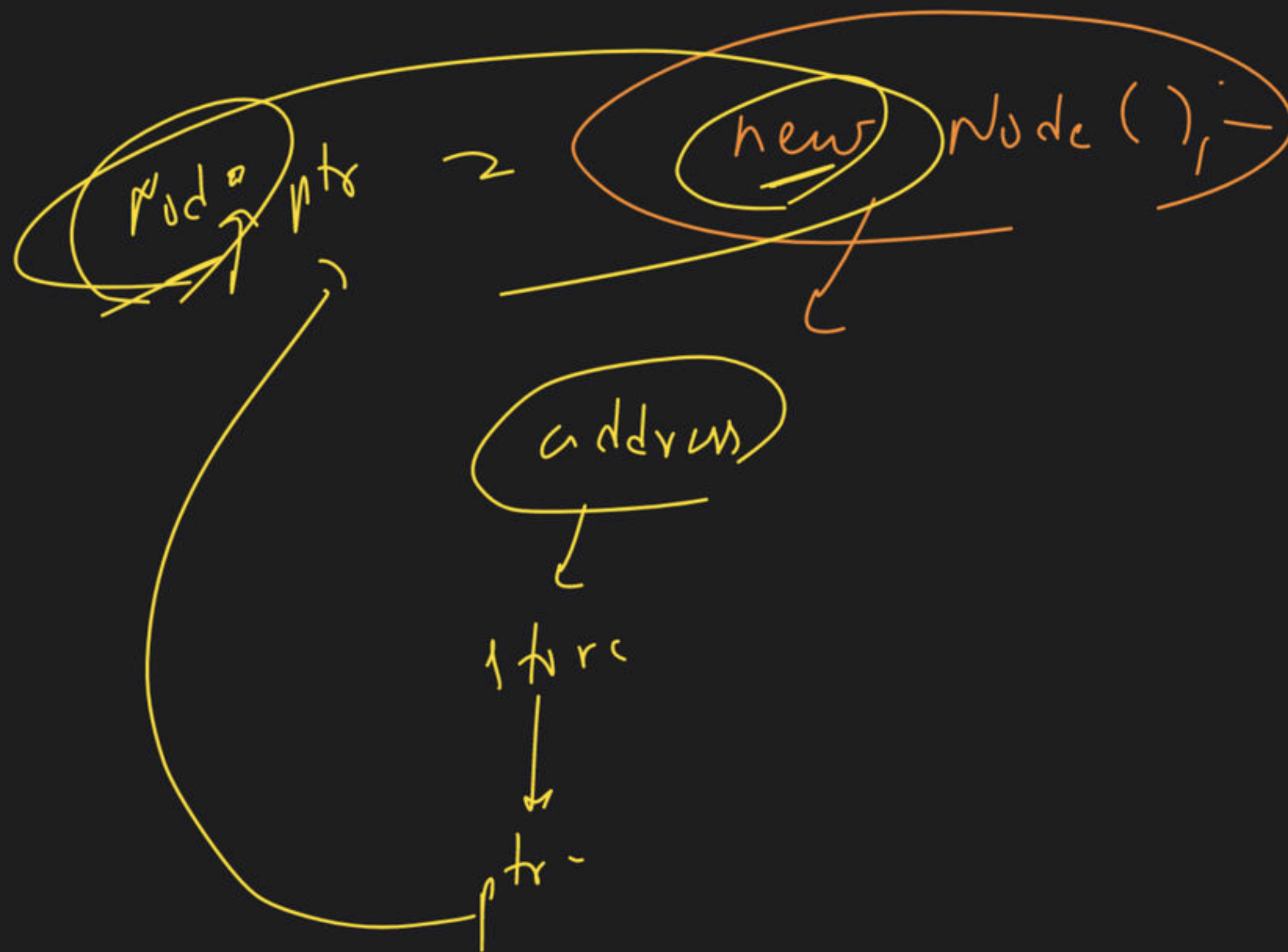
address
of
node

LL

Singly Linked
List

pointer → store
address





→ obj → Node

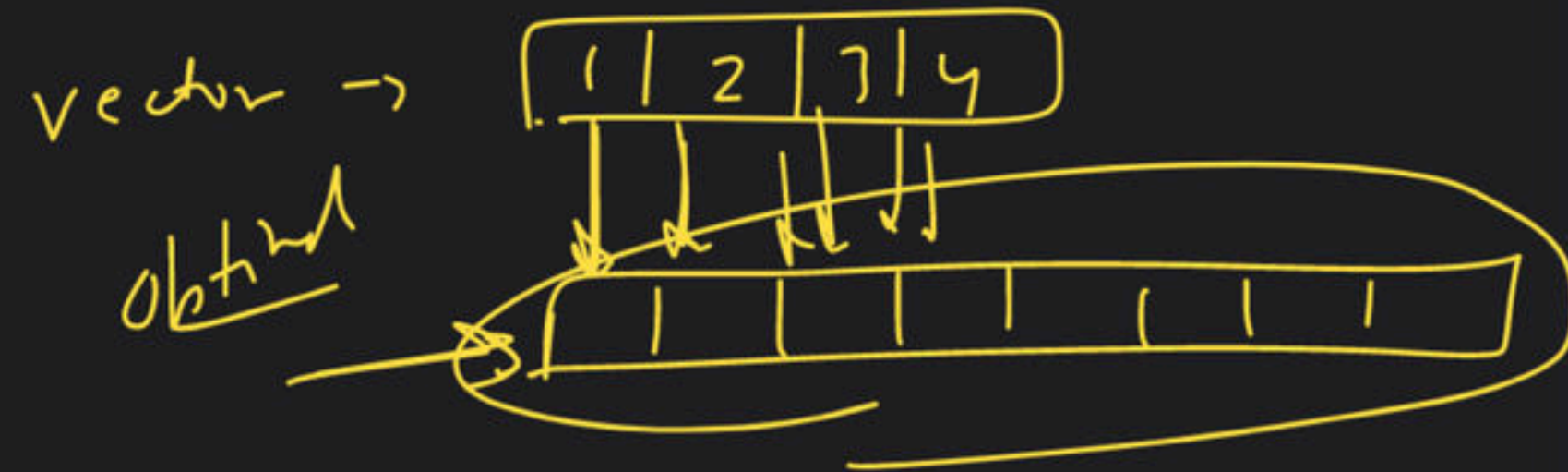
LL → (why) → ?

array
vector

arr [15]



(contiguous)
continuous



Adv:-

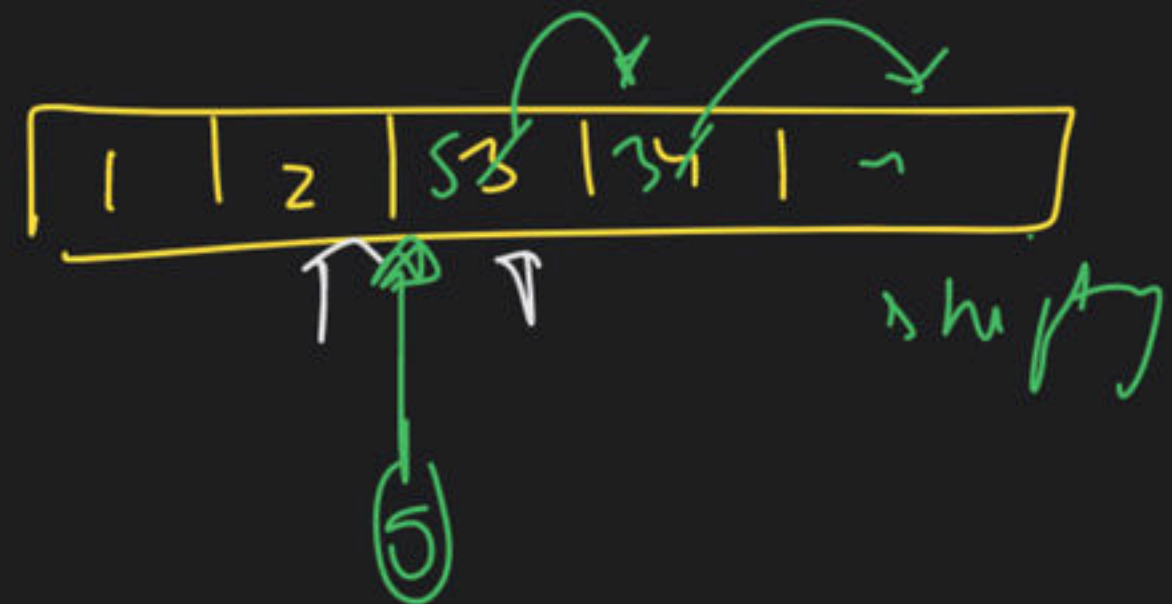
→ grow/shrink

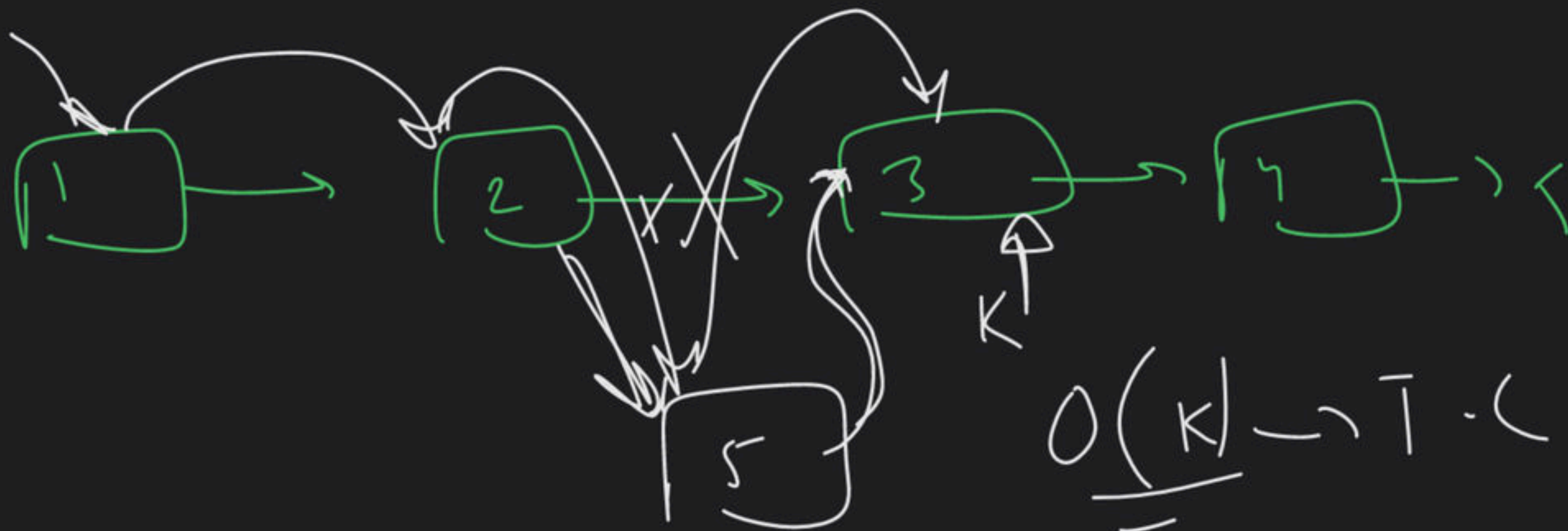
dynamic

Disadv:-
to 9

→ does not require

continuous memory

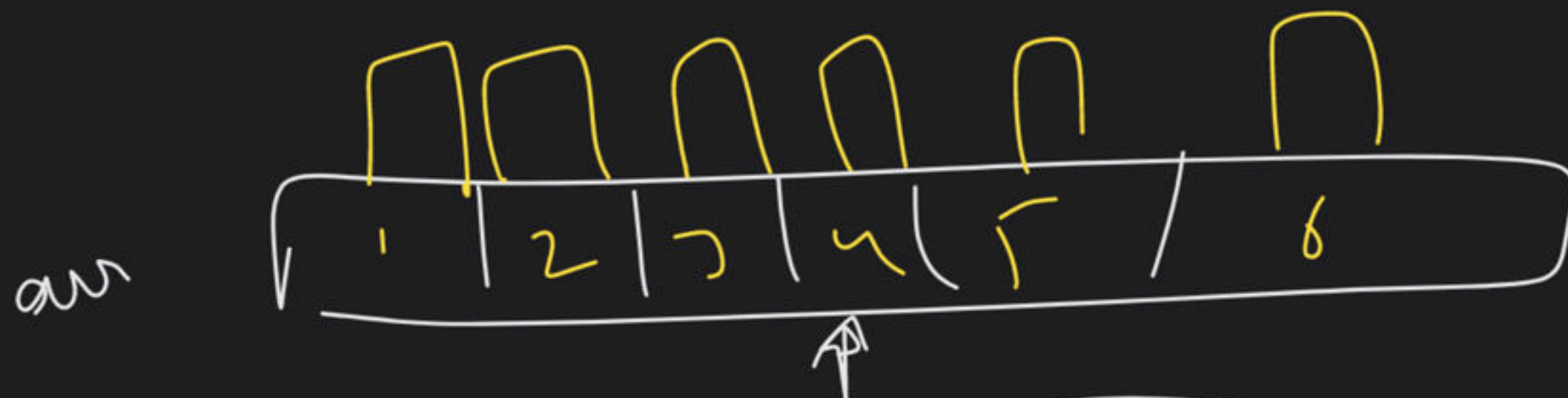




$O(K)$ \rightarrow T.C

LL
↓
ans / D in ans

Ques
T.C



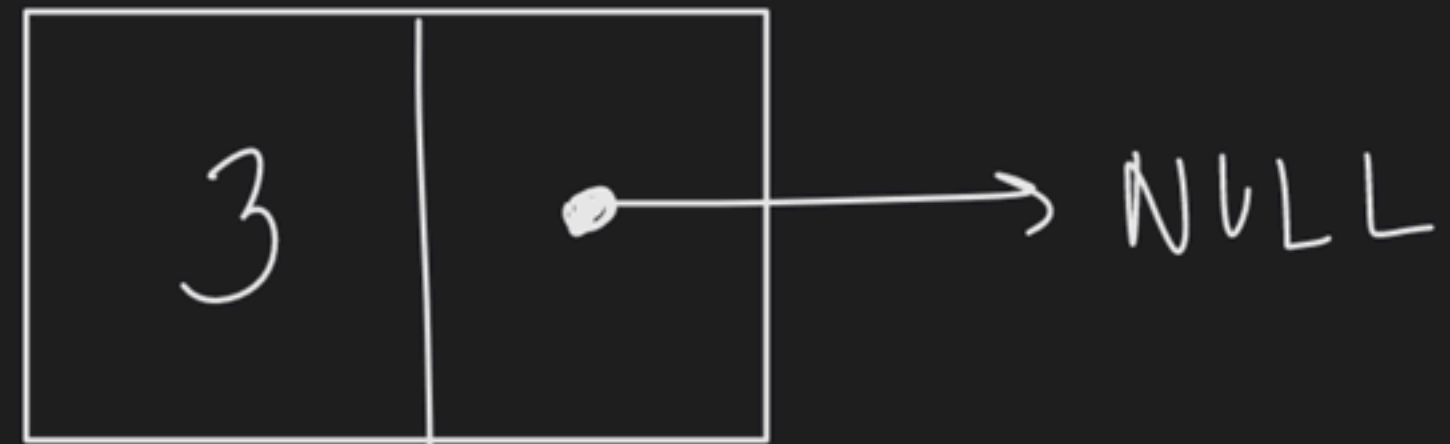
L.C
↔

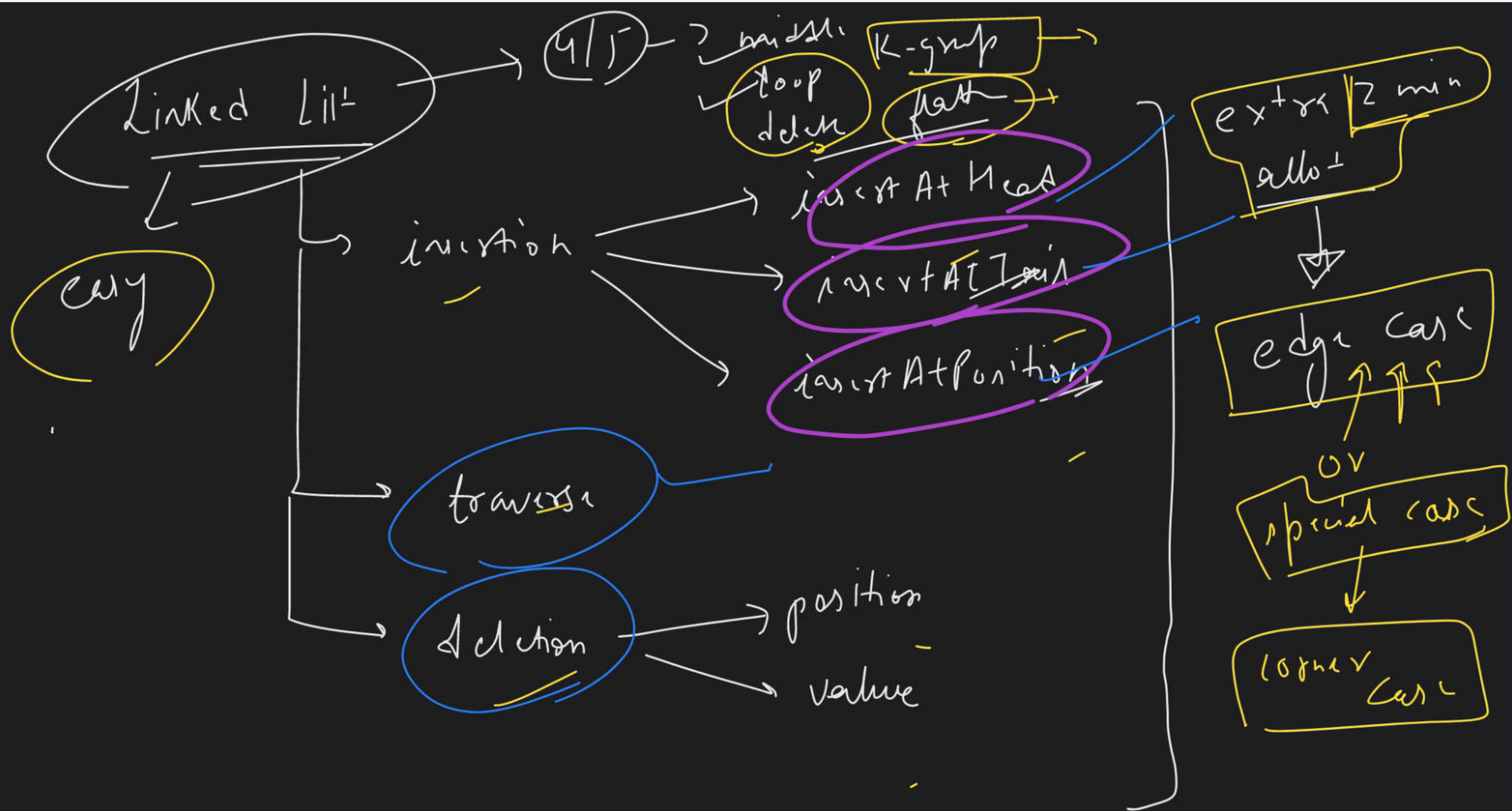
ans[3] \rightarrow 8(1)



ans
itr =

Node * first = new Node(3)

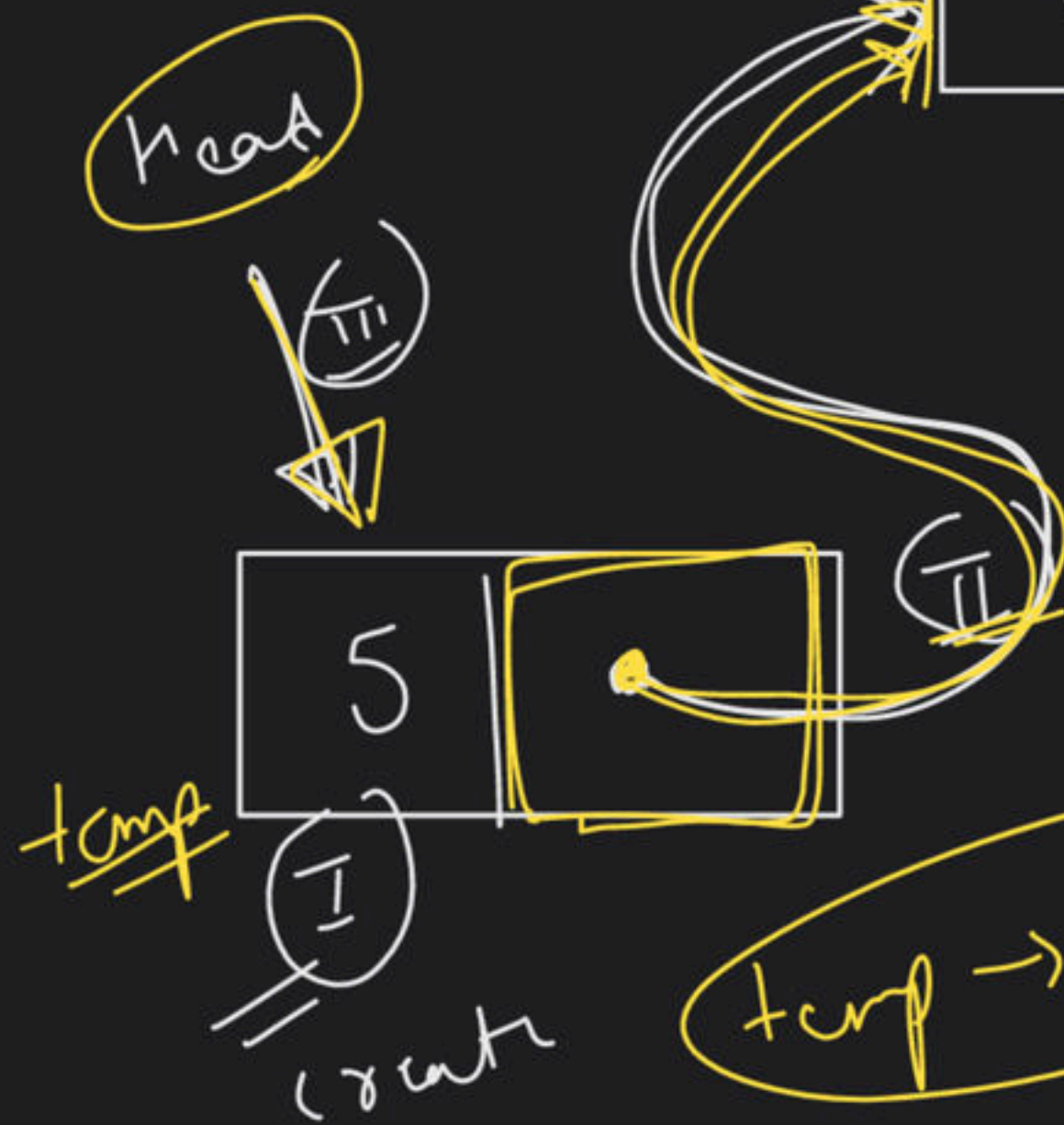




↳ insert at Head



insertAtHead(5)
↑

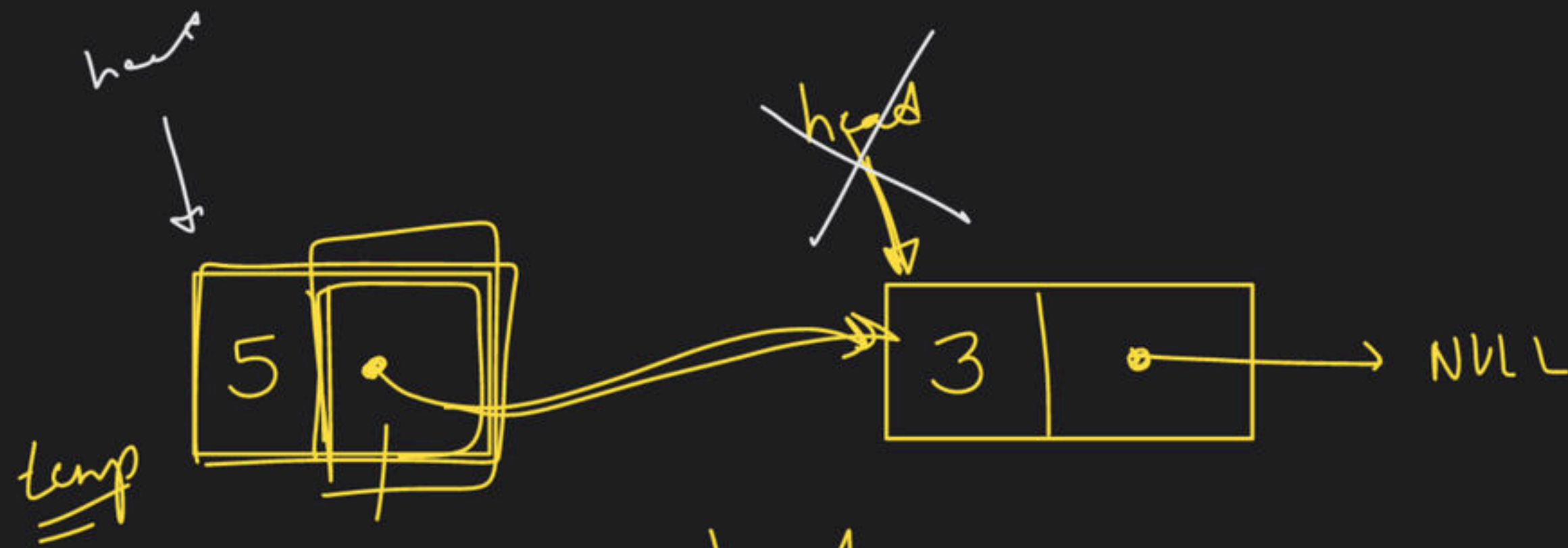


temp → next = head

Node * temp = new Node(5);
temp → next = head
head = temp

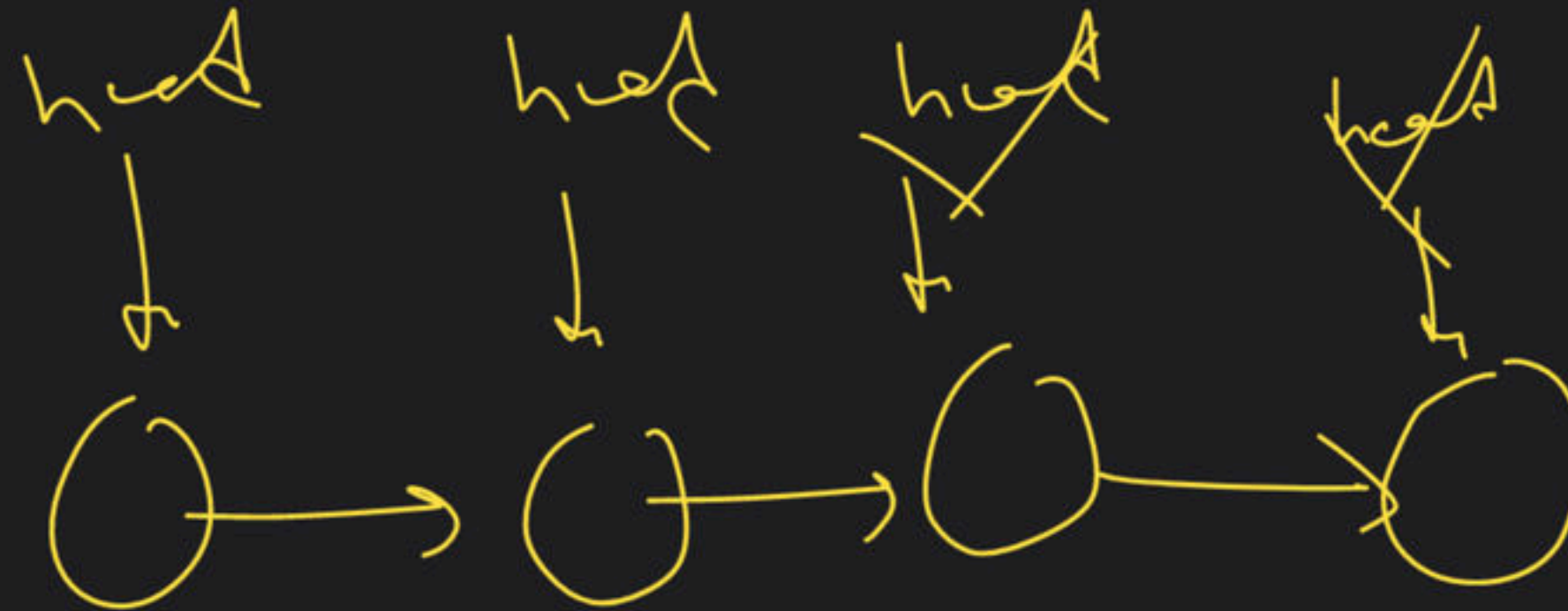
insert AtHead(5)

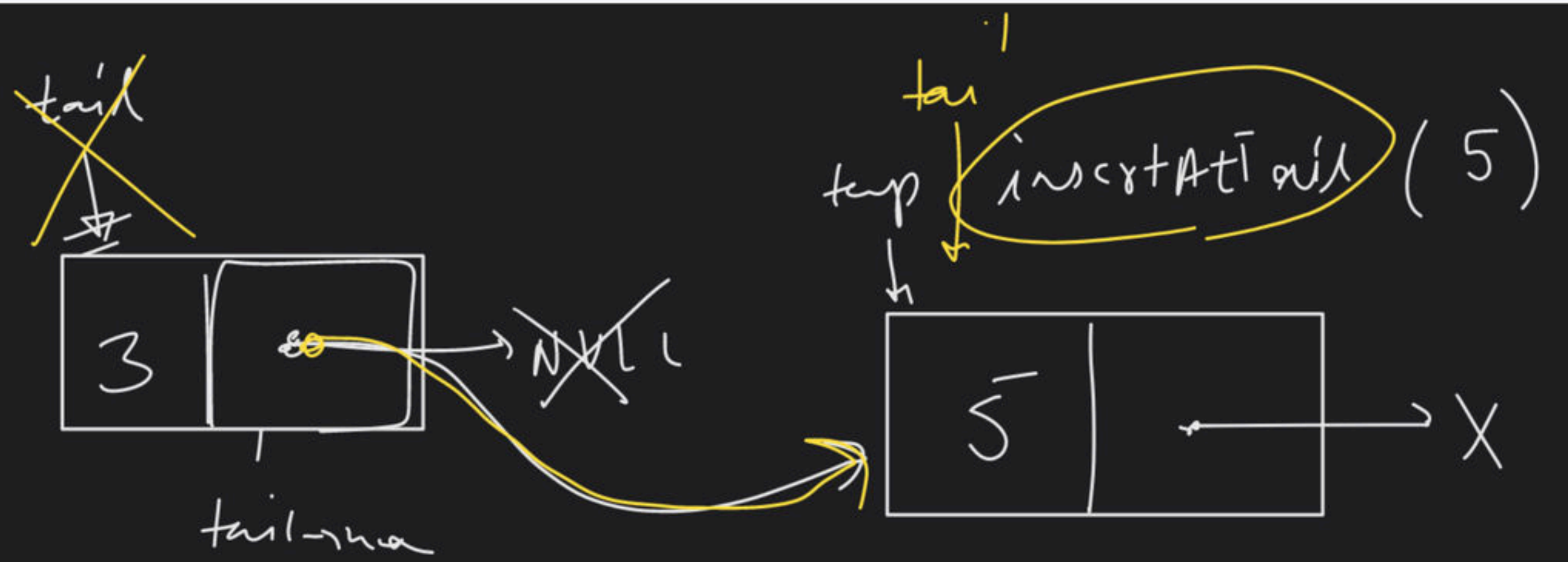
(I) create 1 node



temp->next = head
(II)

head = temp
(III)

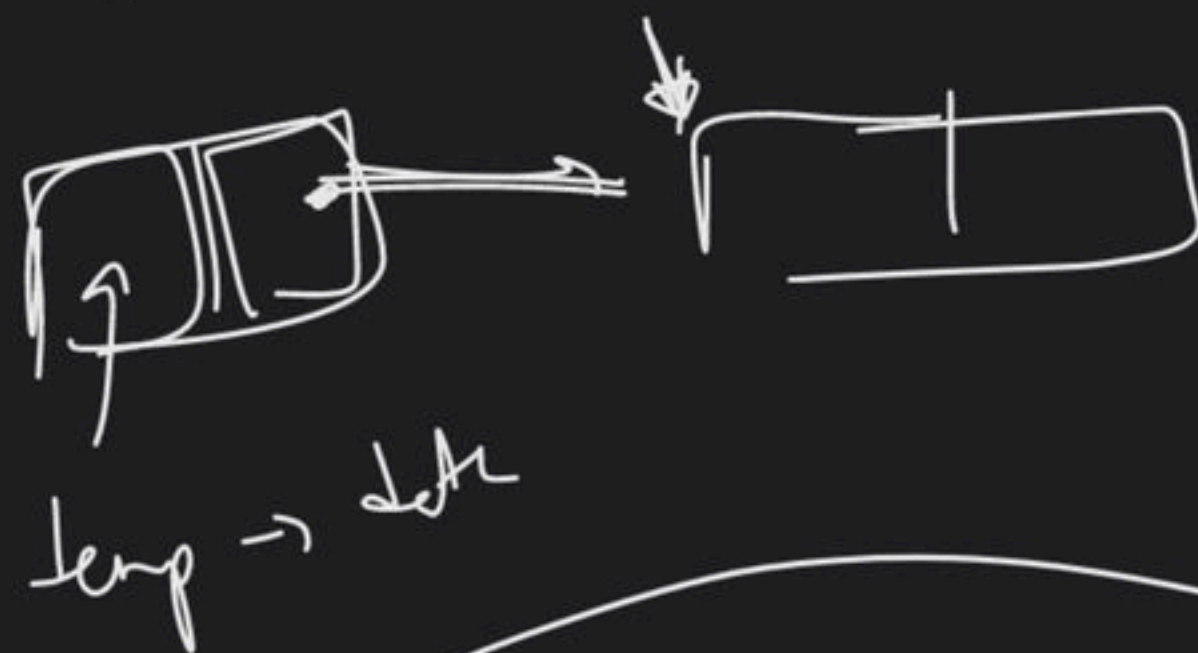
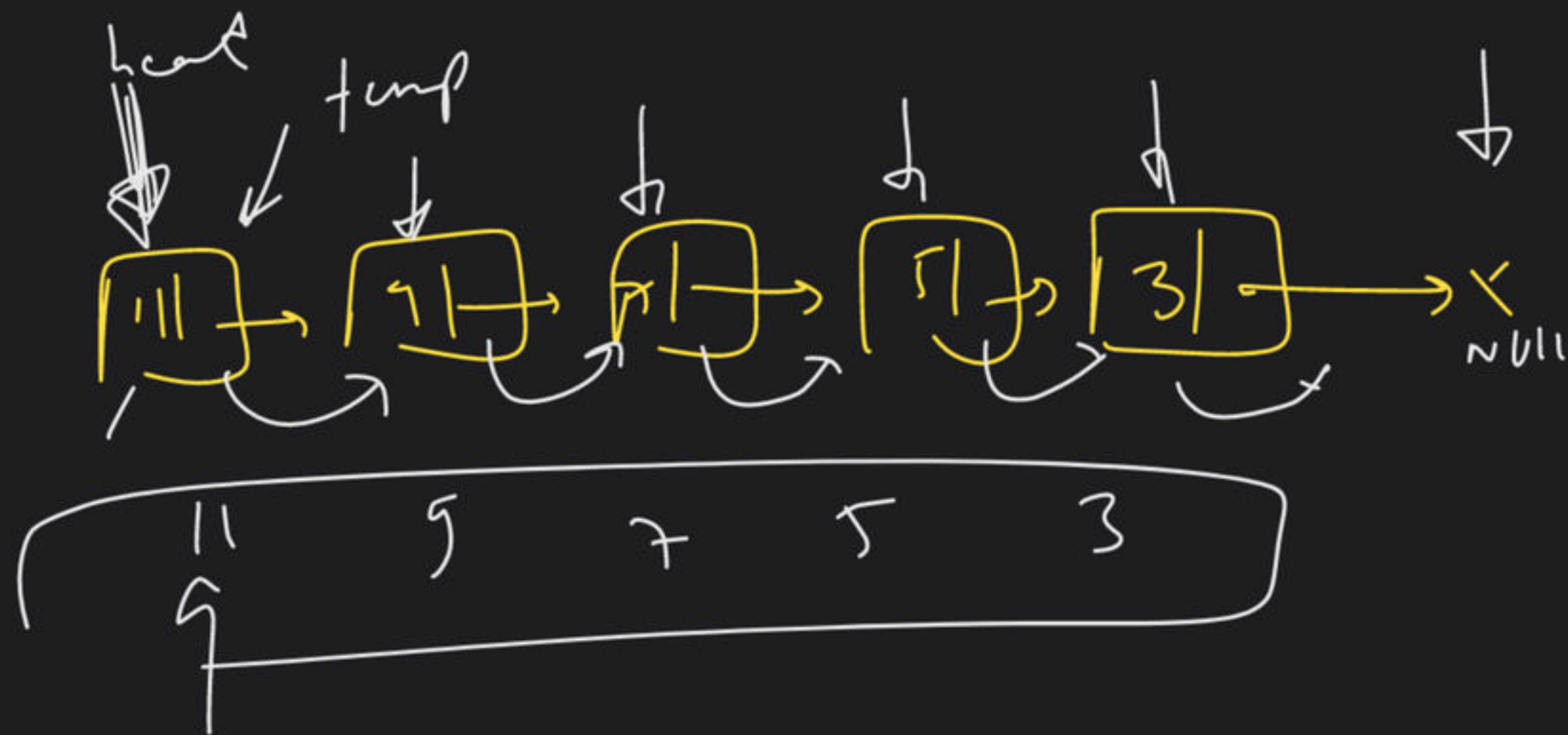




(I) Creation

(II) $\text{tail} \rightarrow \text{next} = \text{temp}$

(III) $\text{tail} = \text{temp}$



$temp = temp \rightarrow next$

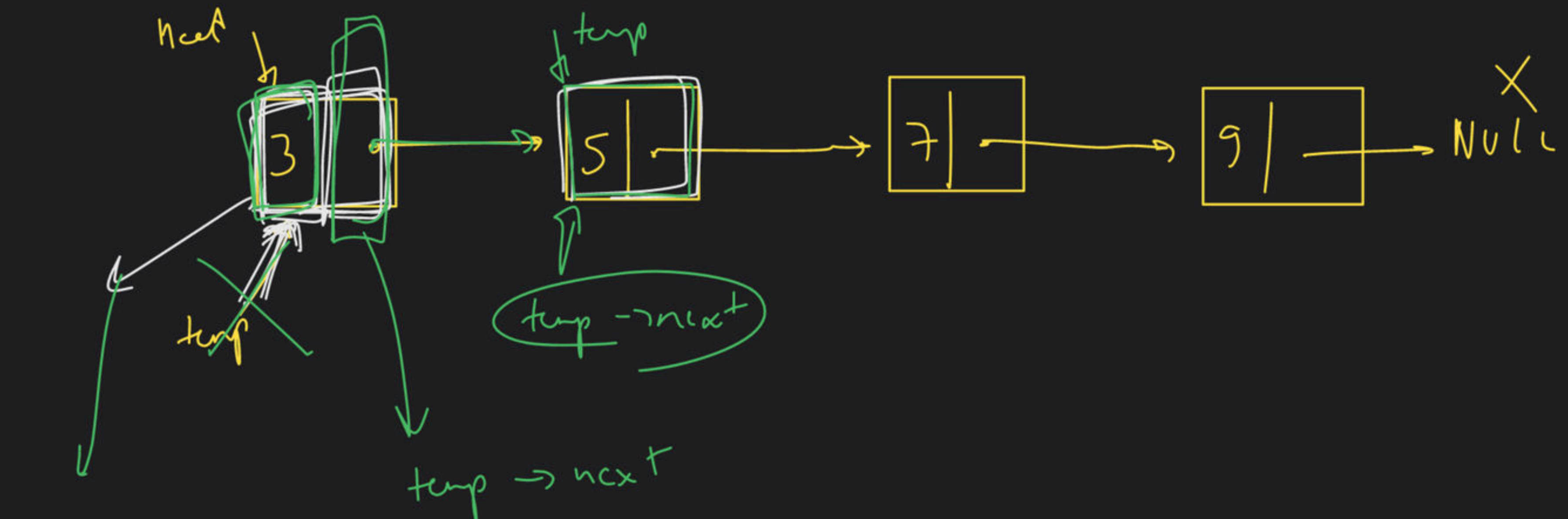
traverse (head)

Node * temp = head

while (temp != NULL)

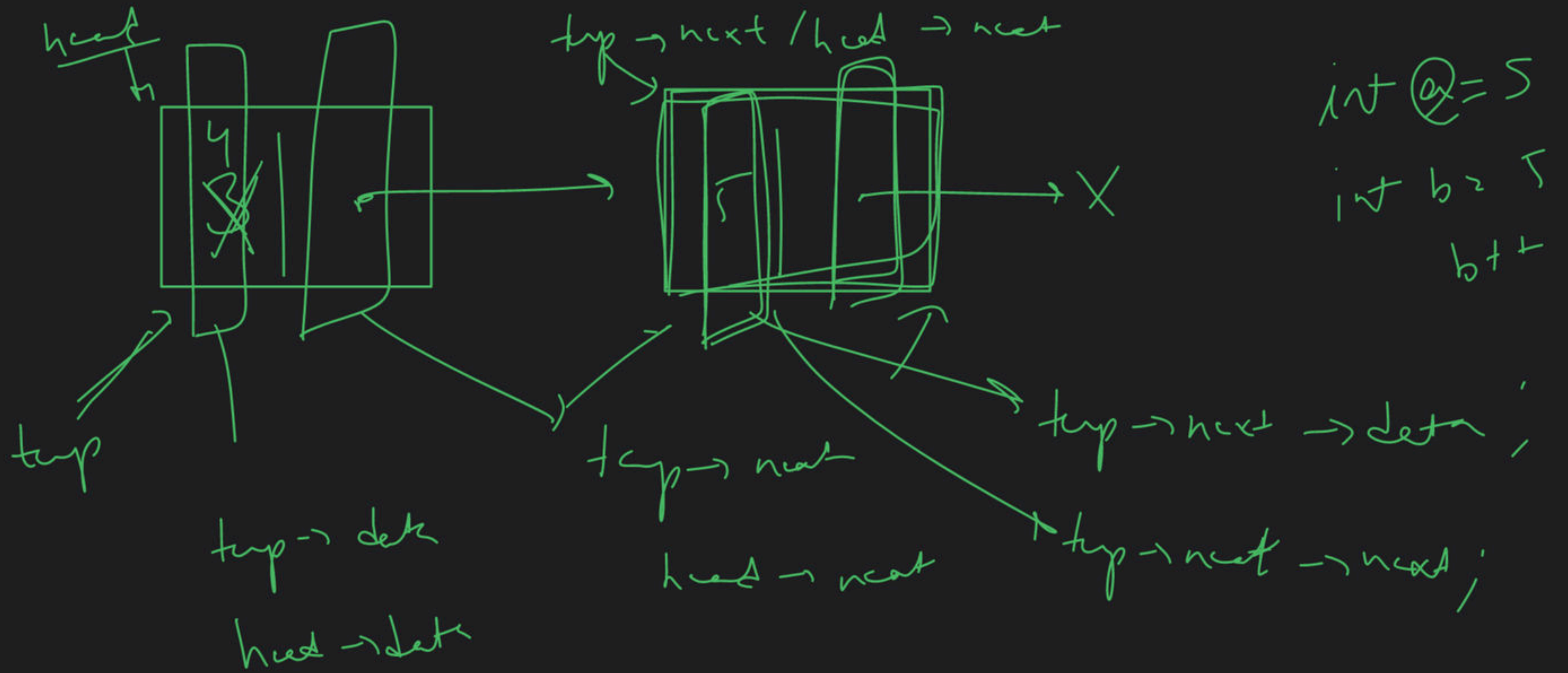
{
 cout << temp->data;
 temp = temp->next;
}

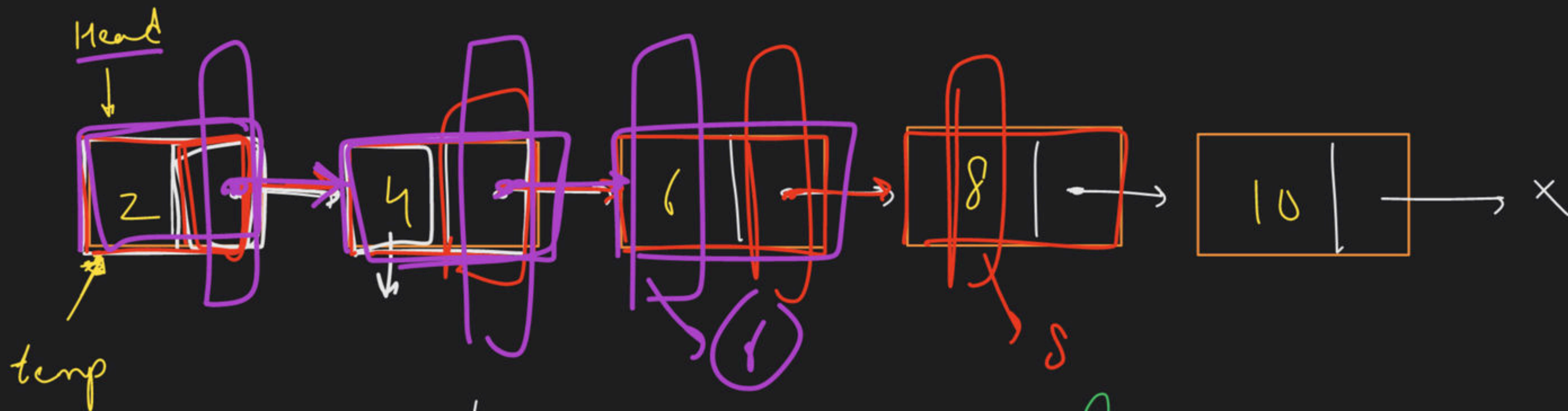
}



temp -> data

temp = temp -> next



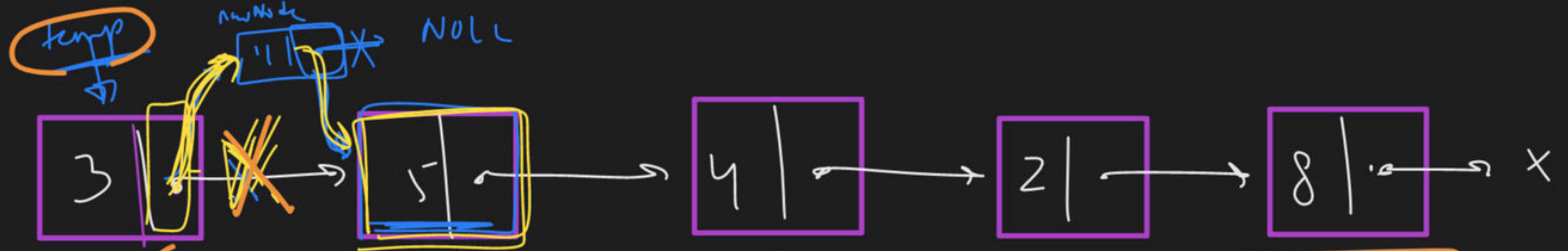


o/p → head → next → data → ? 4

temp → next → next → next → data → ?

head → next → next → data → ?

99%



position \rightarrow 2
(K)

11

$i = 5$

6th \rightarrow this update

$n = 1$

if ($n == 1$)
insertAt(1)
connection

(i)

creation of Node

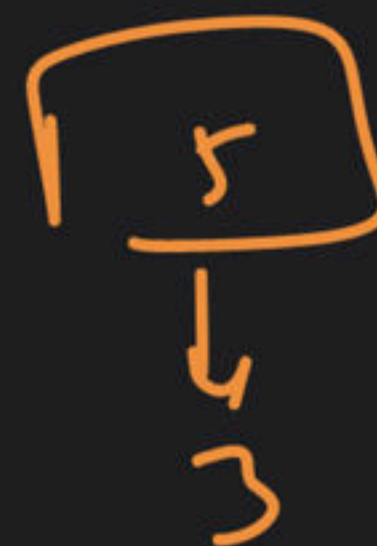
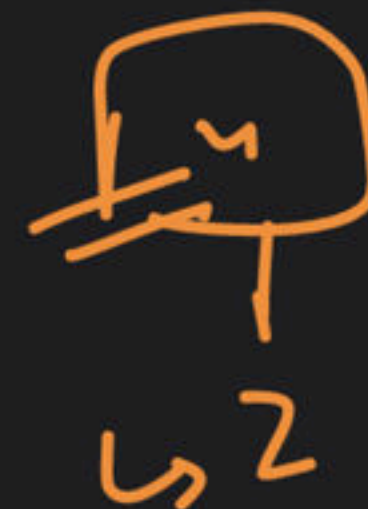
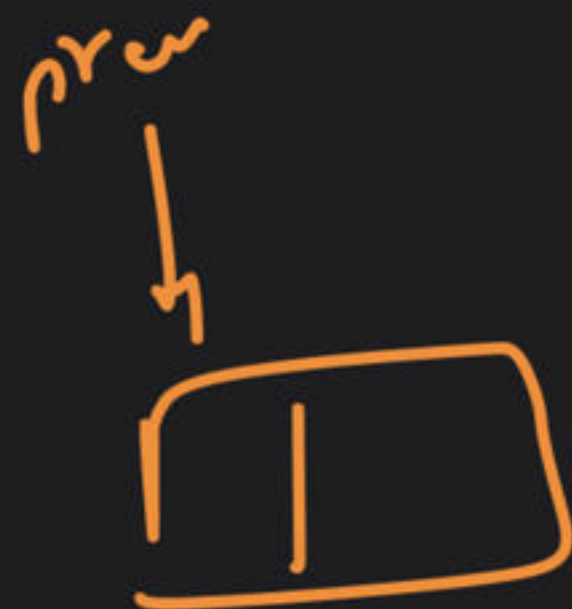
(ii)

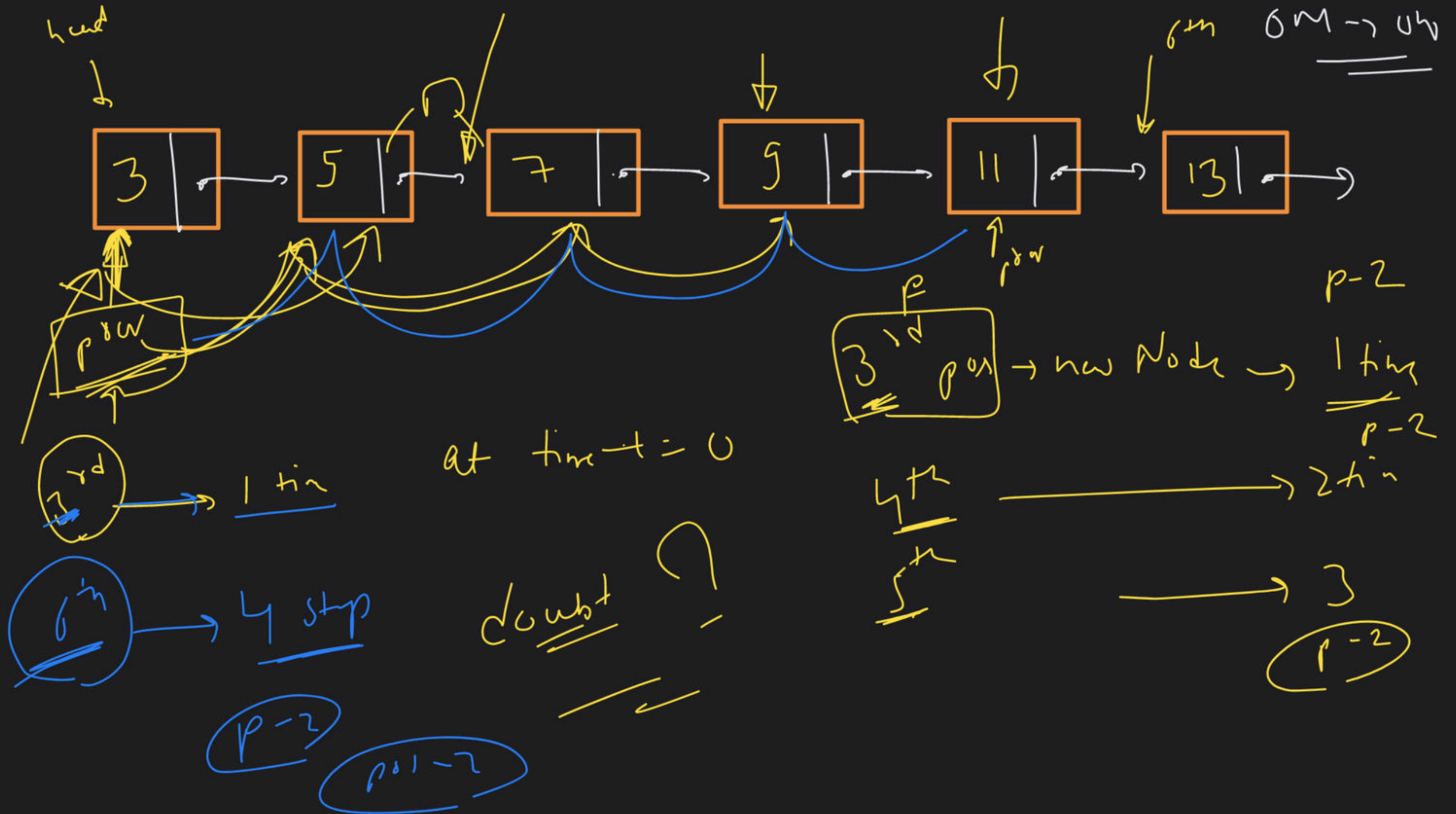
traverse $K-1$ node

(iii)

$newNode \rightarrow next = temp \rightarrow next$
 $temp \rightarrow next = newNode$

}
order





else
{

middle
tail

prev → next = temp → next;

temp → next = NULL

delete temp;

empty link

2 mid

① ~~head~~
tail
update

Done

① tail updation → v/w

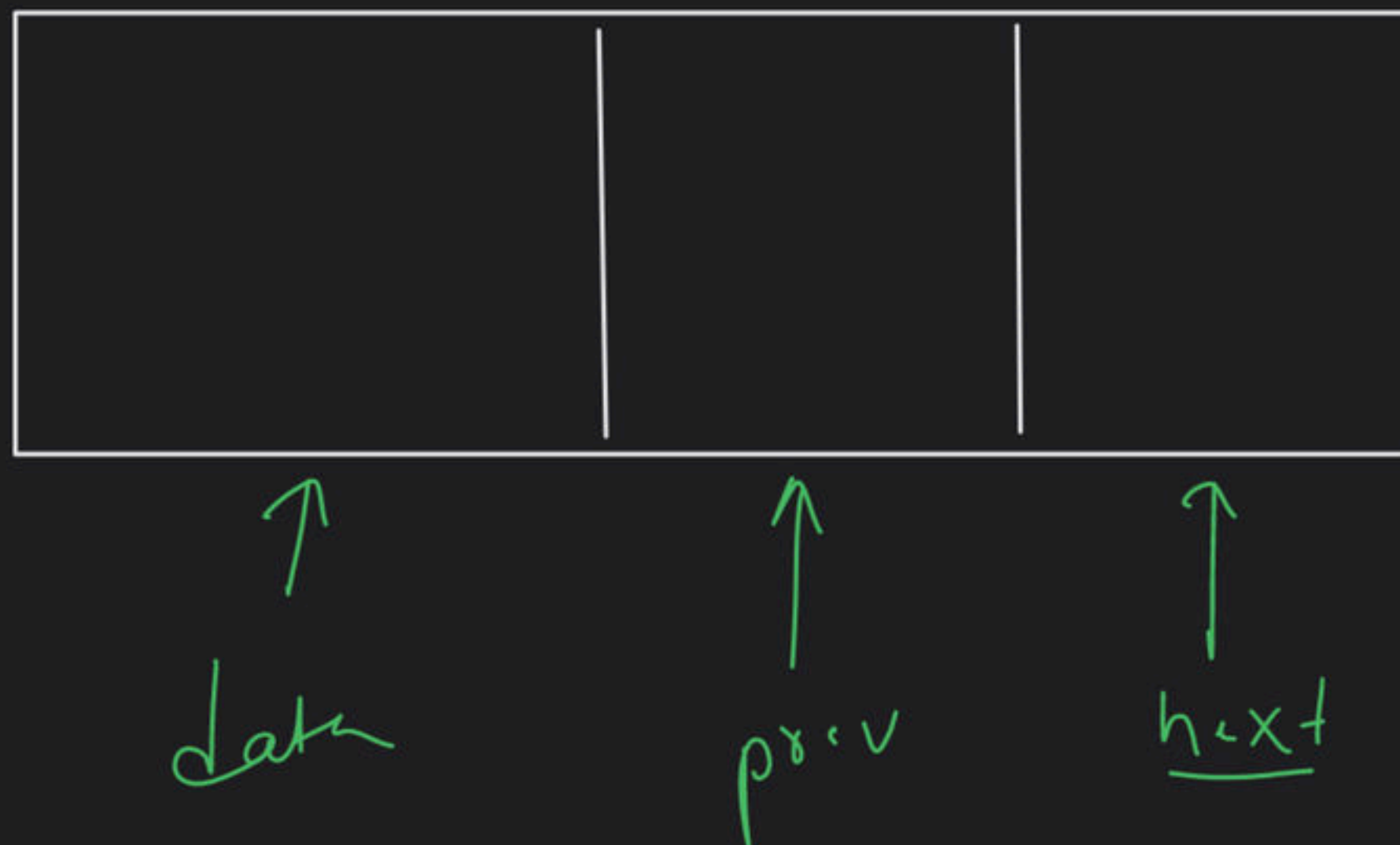
② if we do not find value -

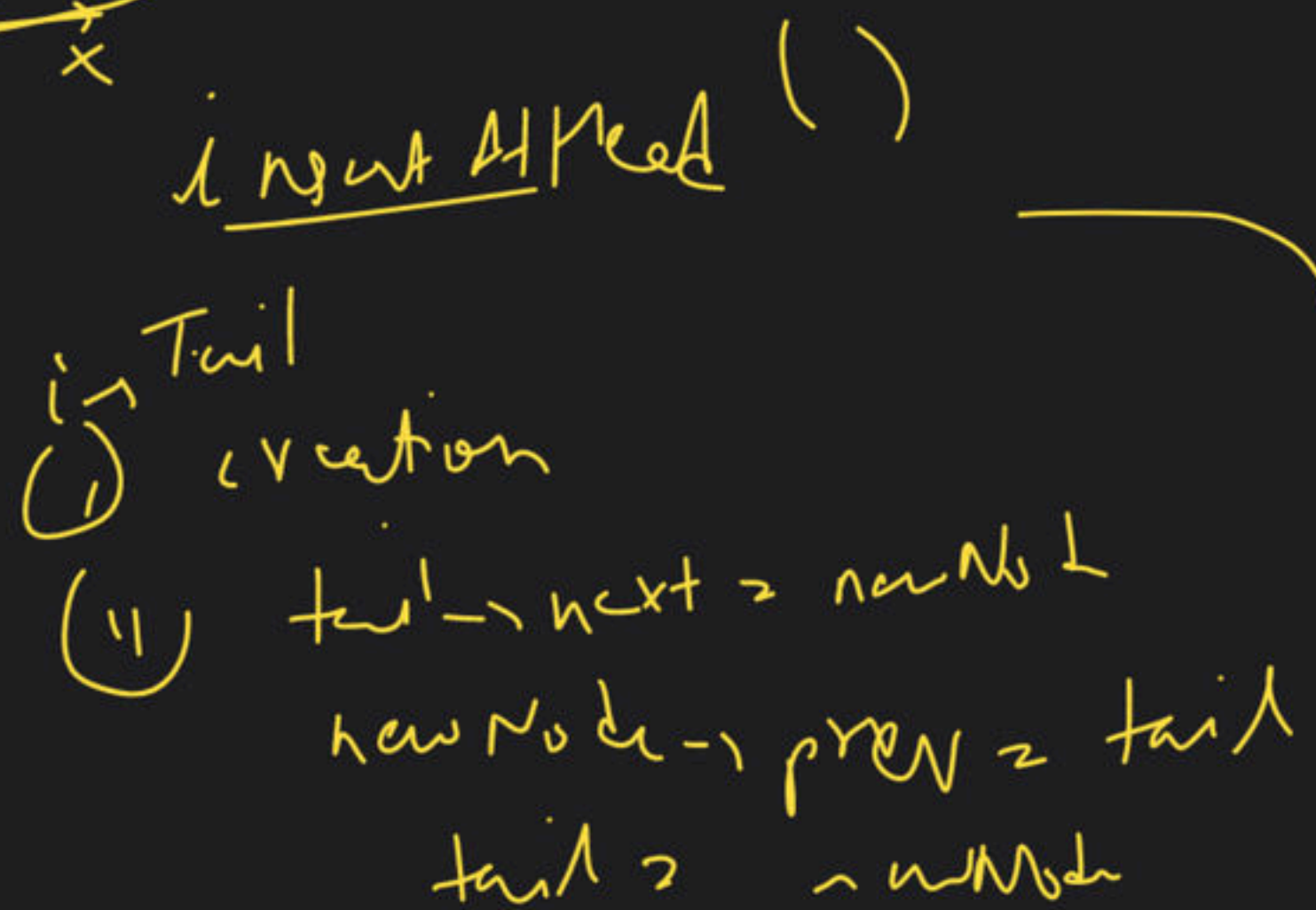
③ ~~length~~

length

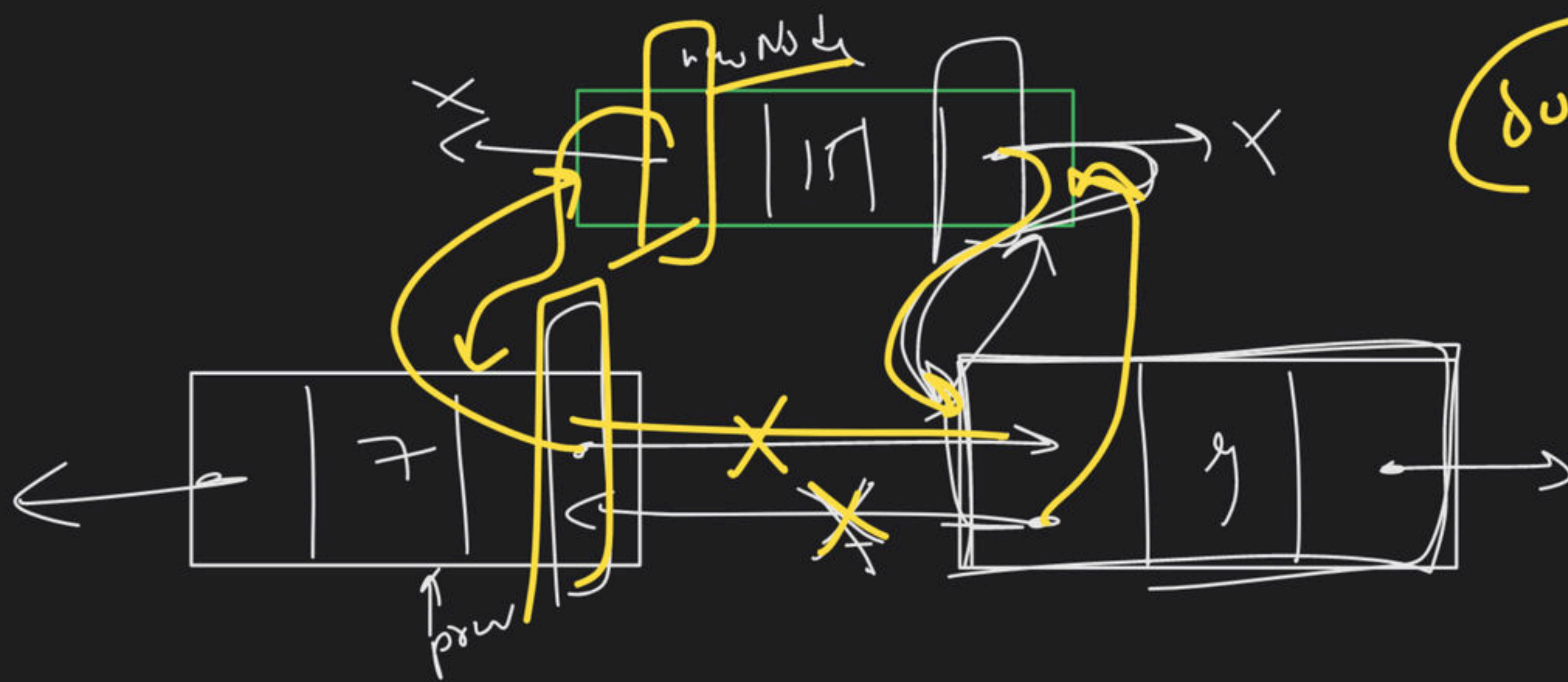
Doubly Linked List

```
class  
{  
    int data;  
    Node * prev;  
    Node * next;  
};
```





(11) $\text{newNode} \rightarrow \text{next} = \text{head}$
 $\text{head} \rightarrow \text{prev} = \text{newNode}$
 $\text{head} = \text{newNode}$



(1)

create

(11)

$\text{newNode} \rightarrow \text{next} = \text{prev} \rightarrow \text{next}$

$\text{newNode} \rightarrow \text{next} \rightarrow \text{prev} = \text{newNode}$

$\text{prev} \rightarrow \text{next} = \text{newNode}$

$\text{newNode} \rightarrow \text{prev} = \text{prev}$

detect
&
detect
loop



deletion

DBLL

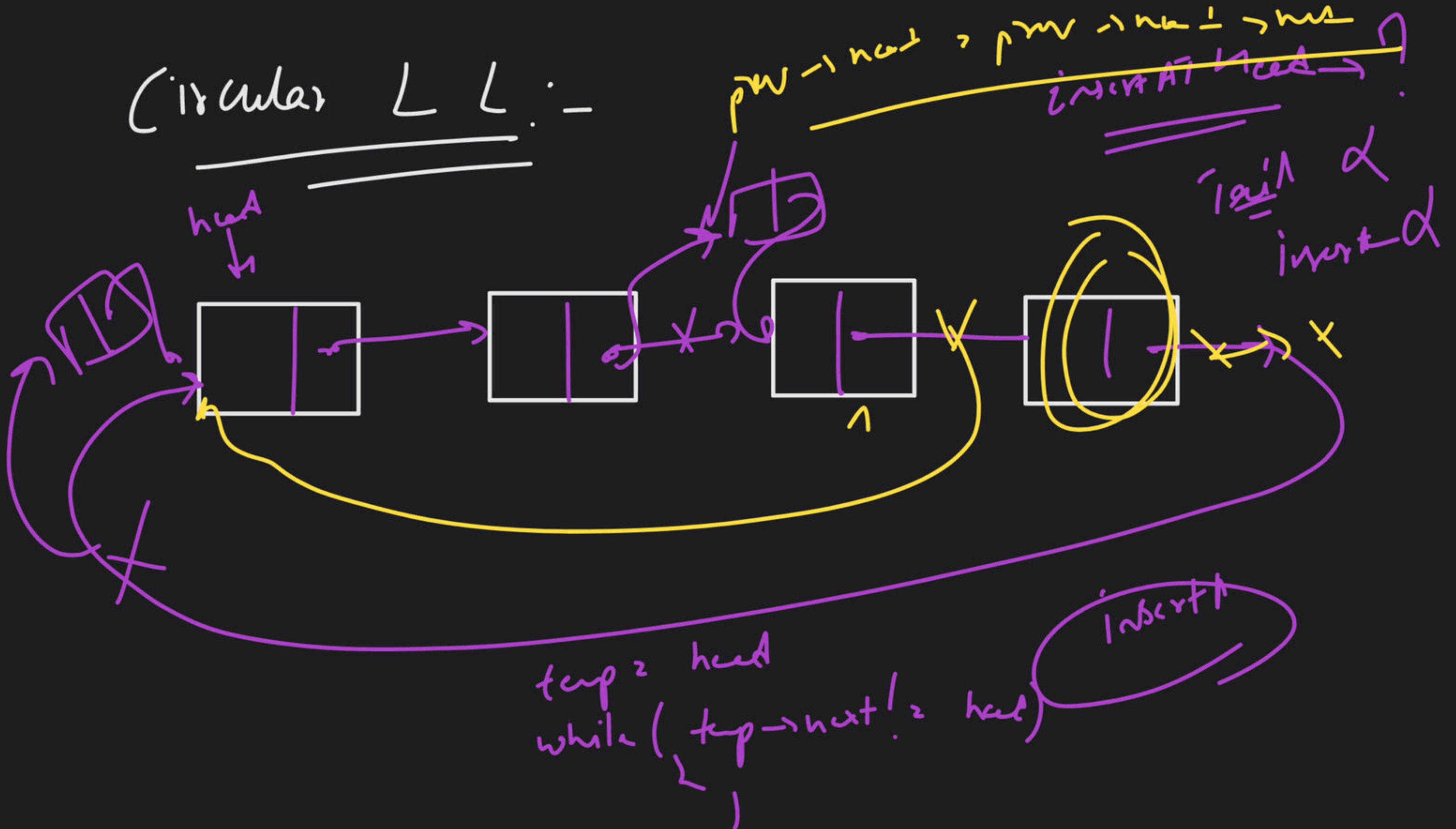
inv

trav

del

} H/w

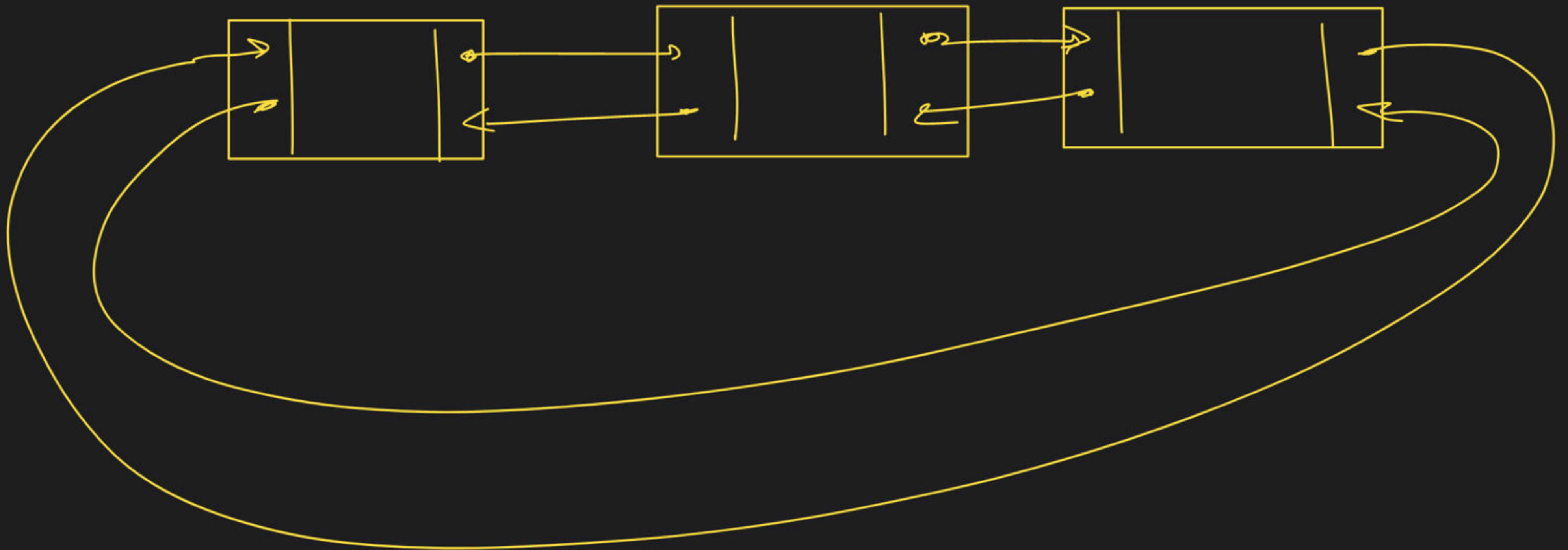
Circular LL :-

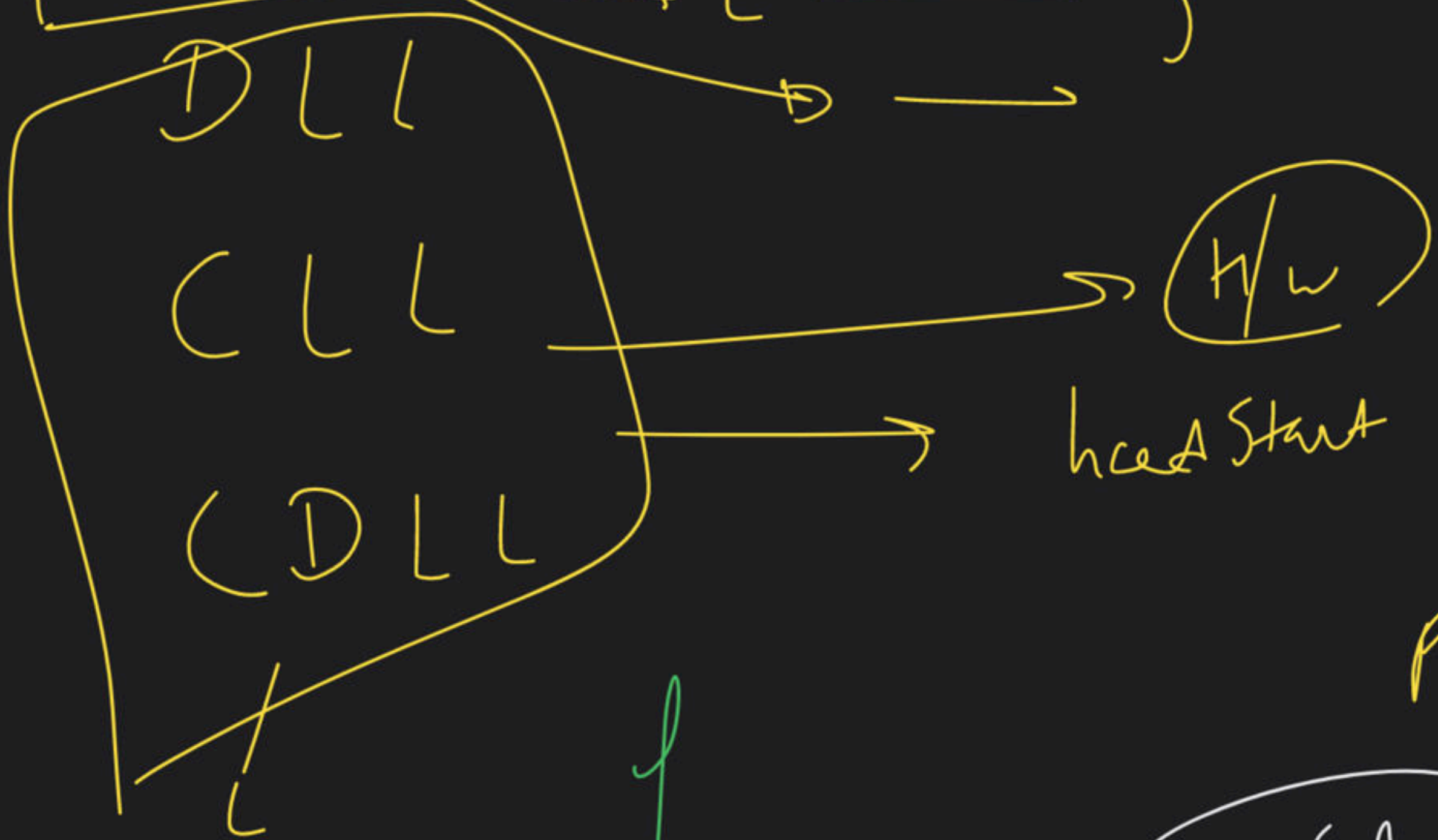
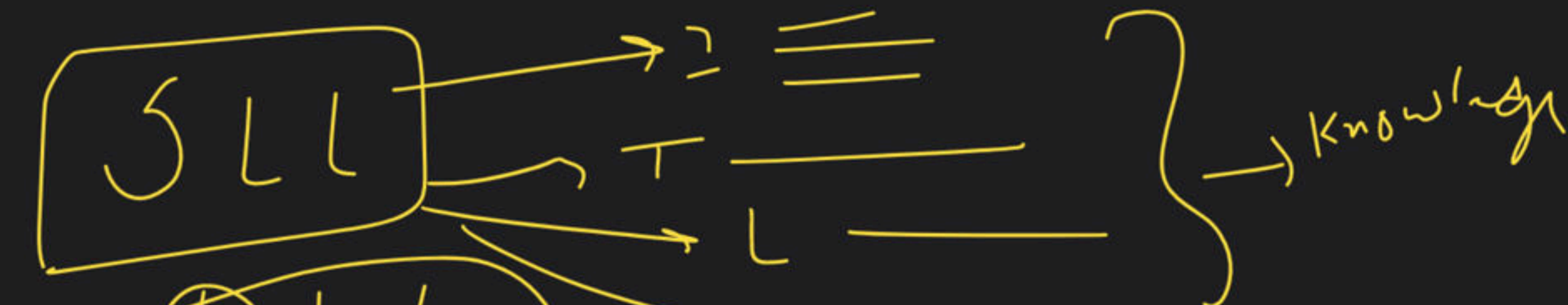


CDLL

pointer updation
↳ order

Edge
Case

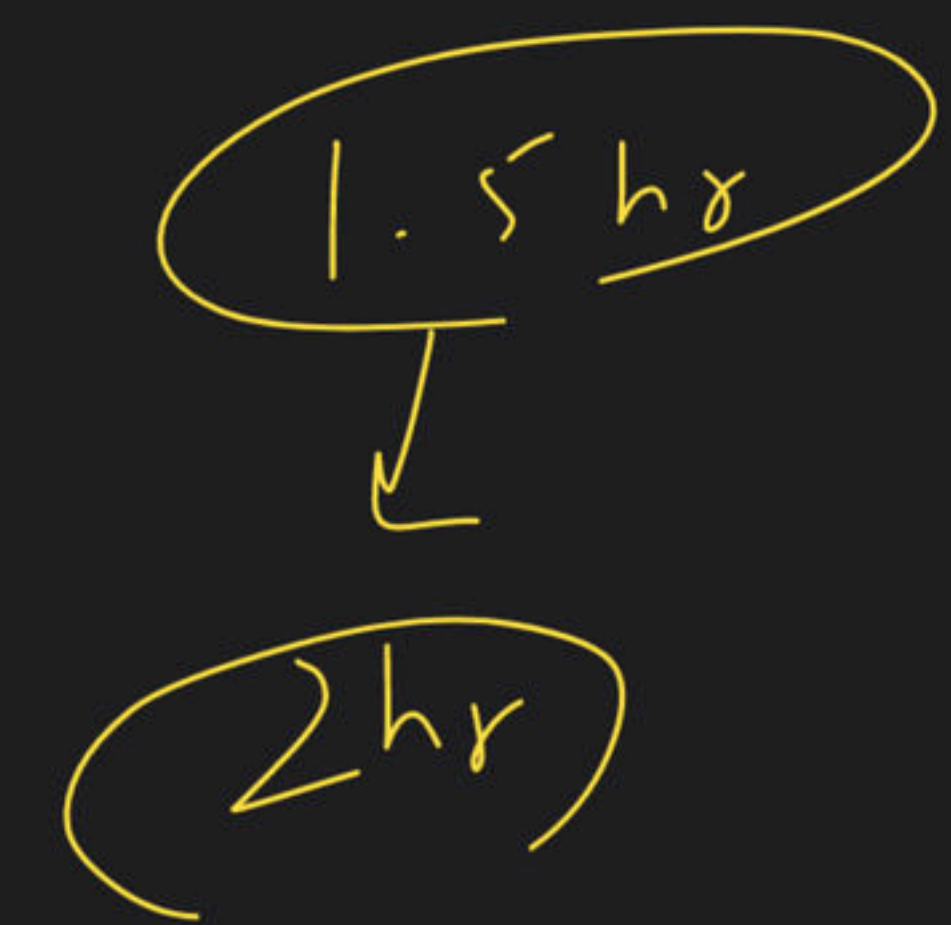




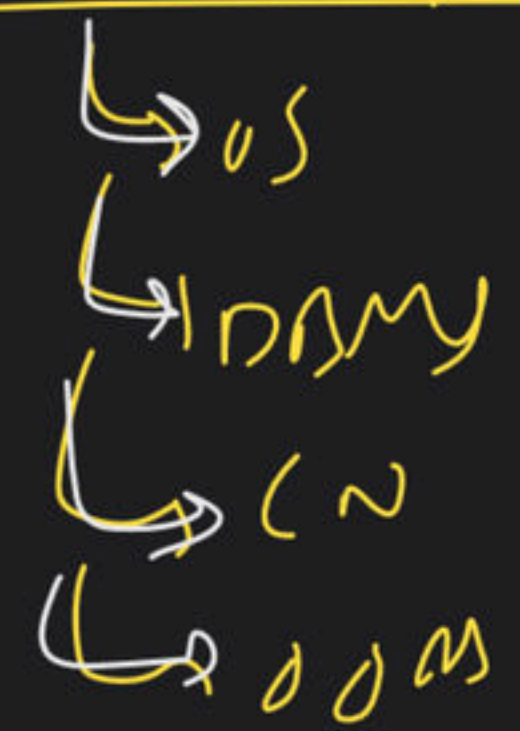
code

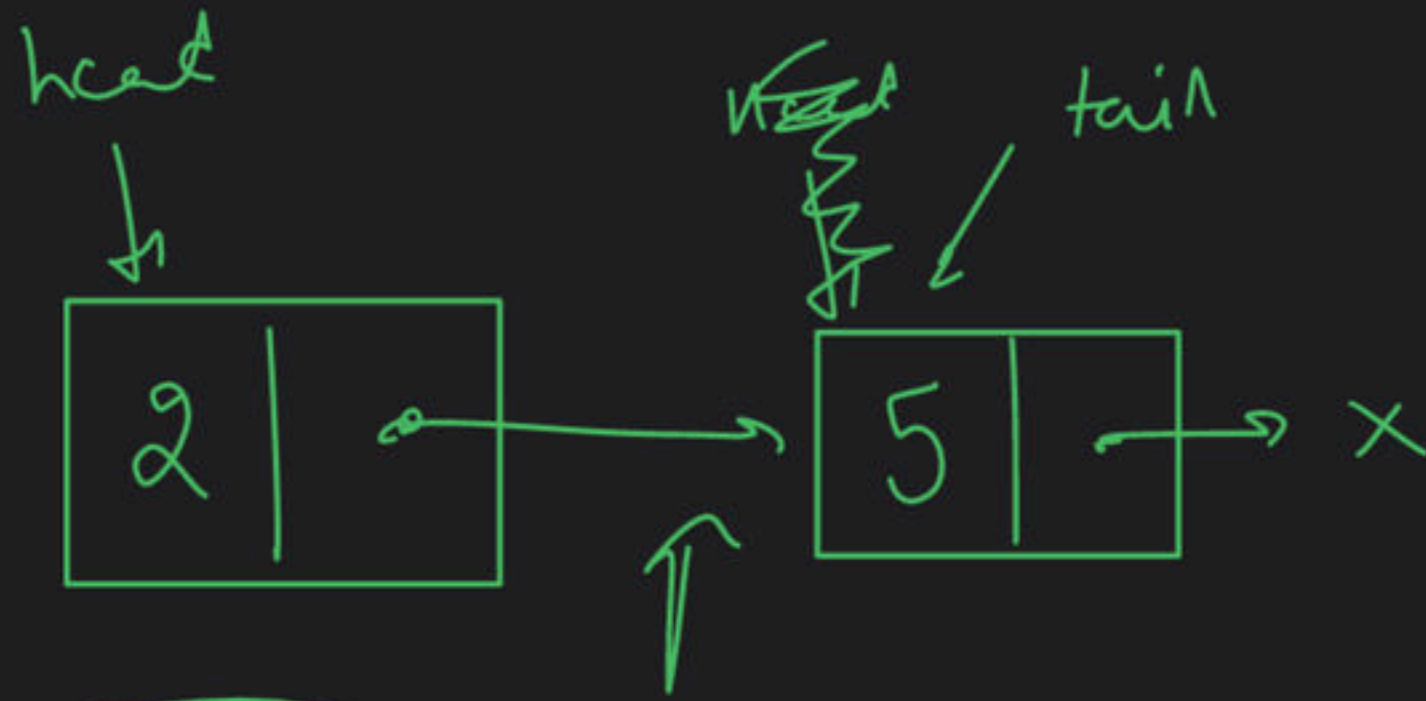
Love
Babbbar

LOVEB




Pano -> Interview Prep BC

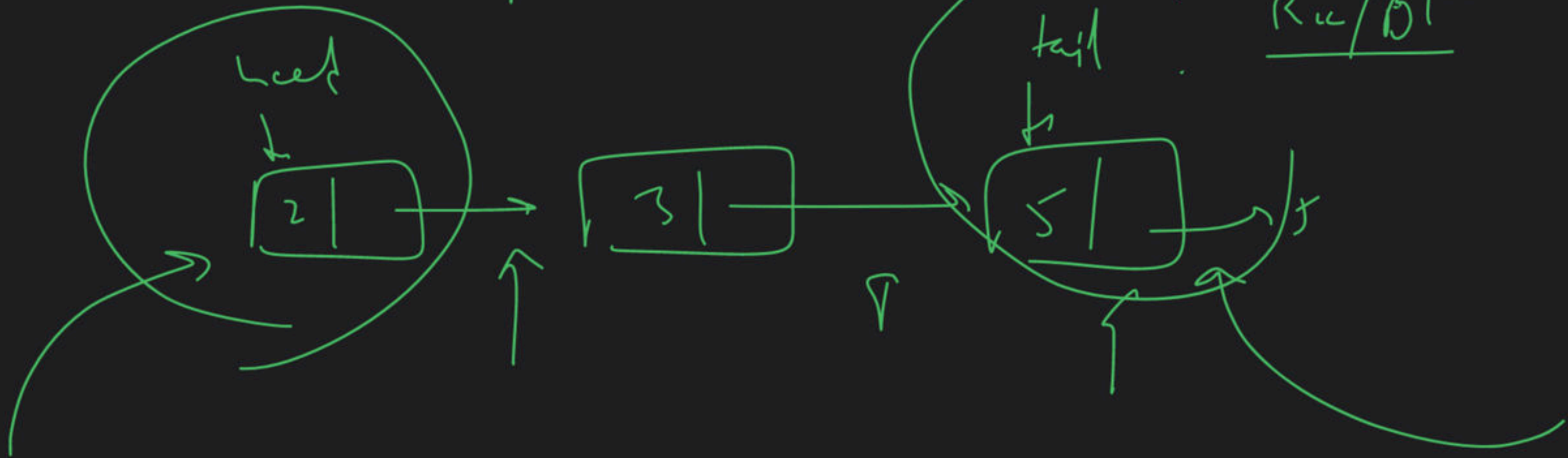





Mock Test (23)

- ↳ CS-14tr
- ↳ LFH → 
- ↳ L.C
- ↳ Rec/BT

Q(n)



pop → 



DSA Sheet 2.0 →

~~Off - Campus~~

- ① MSFT
 - ② LFLH
 - ③ 1 + video
- ↓
- after that