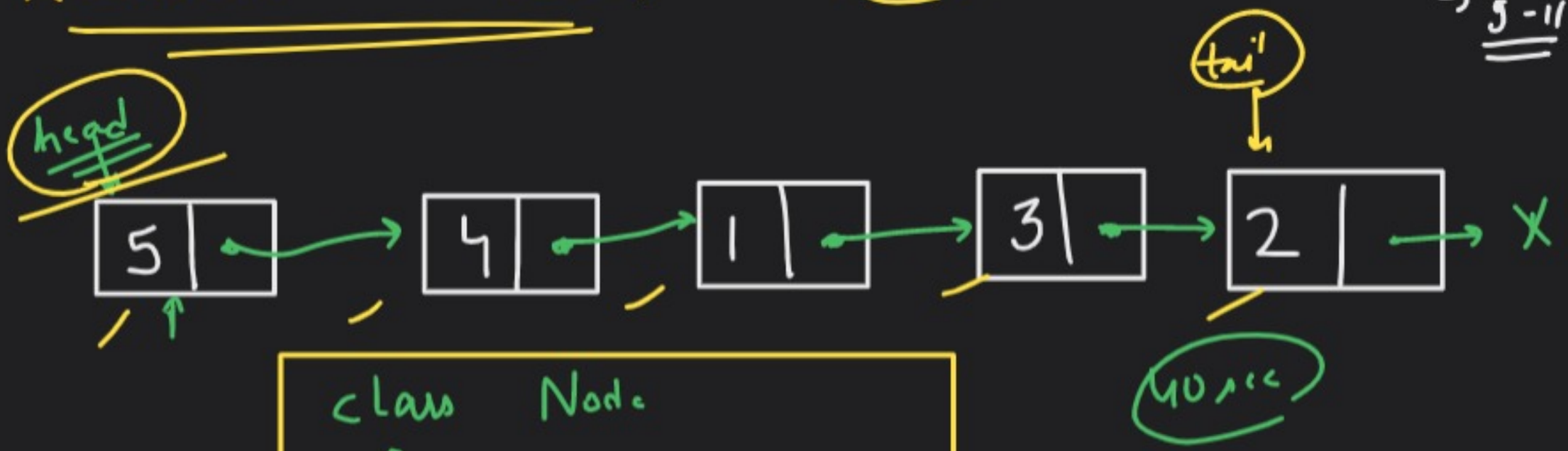
Recursion in LL :-

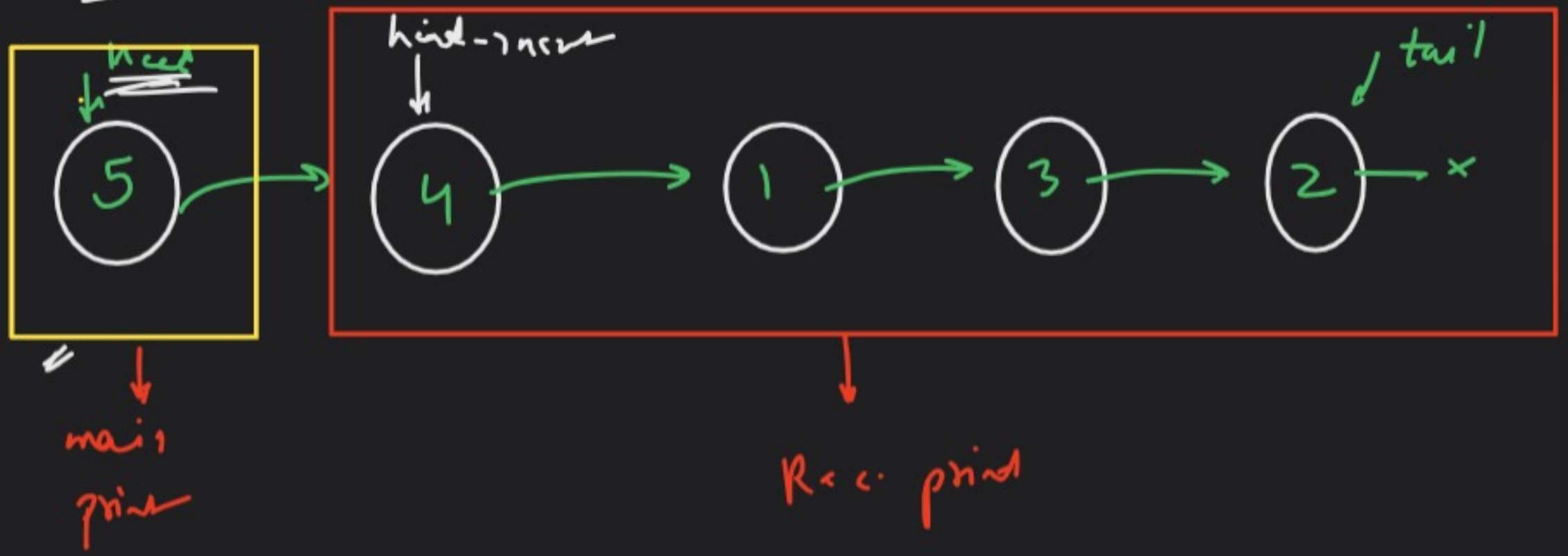
LL

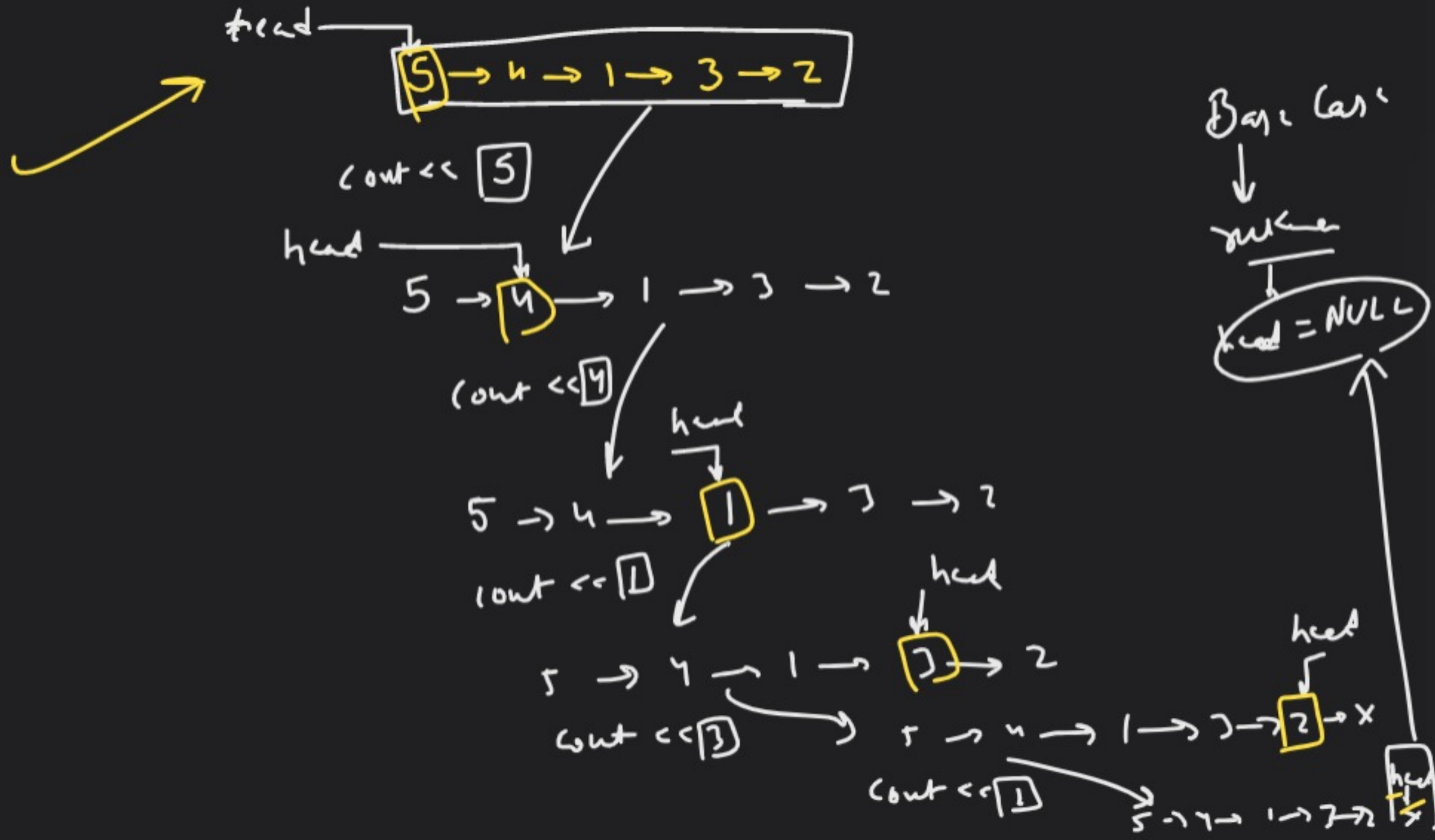
Thu night
↳ 9-11




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

① Print LL / traverse




head
5 → 4 → 1 → 3 → 2
cout << 5
head
5 → 4 → 1 → 3 → 2
cout << 4
head
5 → 4 → 1 → 3 → 2
cout << 1
head
5 → 4 → 1 → 3 → 2
cout << 3
5 → 4 → 1 → 3 → 2 → X
cout << 1
5 → 4 → 1 → 3 → 2
Base Case
head = NULL

```
void print (Node* head)
{
    // Base case
    if (head == NULL)
        return;

    cout << head -> data;

    // R.C
    print (head -> next);
}
```

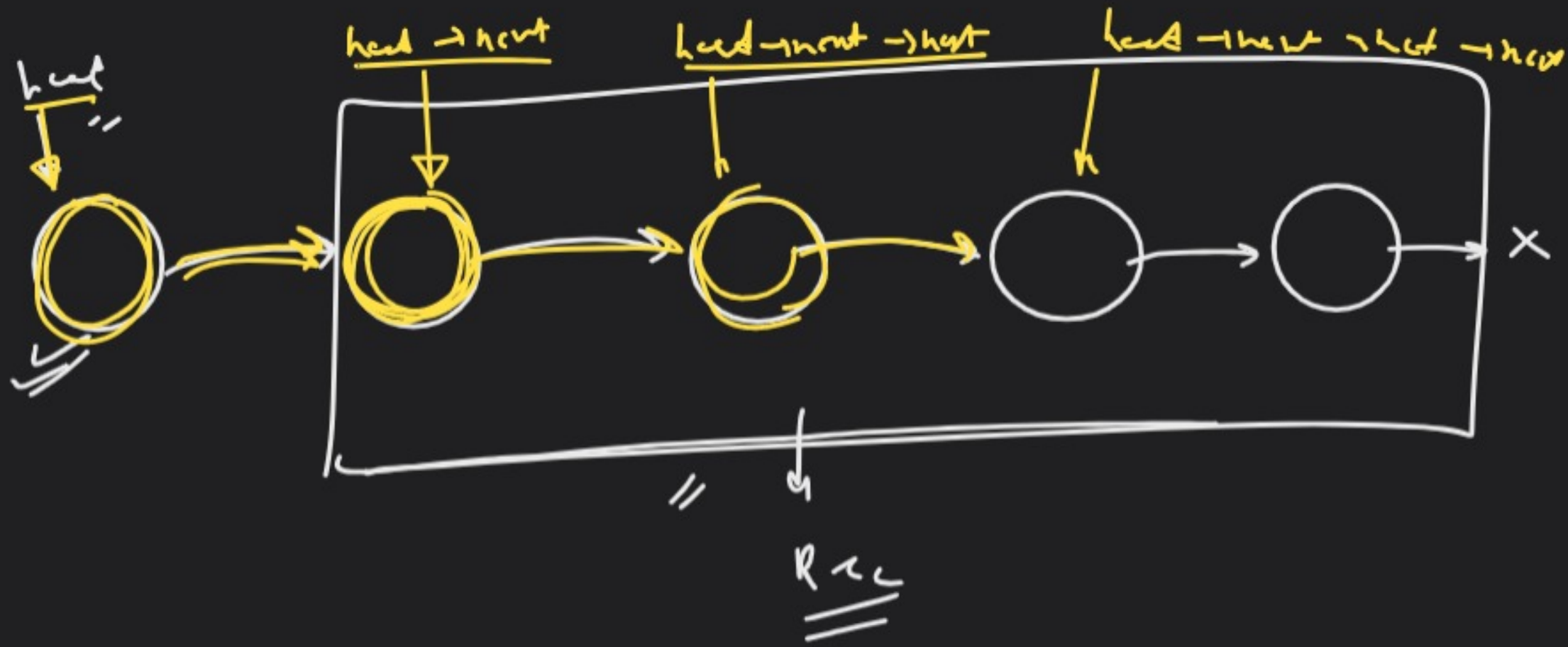
i = size

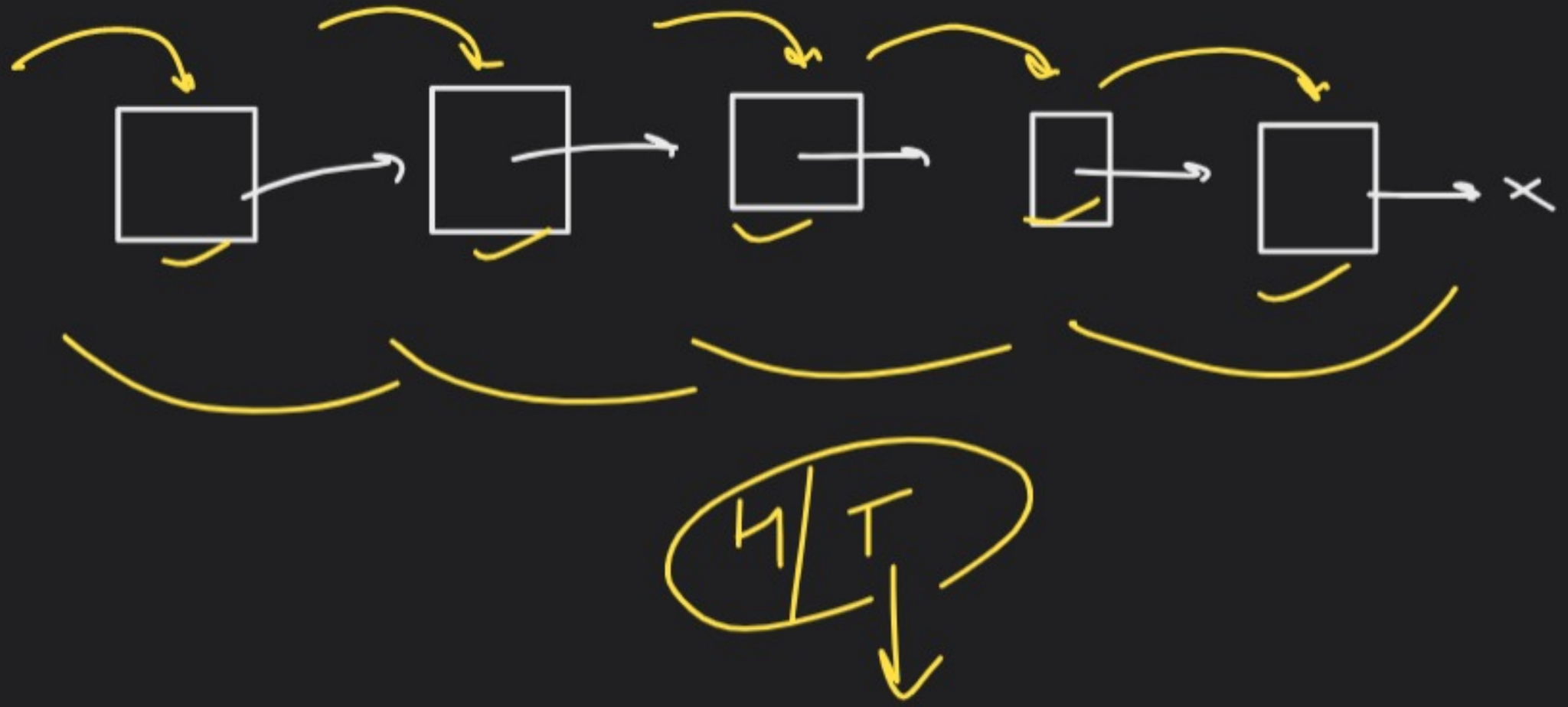
arr[i]
head -> data

arr[i+1]
head -> next

1 case
arr[i]

i+1


head
head -> next
head -> next -> next
X


H/T
X

print K^{th} node from last / end

→ SF → TR
→ CX → FA

1 2 3rd node from end 2nd node from end 1st node from end

1st 2nd

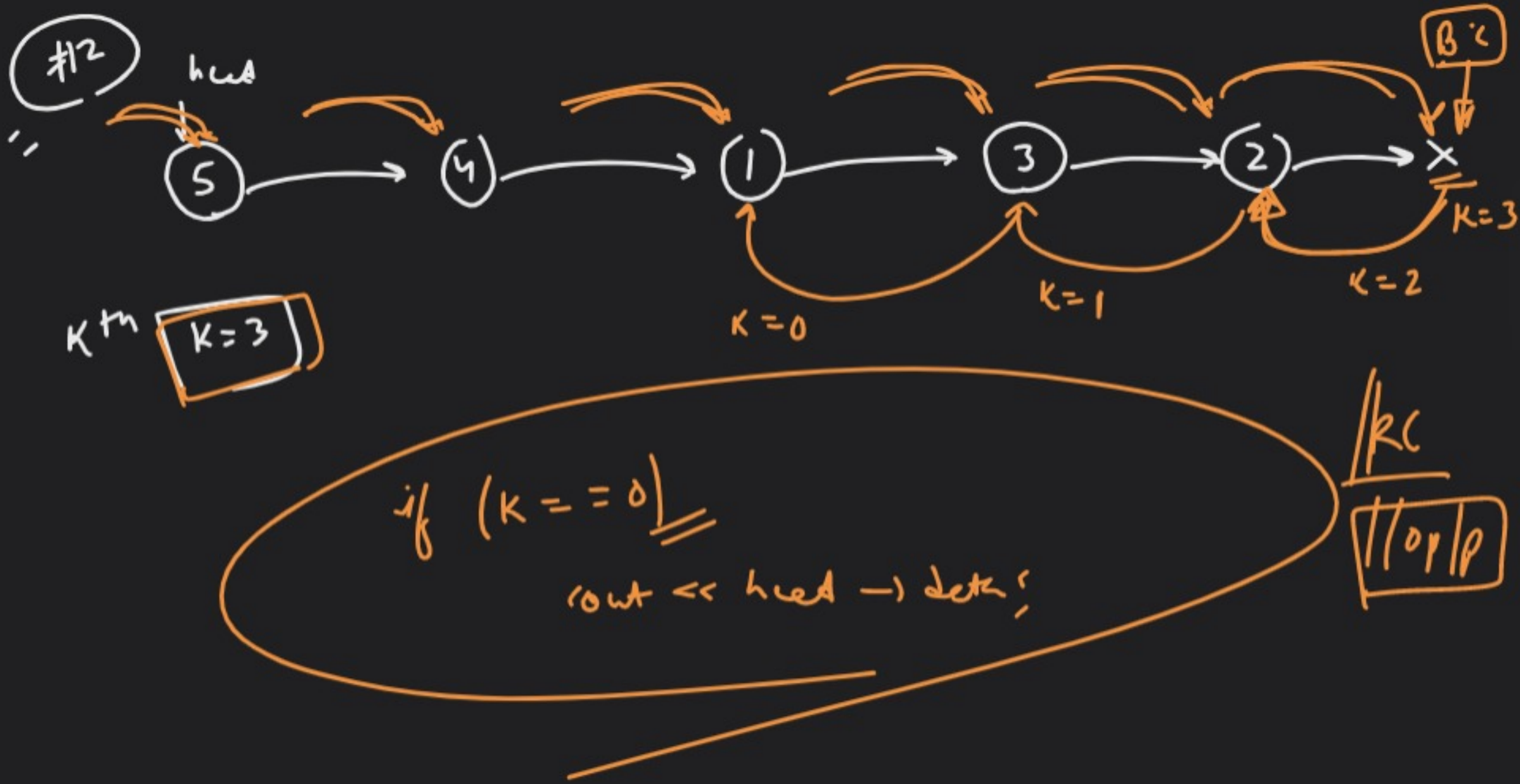
K = 3

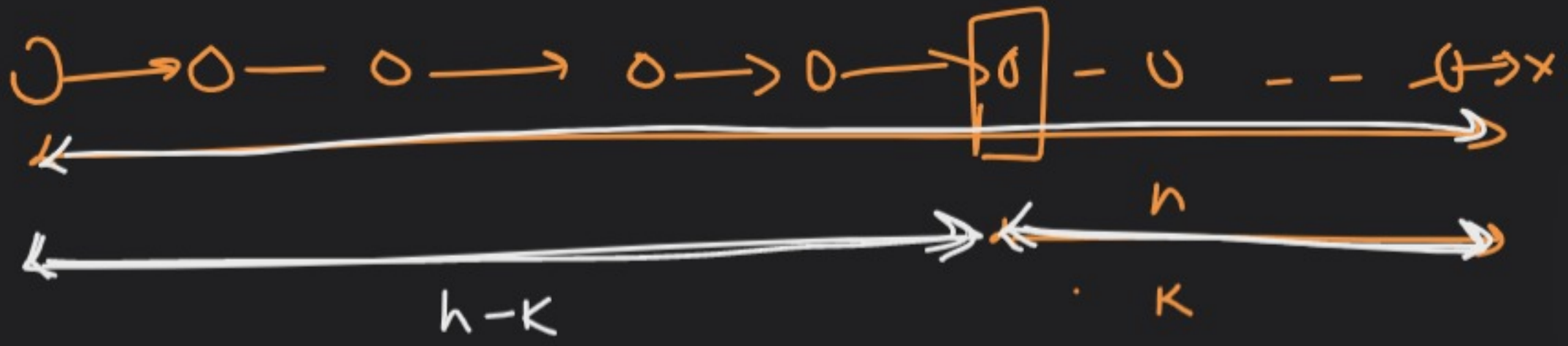
ans → 1

#1 → length → 1st pass

→ → length − K = 5 − 3 = (2)

→ 2nd node return → 2nd pass


#12
head
5
4
1
3
2
X
B'c
K=3
K=2
K=1
K=0
Ktm
K=3
if (K==0)
cout << head -> data;
Kc
Hop



#1 print $(n-K)^{th}$ from start

↓

length of LL

empty

```
if (head == = NULL)
```

return NULL

print -1

print

Head

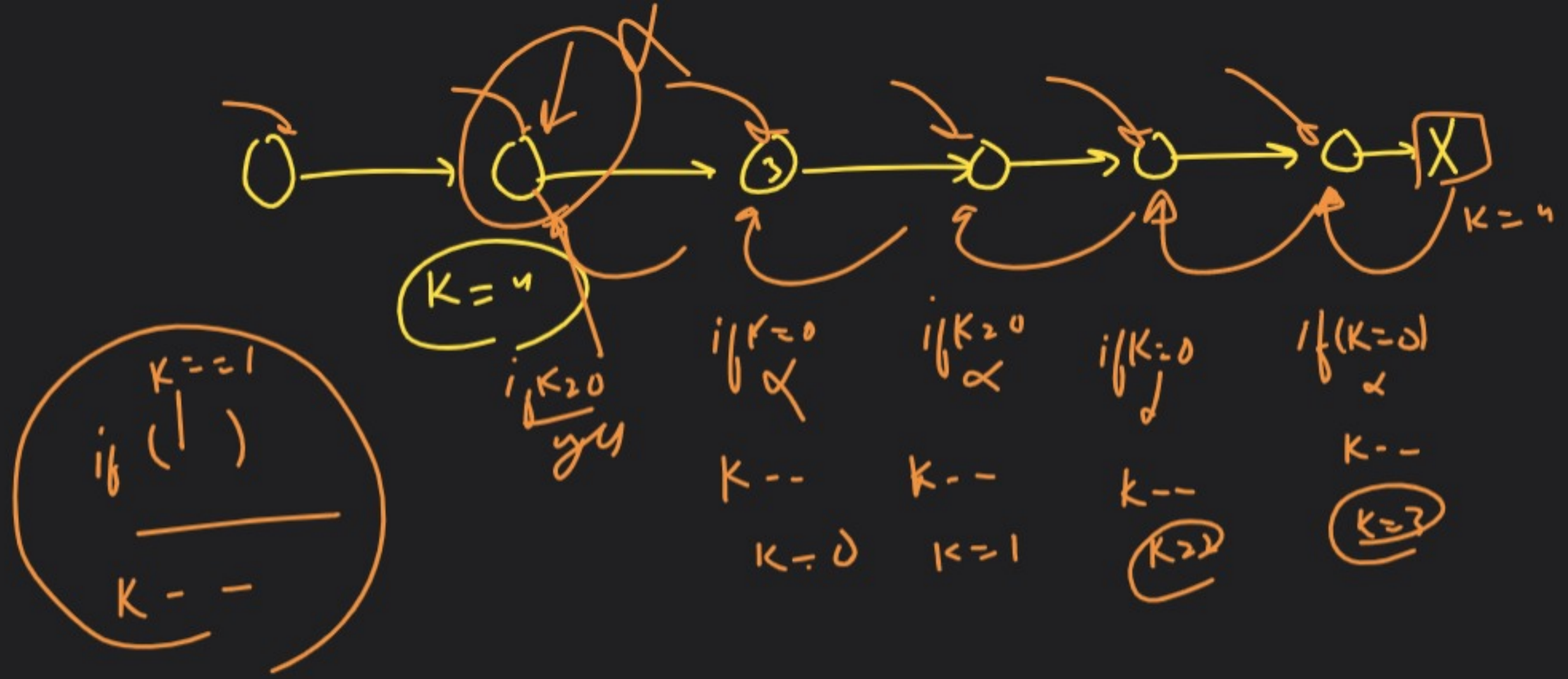
R.C

```
if (K > n)
```

return -1


k = 101
if (k > n)


k - -
if (k == 0)
or
if ( )
k--


K==1
if (|)
K--
K=4
if K>0
3
X
K=4
if K=0
K--
K=0
if K>0
K--
K=1
if K=0
K--
K>2
if (K=0)
K--
K=3

Ist case

IInd case

H:

→ R.C → solve(head → next)

→ process → cout << head → data

→ process

cout << head → data

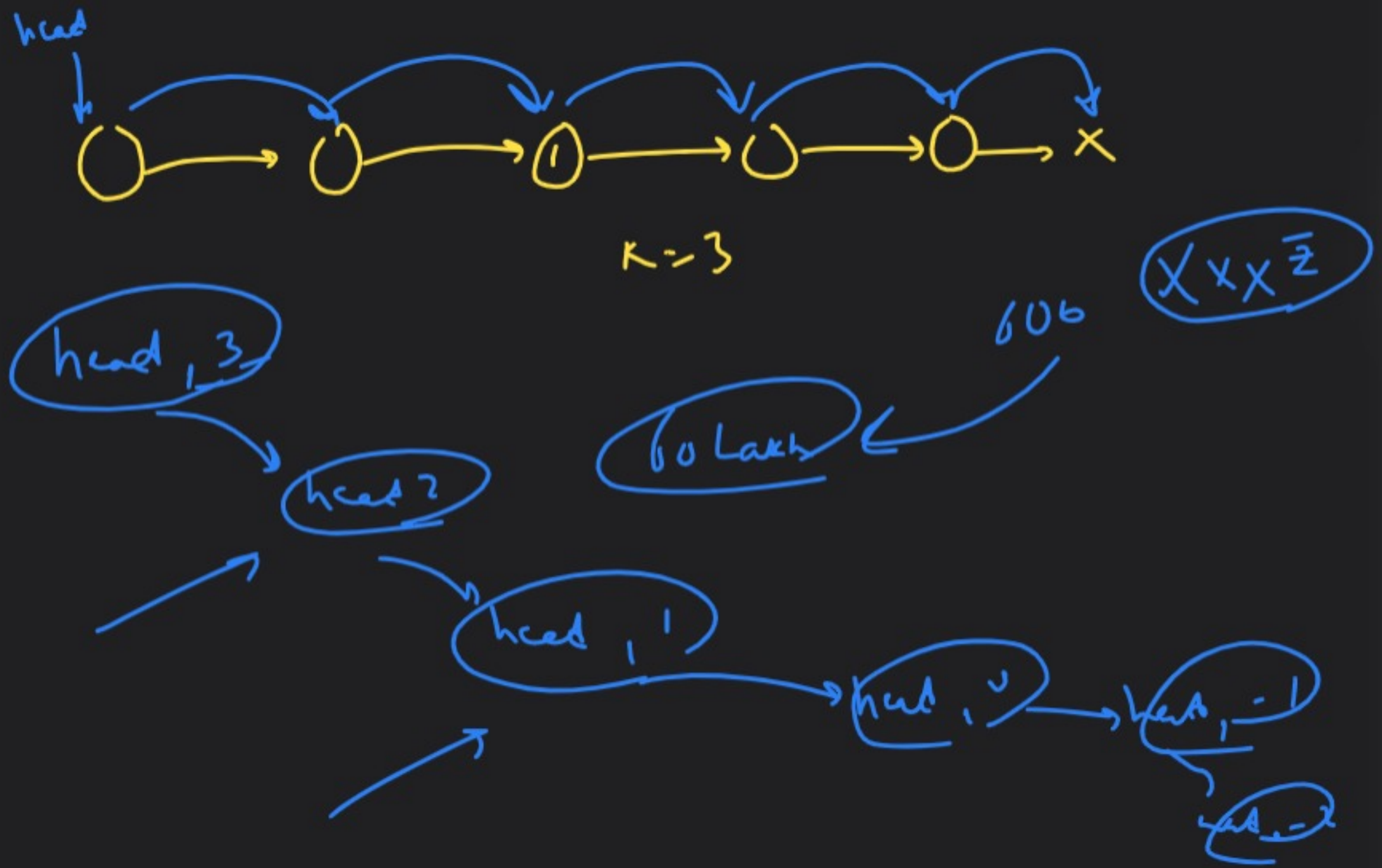
→ // R.C

solve (head → next)

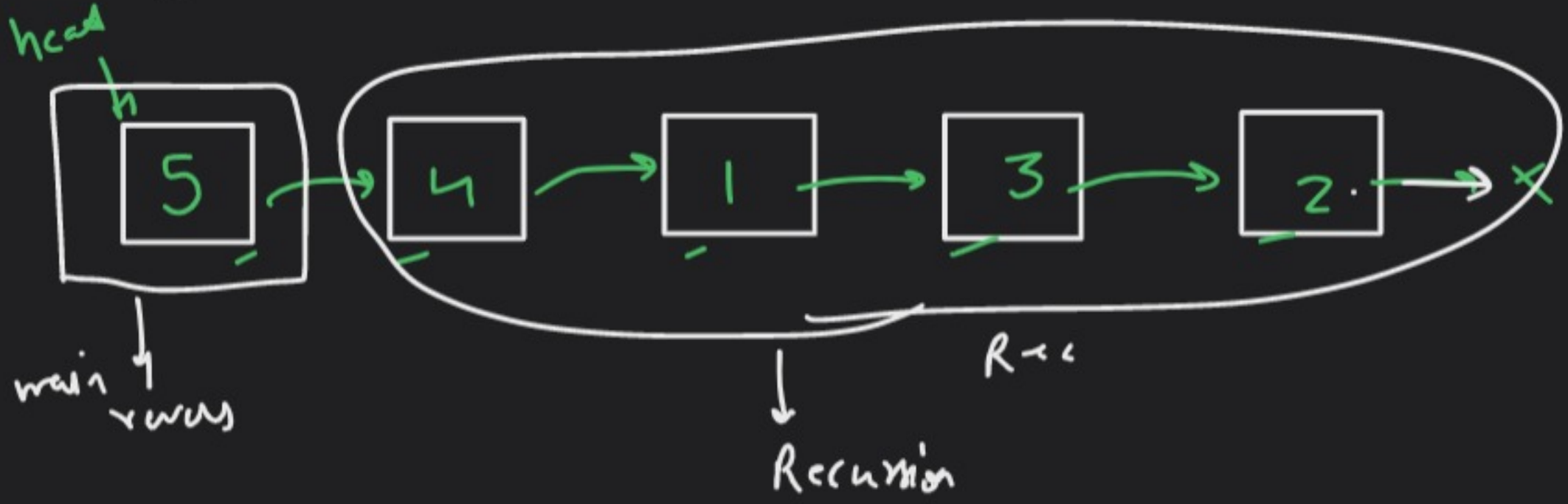
Problem:- → Problem → 5 Dry Run

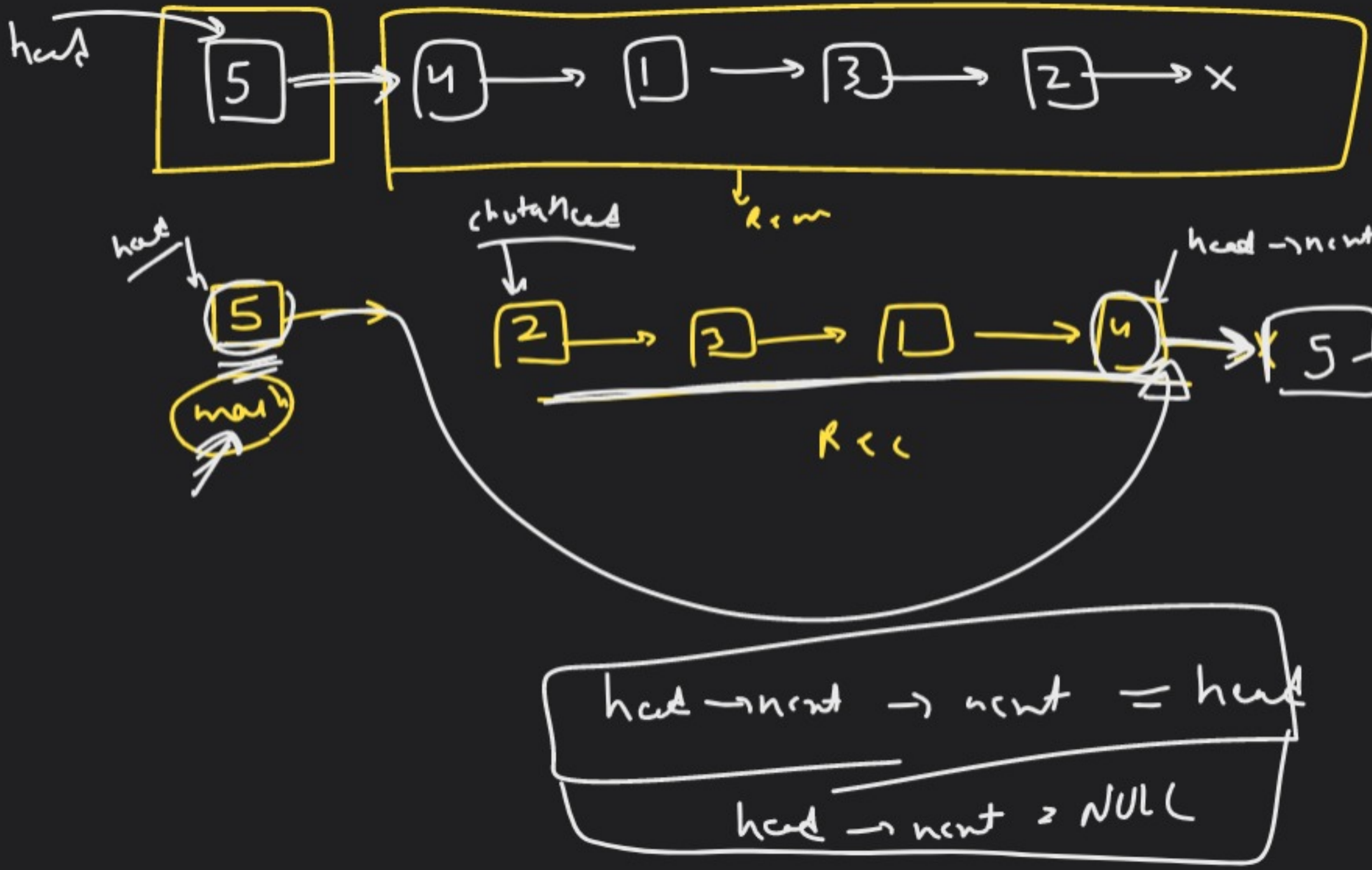
60-70%

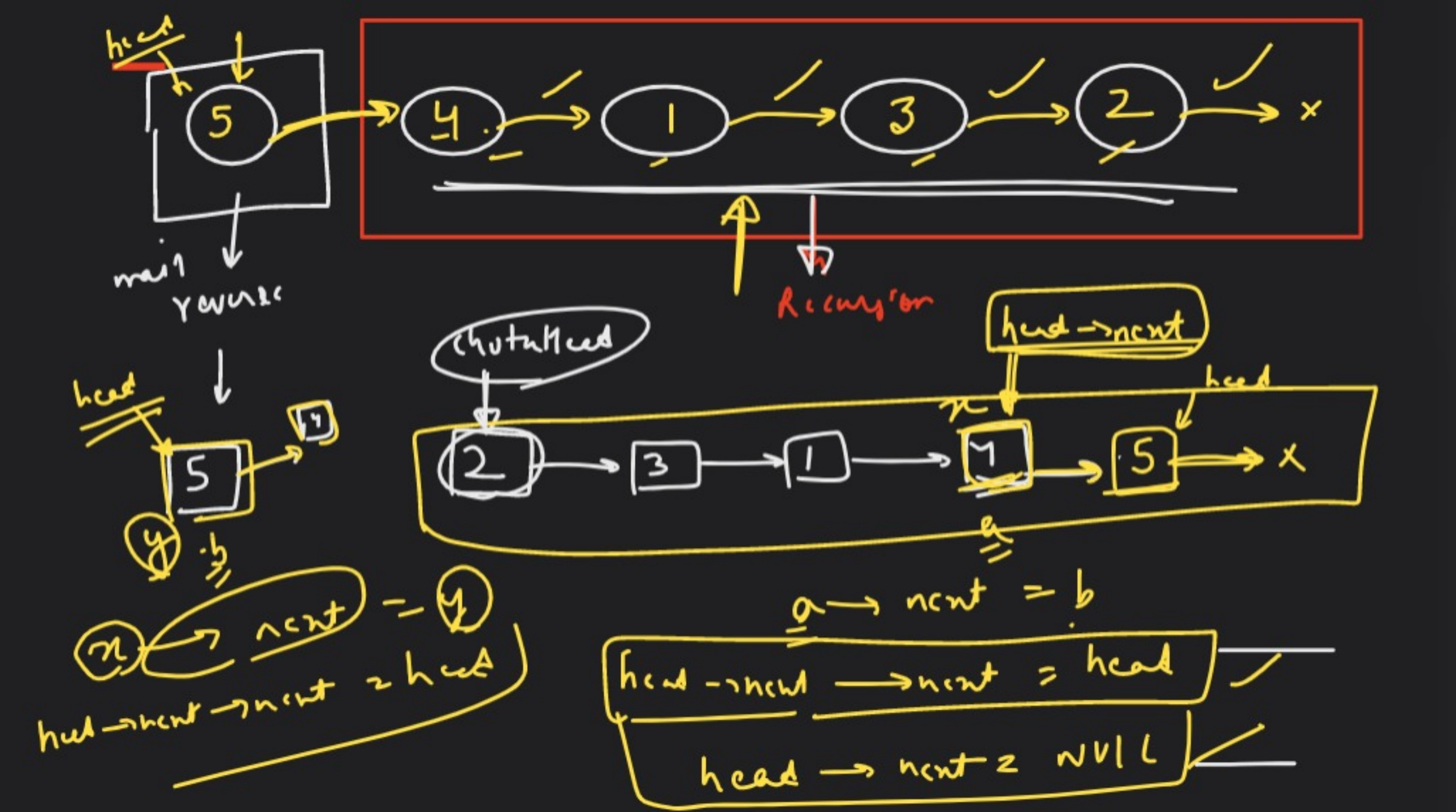
Dry Run


head
X
K=3
XXX
606
10 Lakh

Reverse a LL




head
5
4
1
3
2
X
chotaHead
Rec
head
5
main
2
2
1
4
head->next
head
5
X
Rec
head->next->next == head
head->next = NULL

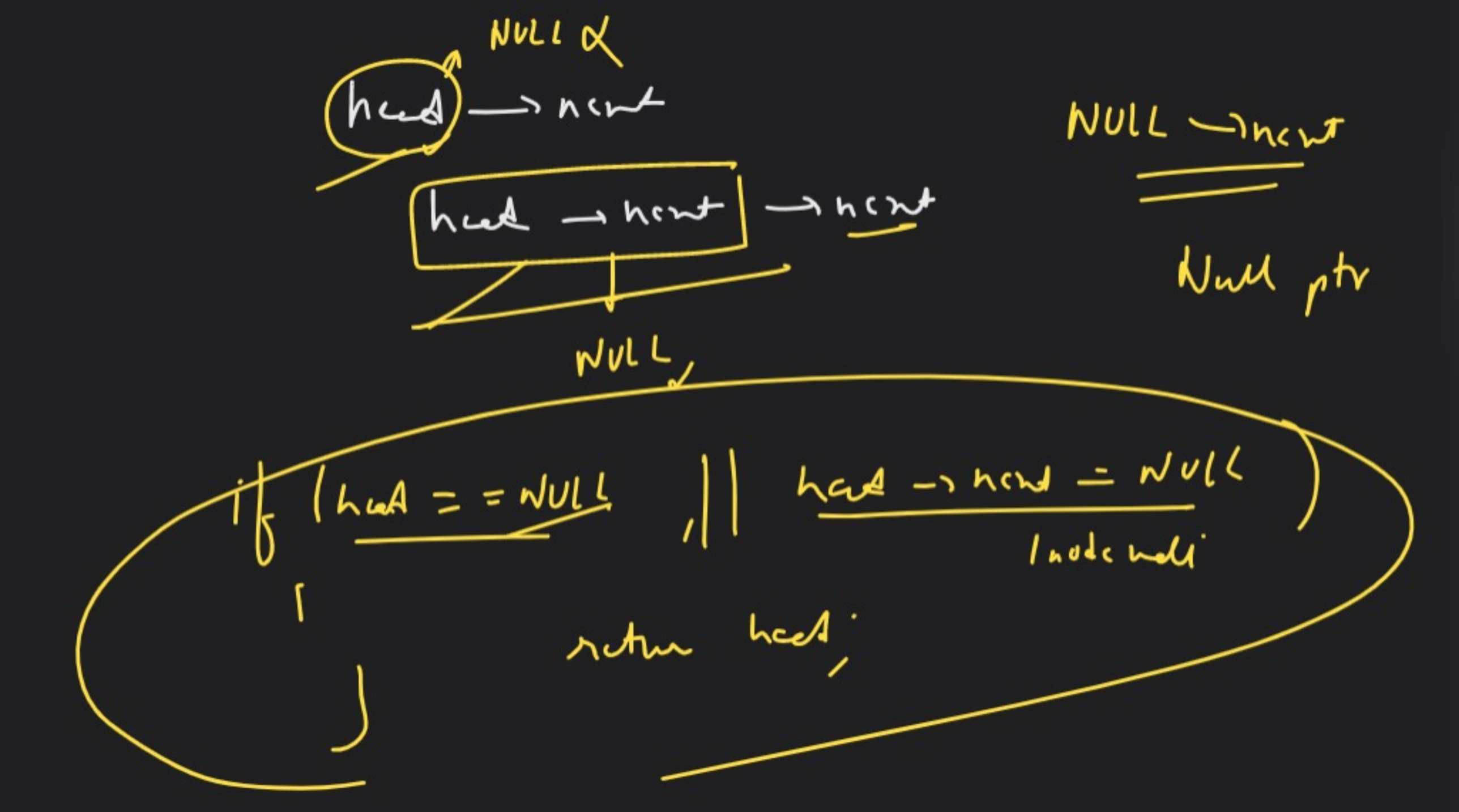

head
5
4
1
3
2
x
main
reverse
Recursion
chotaHead
head->next
head
5
4
y
2
3
1
7
5
x
a->next = b
head->next->next = head
head->next = NULL

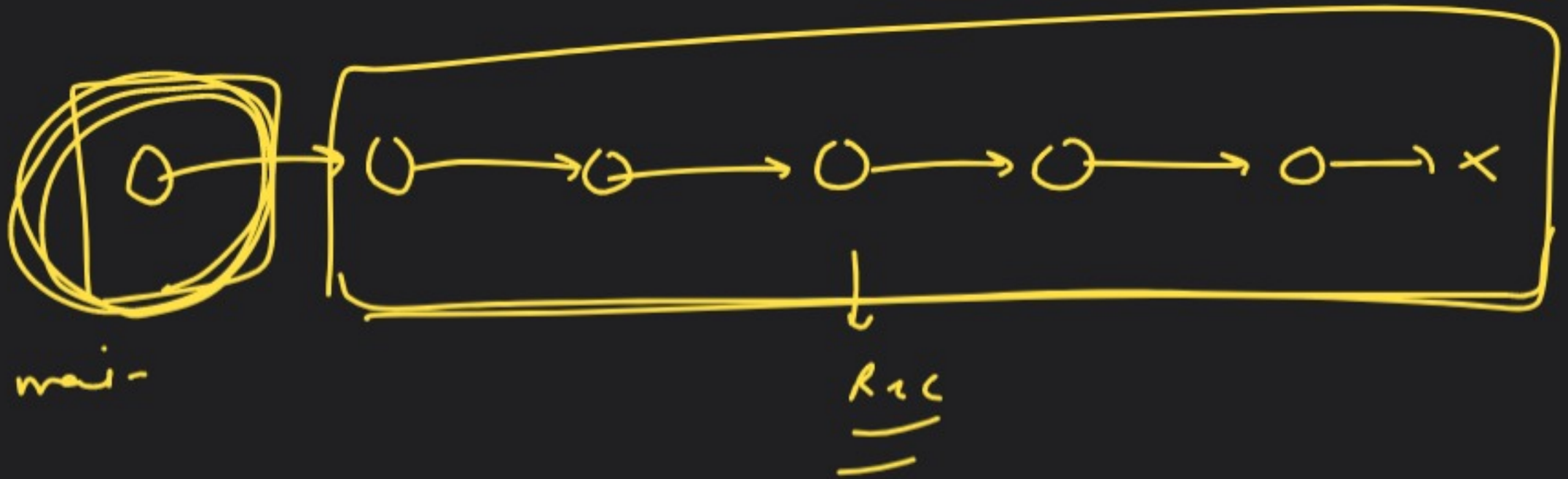

a
b


x -> next = b

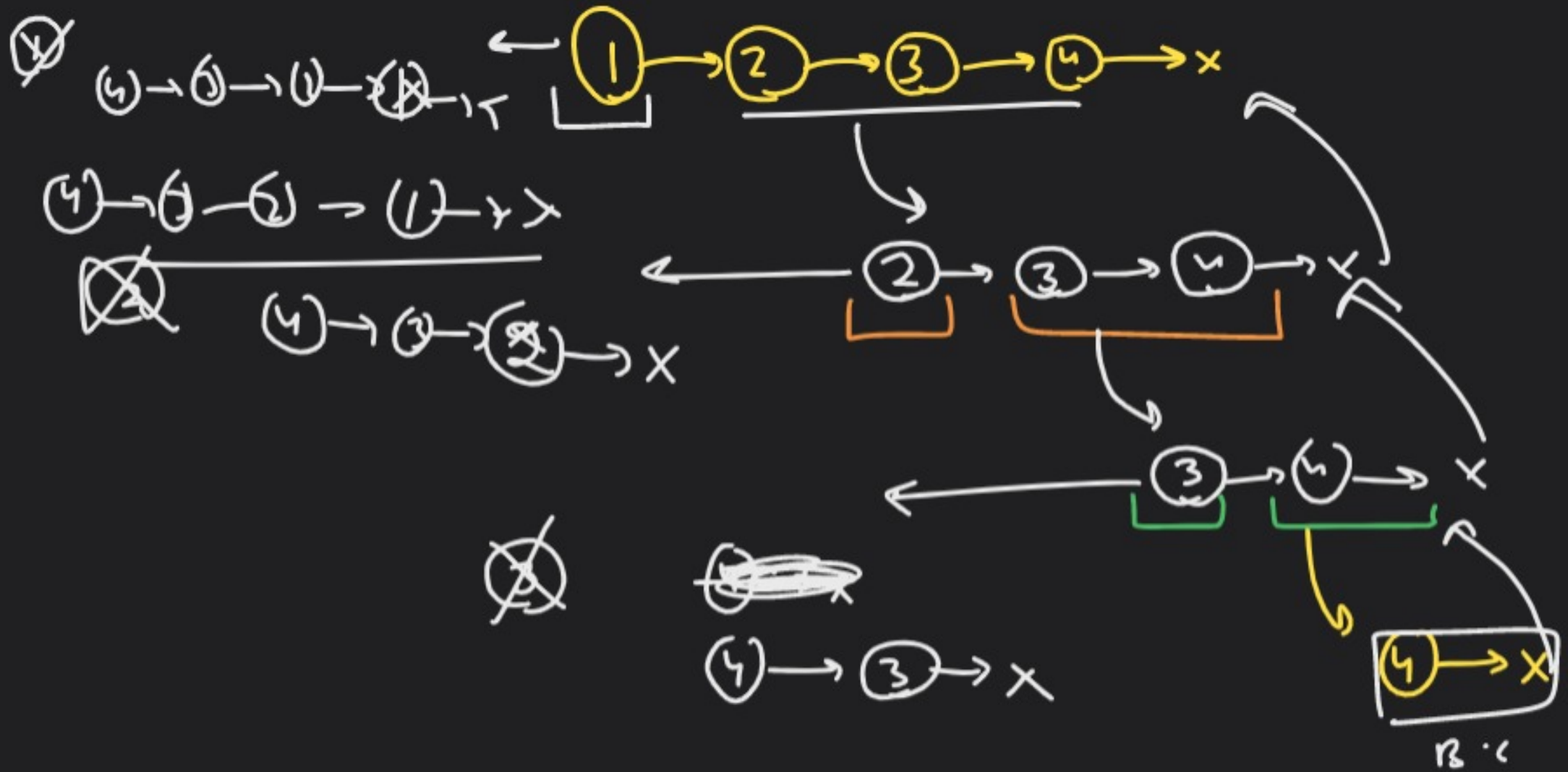

x -> next = y

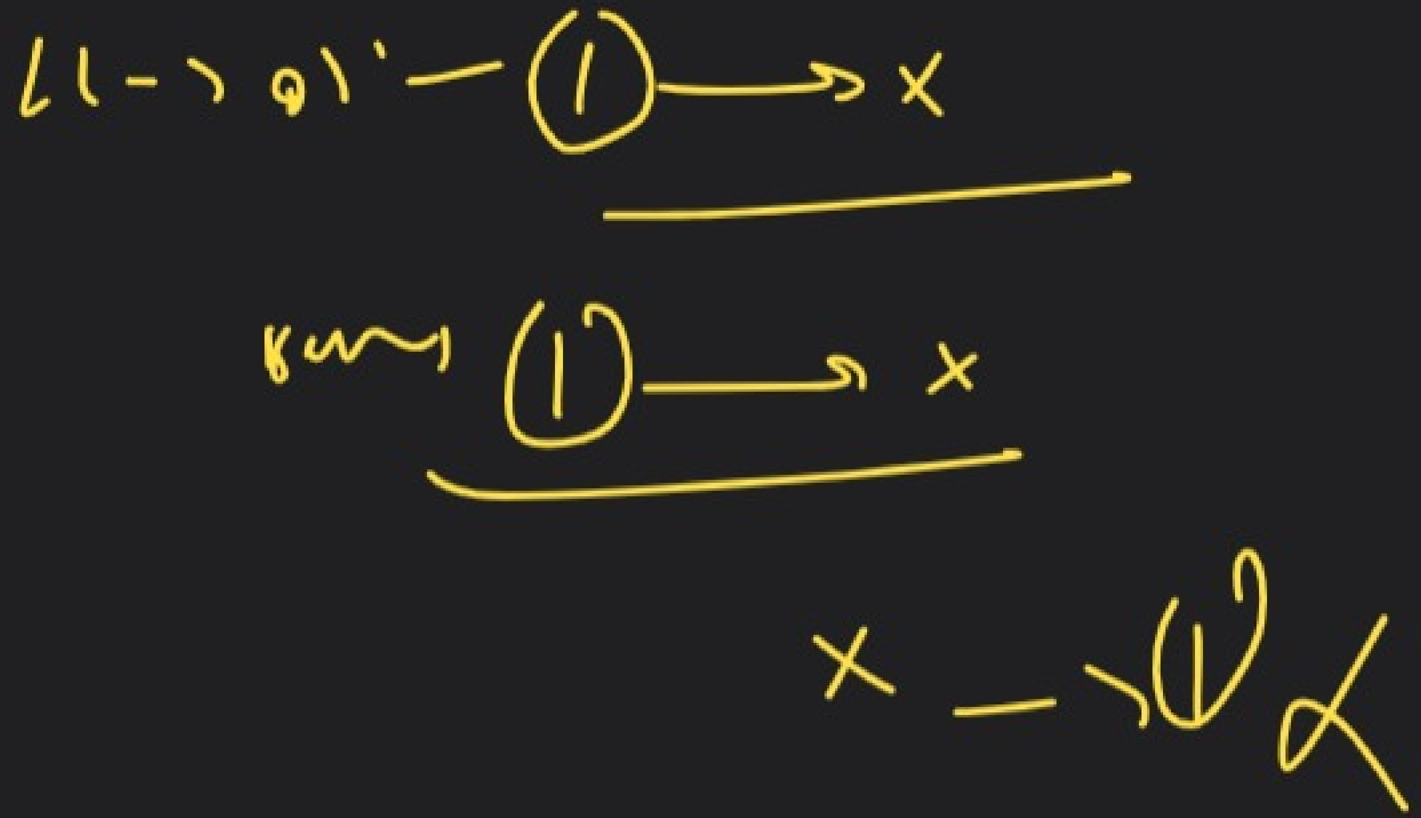

x
y

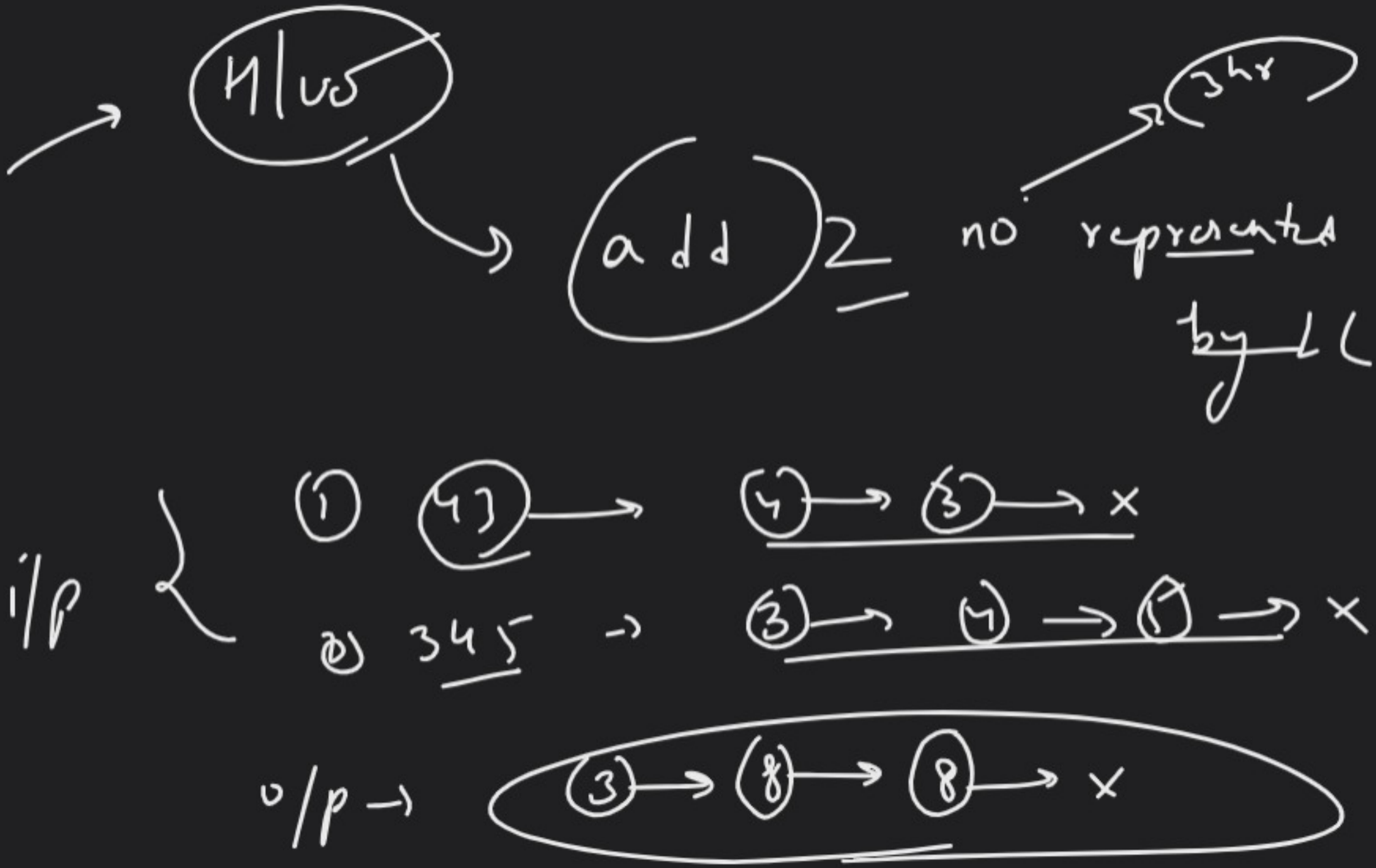

NULL
head
head → next
head → next → next
NULL → next
Null ptr
NULL
if (head == NULL || head → next == NULL)
return head;

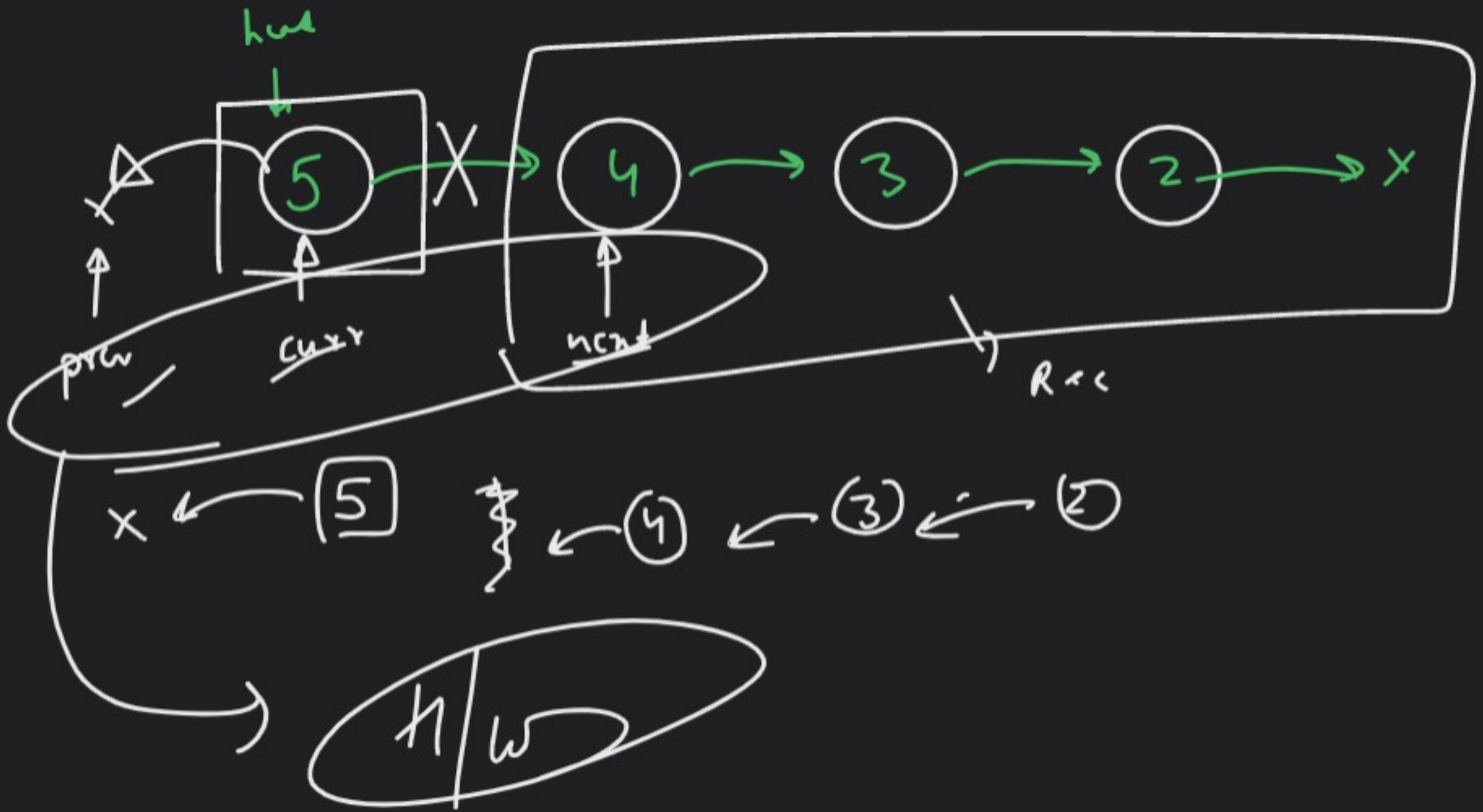

mai-
Rec

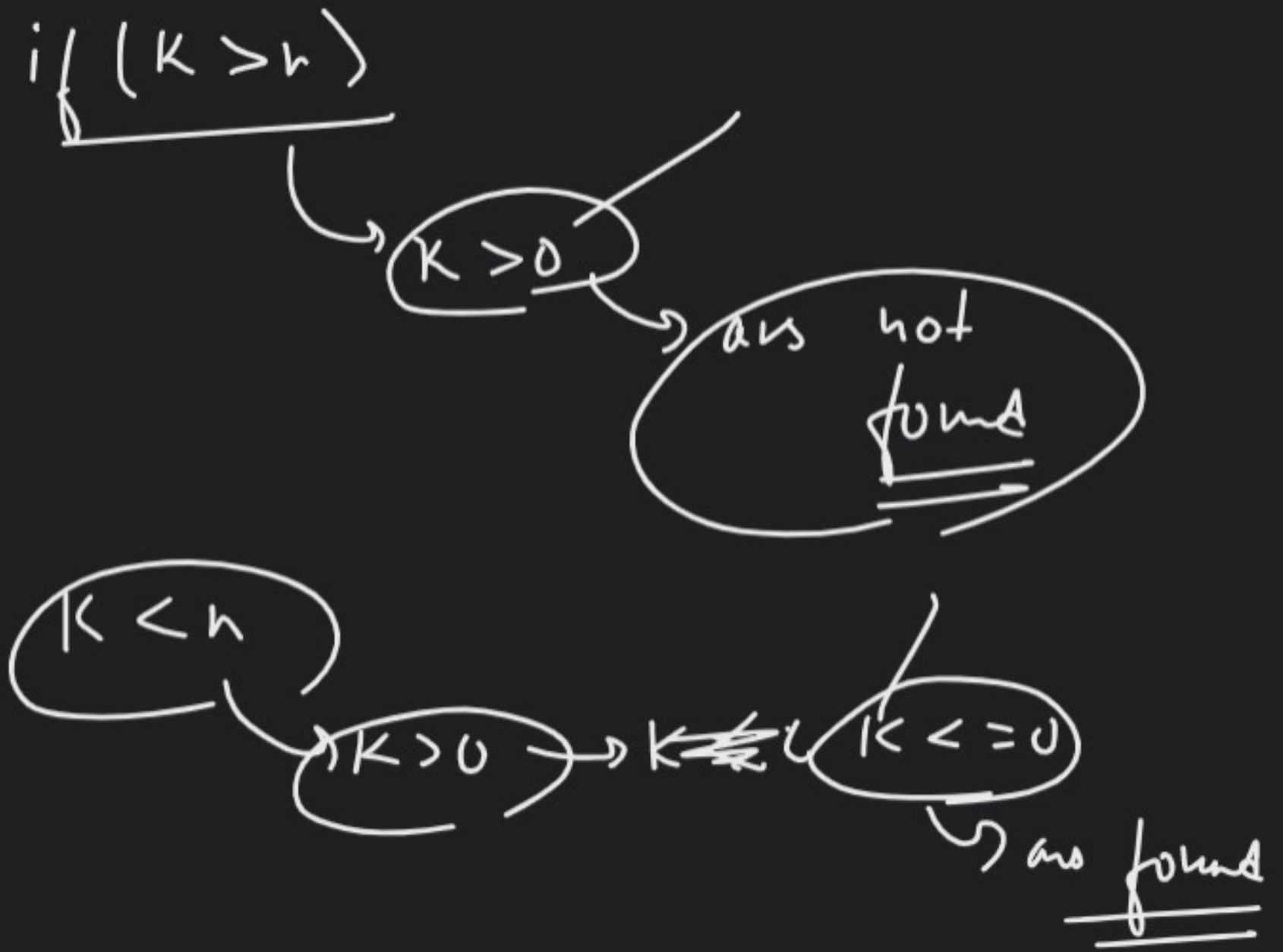
2 min break!

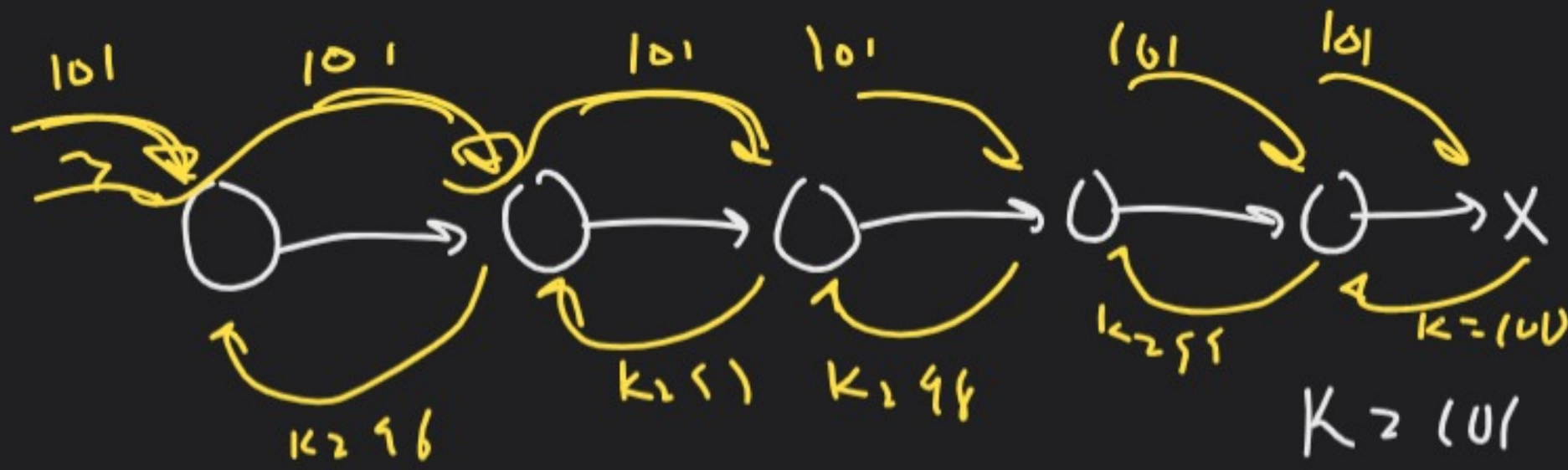

1 2 3 4 X
2 3 4 X
3 4 X
4 X
4 3 X




Main
add
2
no
represents
3hr
by LL
i/p
(1) (4) → (4) → (3) → x
(2) 345 → (3) → (4) → (5) → x
o/p →
(3) → (8) → (8) → x

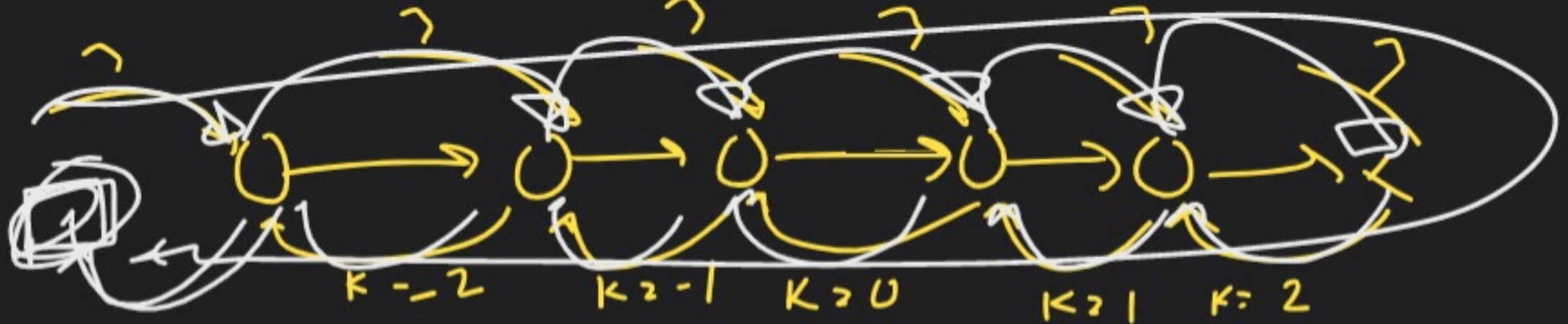

head
5
4
3
2
X
X
prev
curr
next
Rec
X
5
4
3
2
h
w


if (k > n)
k > 0
ans not found
k < n
k > 0
k <= 0
ans found



K = 98

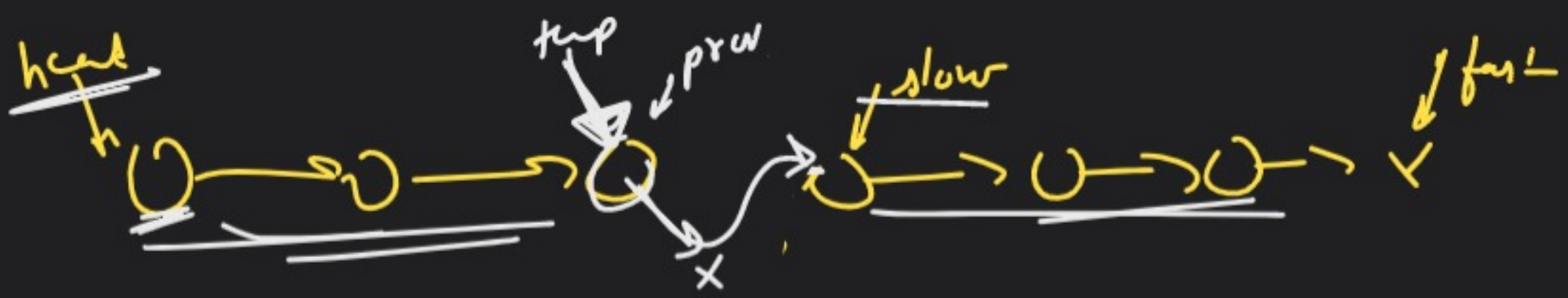
K > 0 → pattern not found



as print / show

as found


```
main()
{
  p
```



Tortoise algo

(4)

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

(1) prev → next = NULL

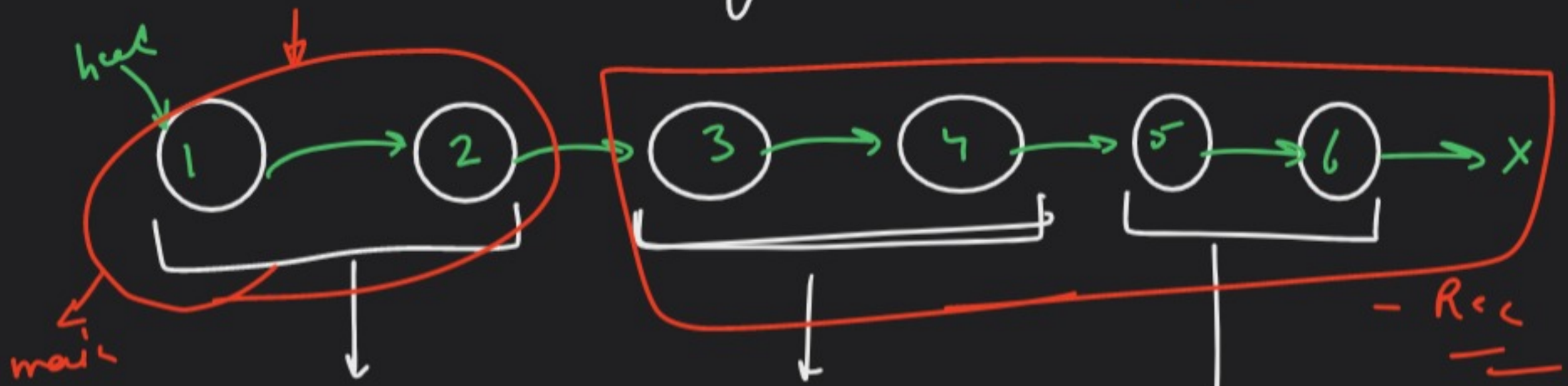
(2) head = reverse(head)

(3) slow = reverse(slow)

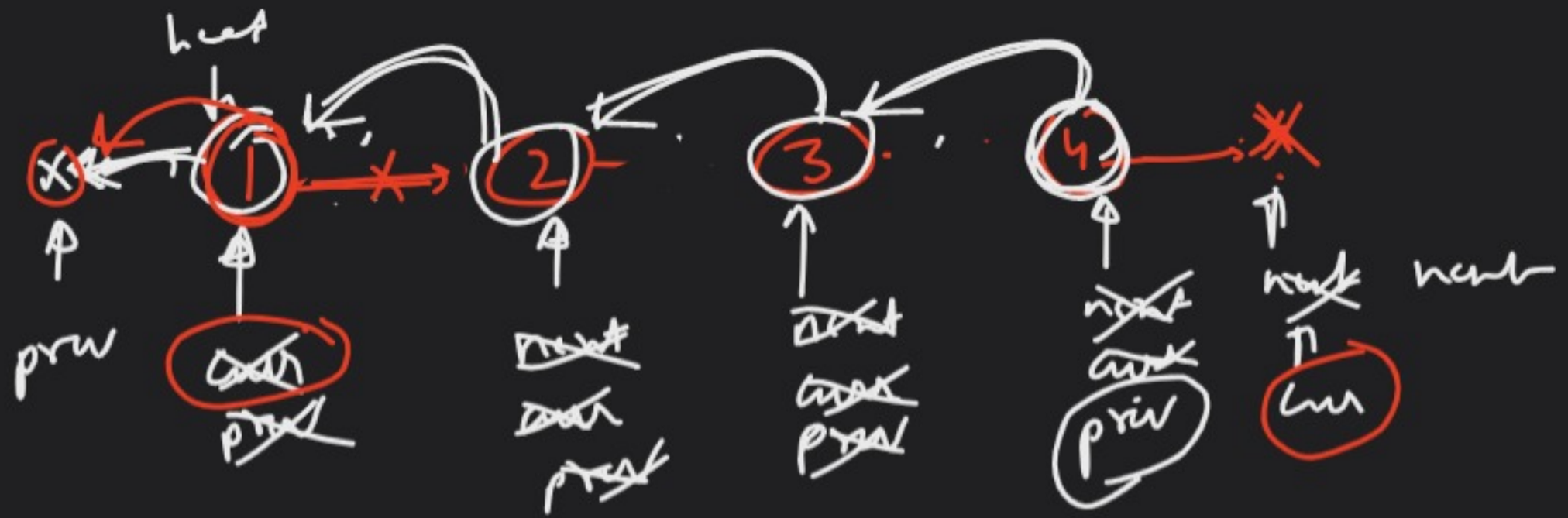
(5) temp->next = slow

→ Reverse LL in group of "K"

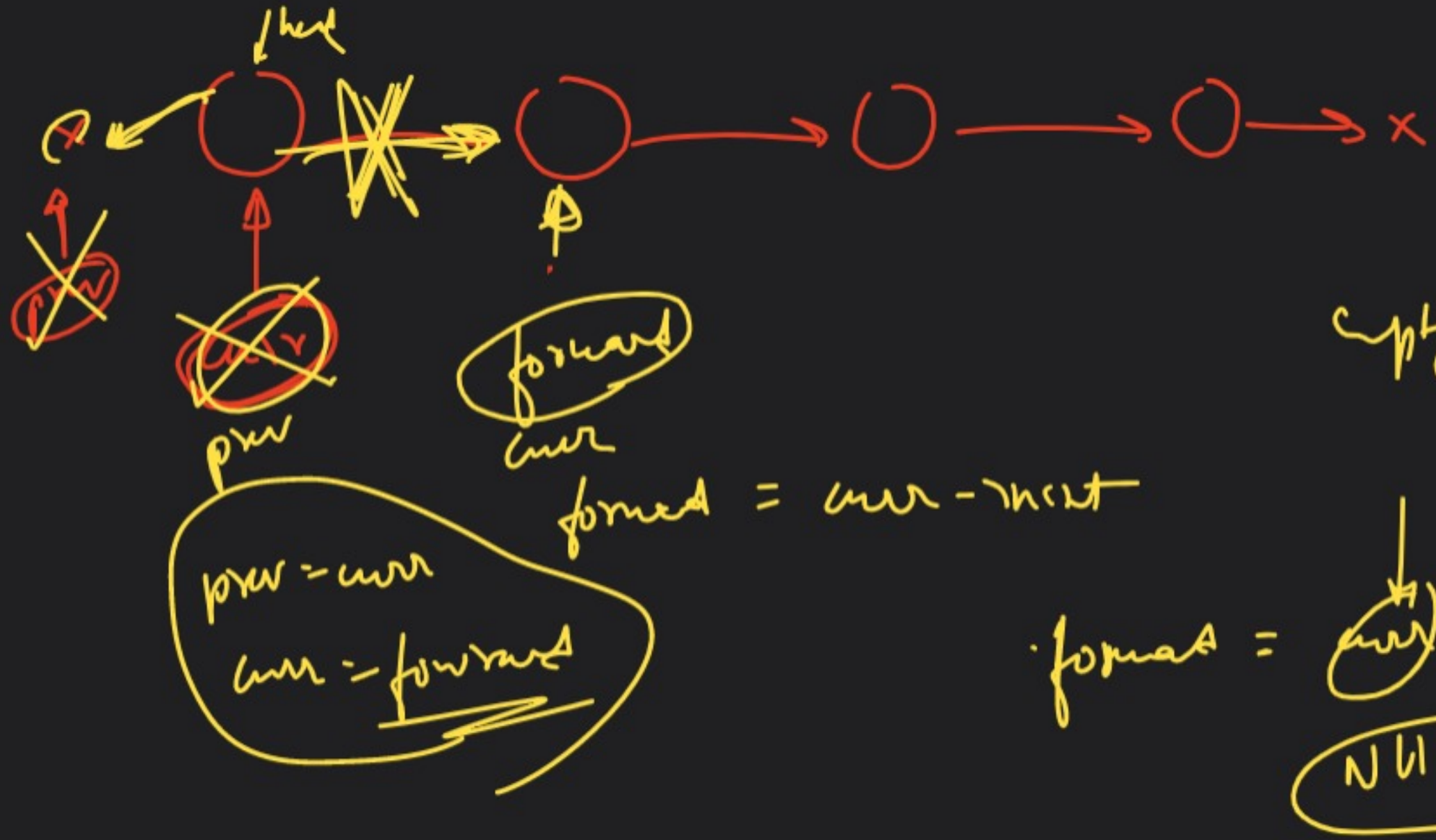
K = 2

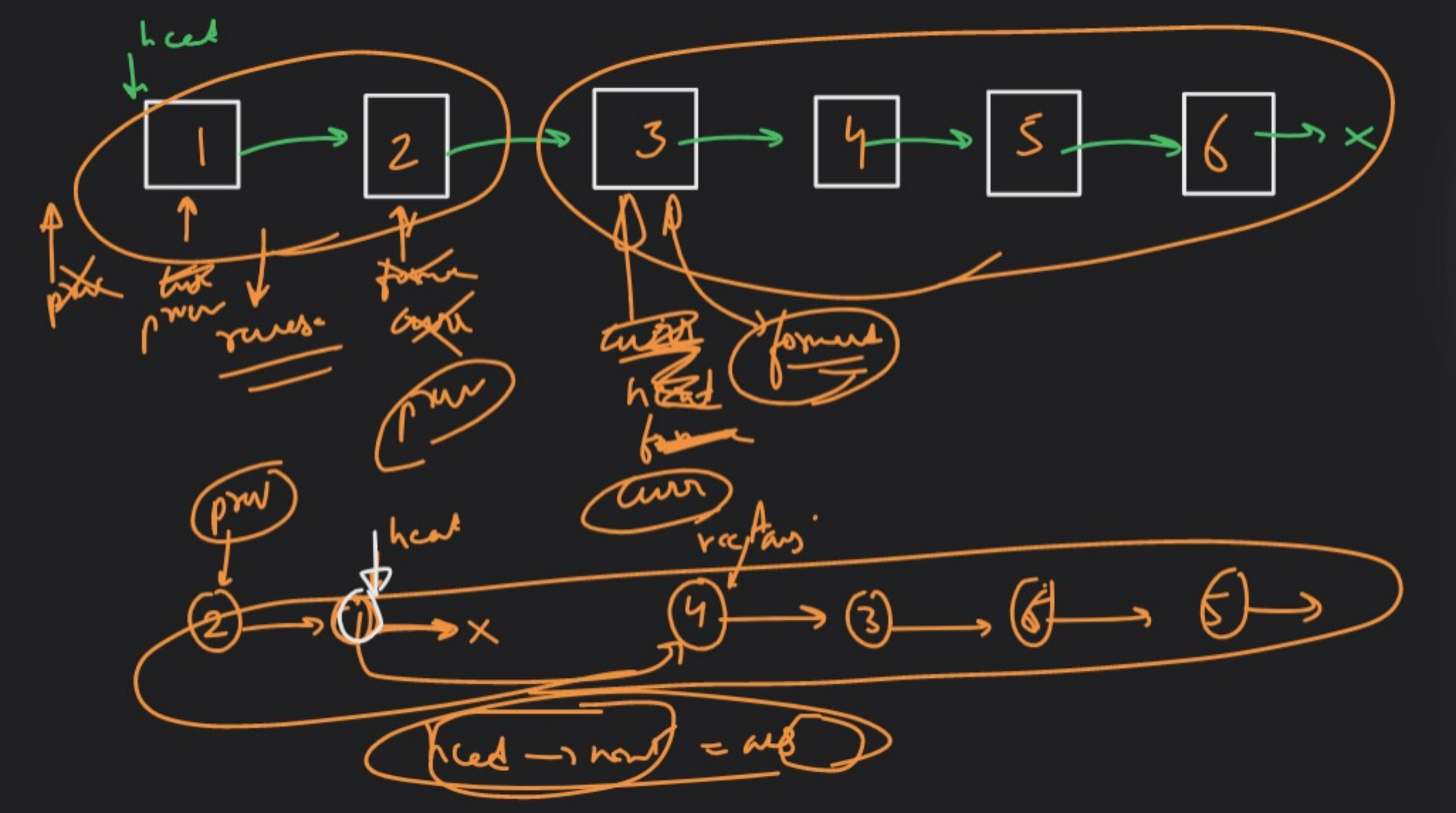


o/p ->



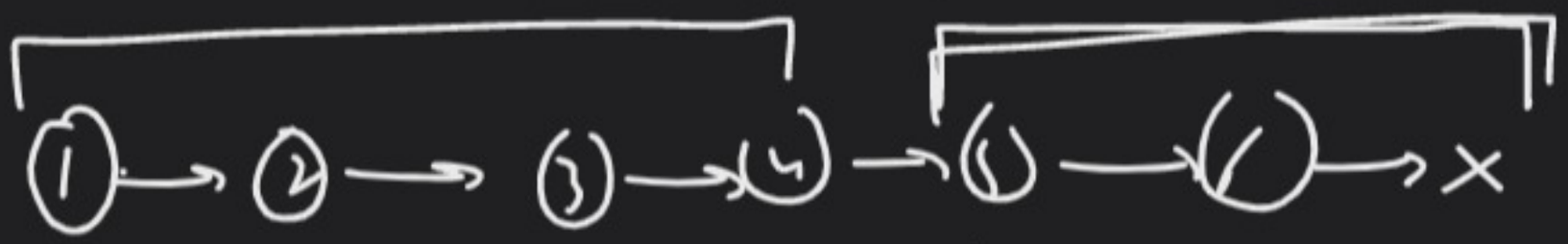
return prev;


head
prev
curr
forward
prev = curr
curr = forward
forward = curr->next
forward = curr->next
NULL- next
copy list


head
1
2
3
4
5
6
x
prev
curr
reverse
prev
curr
forward
prev
head
curr
reqAns
2
1
x
4
3
6
5
head → next = ans

K → right → K node exist

if (<K) nodes, don't traverse

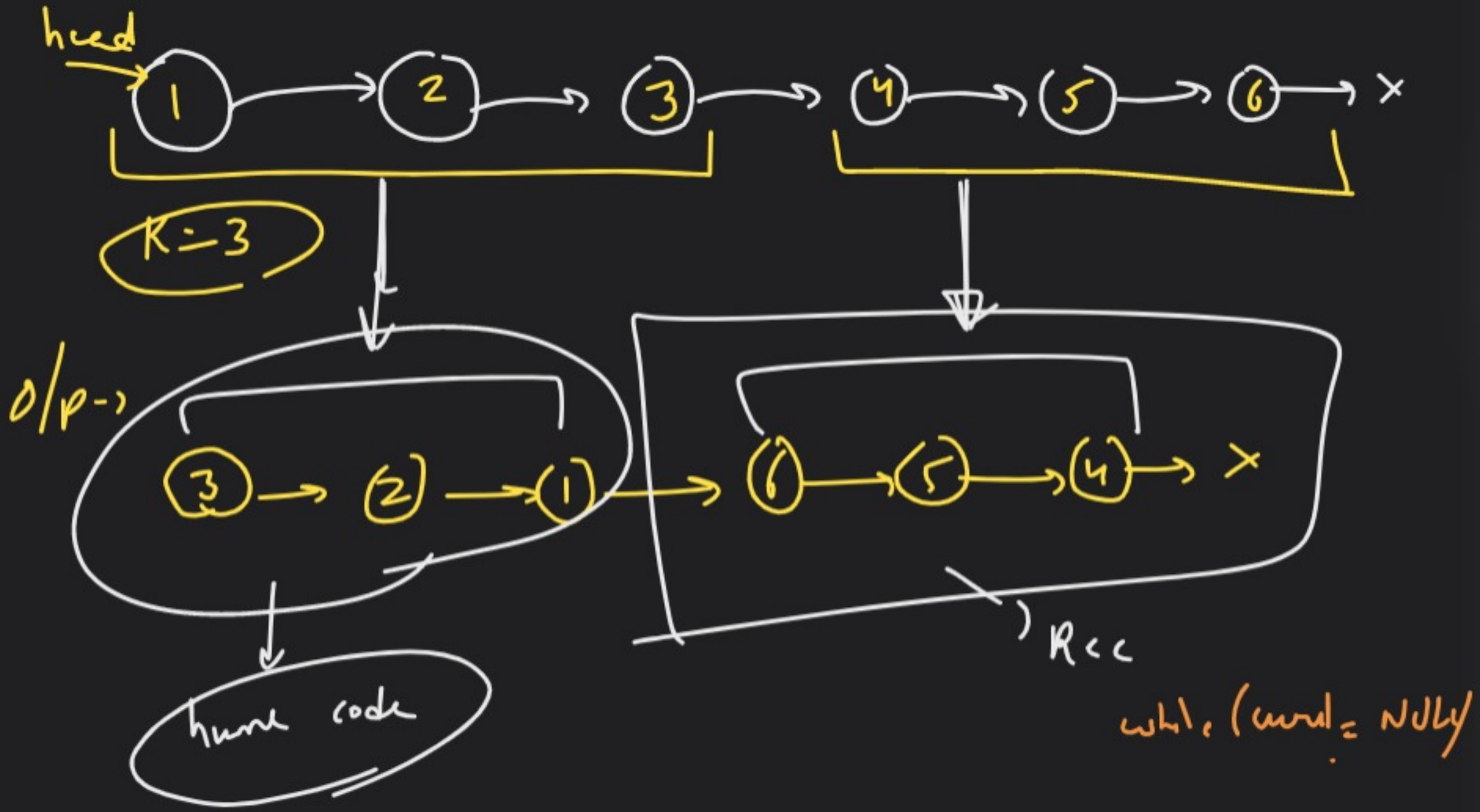


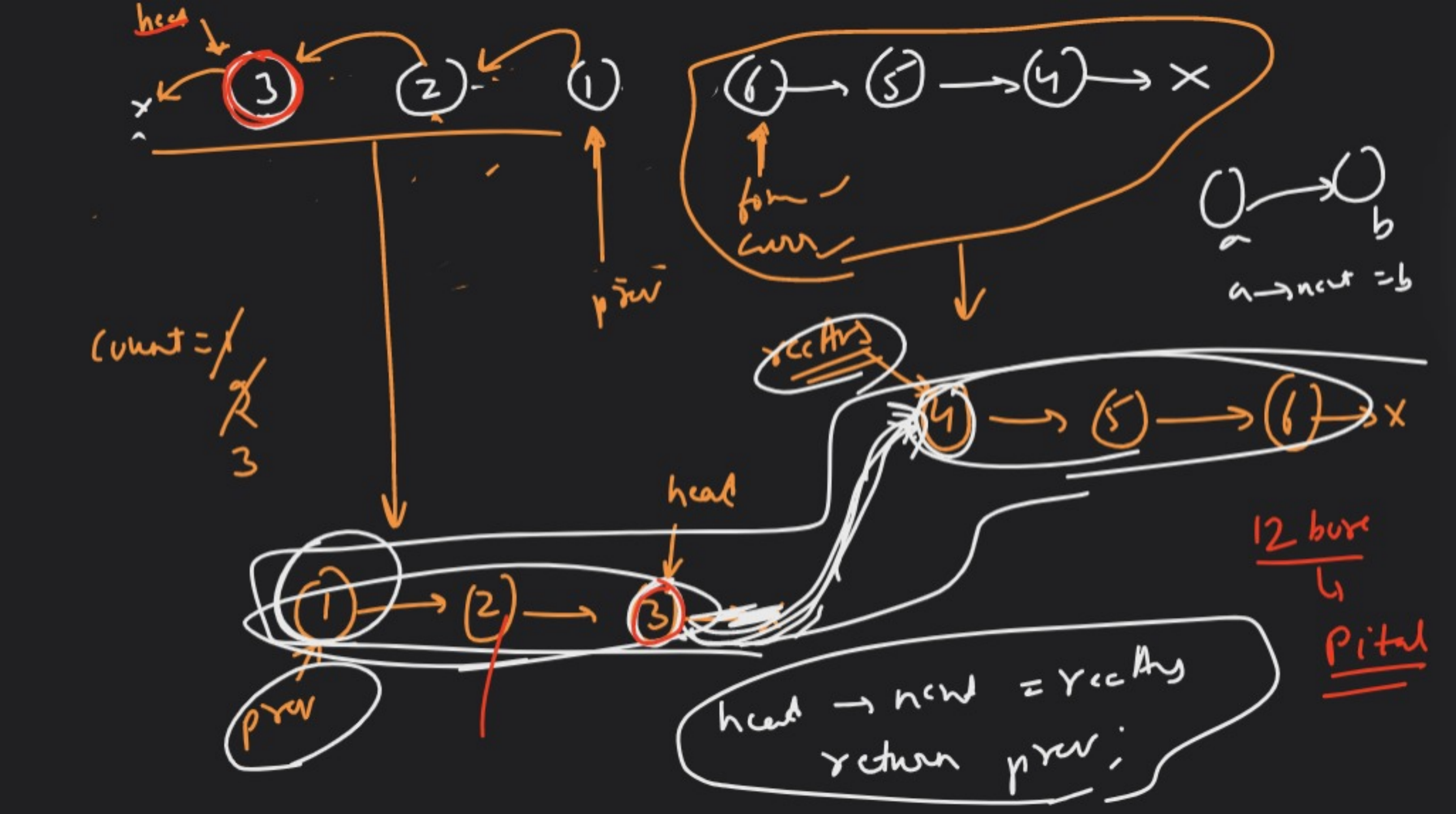
K = 4

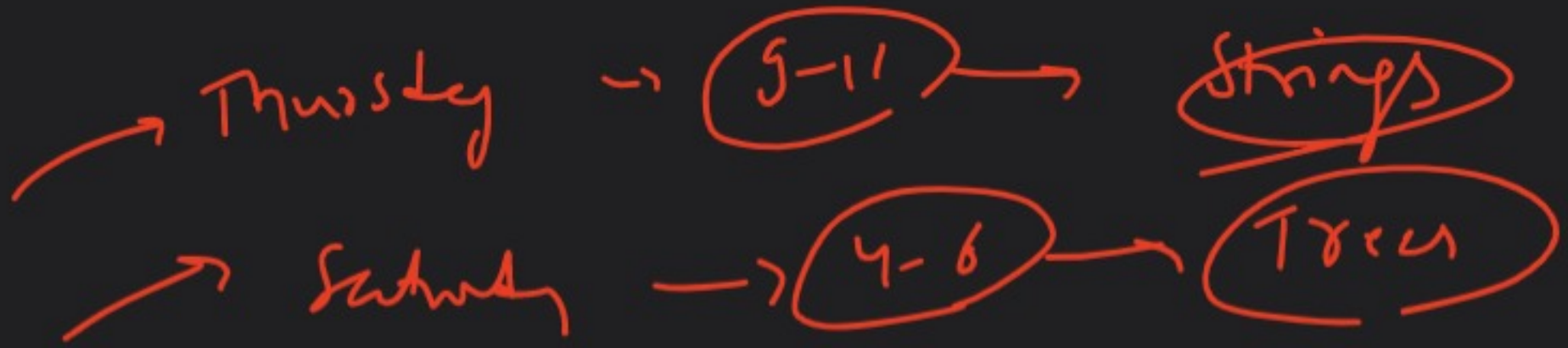
mera Ans → 4 → 3 → 2 → 1 → 6 → 5

rightth → 4 → 5 → 2 → 1 → 3 → 6

correct


head
1
2
3
4
5
6
X
K=3
o/p->
3
2
1
1
5
4
X
Rec
human code
while (curr != NULL


head
3
2
1
x
6
5
4
x
for curr
prev
a
b
a->next = b
current = 1
2
3
recAns
4
5
6
x
head
1
2
3
prev
head -> next = recAns
return prev;
12 bus
Pitol


Thursday
9-11
Strings
Saturday
4-6
Trees