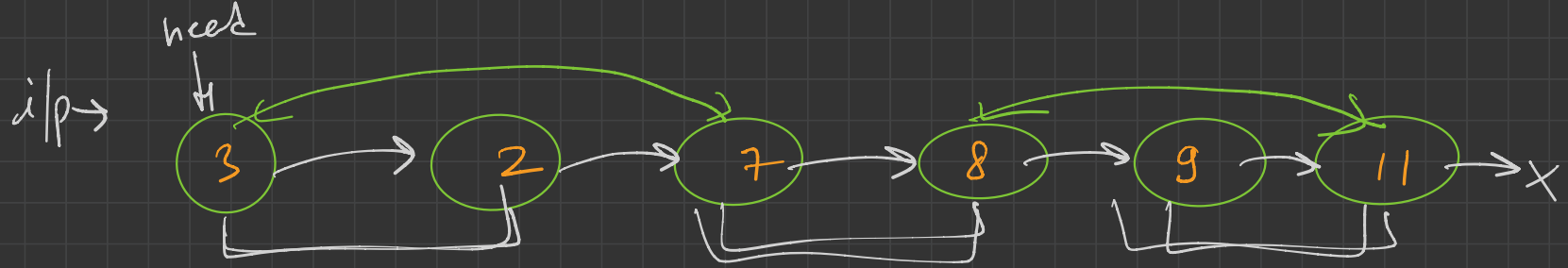
Linked List Questions

→ Reverse → 3/

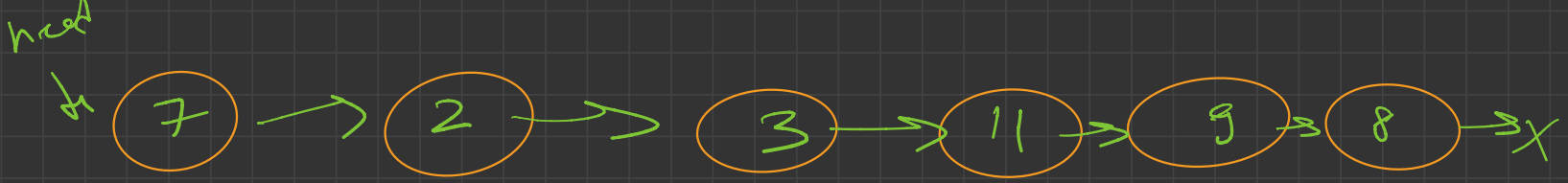
→ Middle → 2/

→ Linked List Reverse [K - groups]



K=2

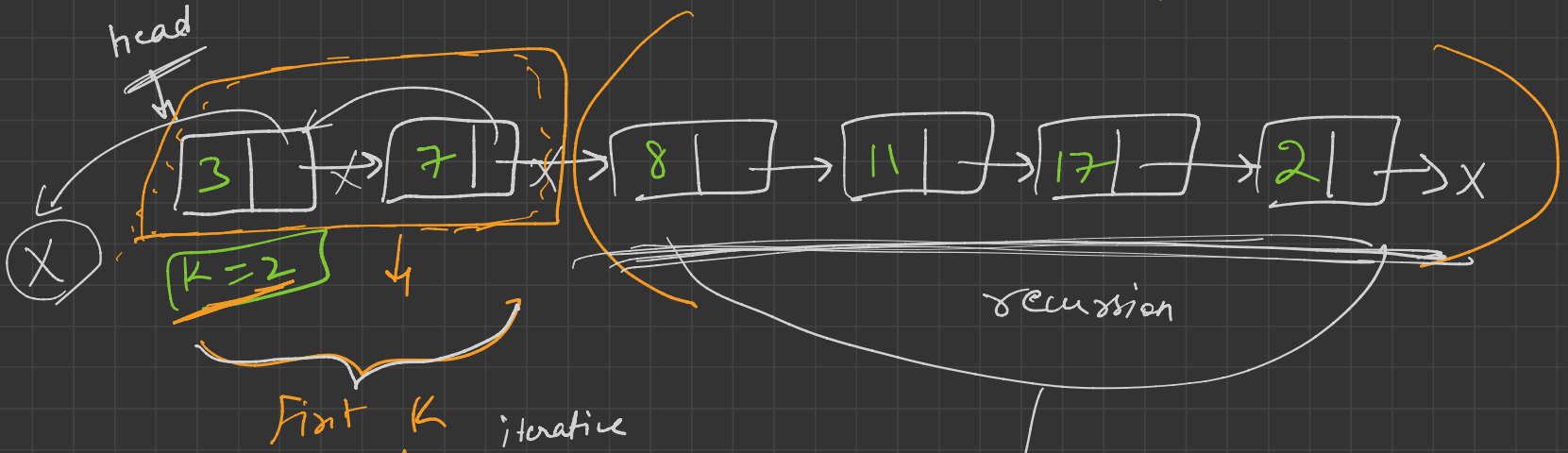
K=3



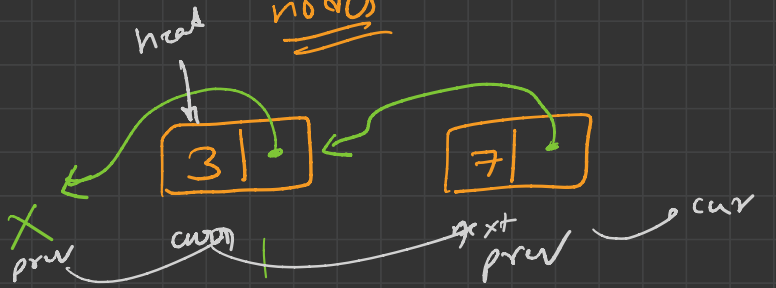
Approach :- ?

Approach:- Recursion → I solve Krdo

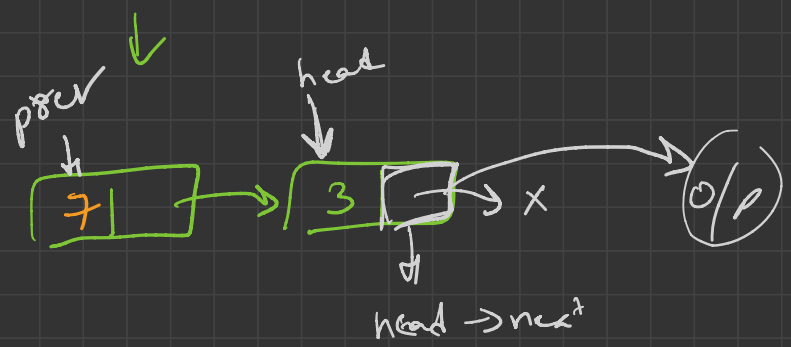
↳ (Baaki main Krdunga)

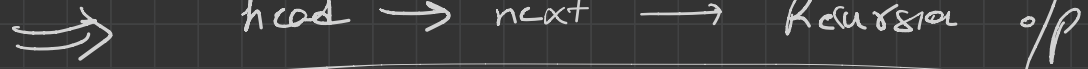


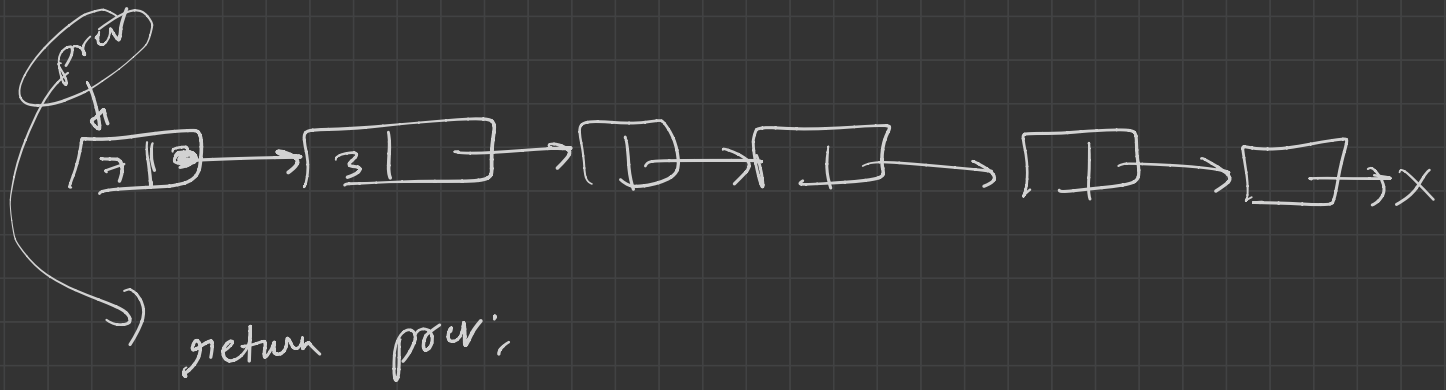
First K nodes iterative



O/P


prev
head
7
3
X
O/P
head -> next


head → next → Recursion O/P


prev
7
3
X
return prev;

Algo:-

→ 1 case solve krnge

→ iterative algo (count < K)

↓

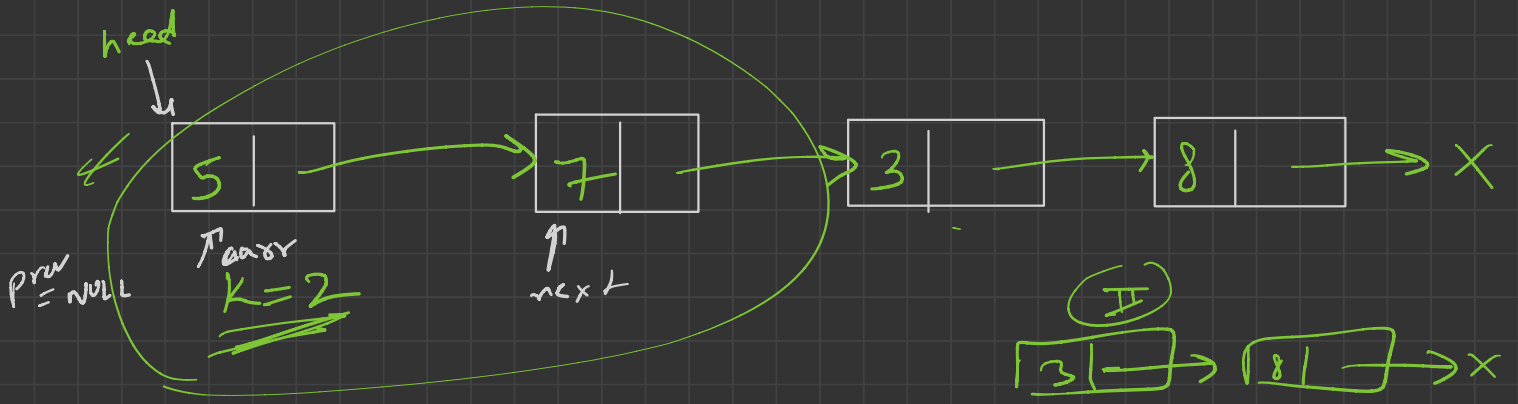
(First k node reverse)

→ head → next = recursion call

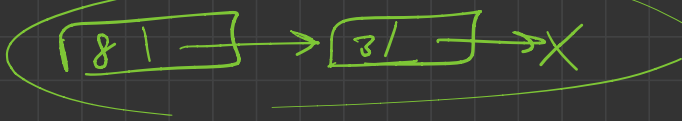
→ return head of reversed LL

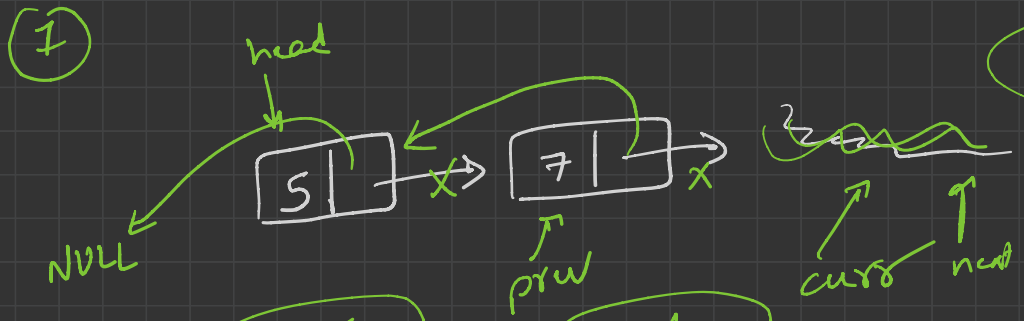
↓

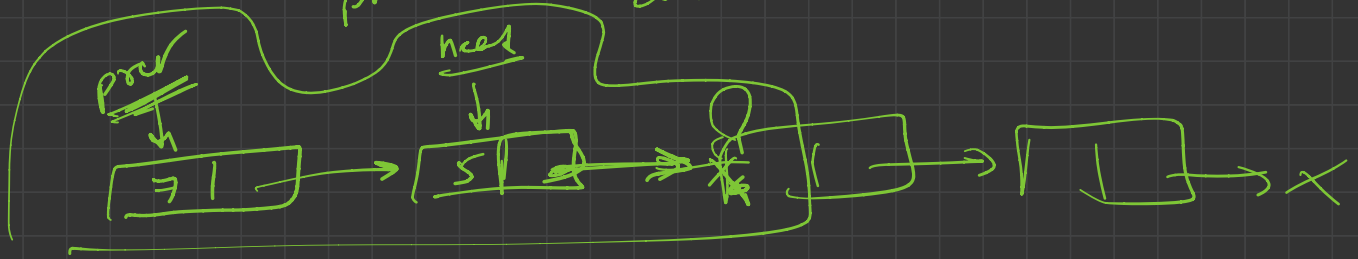
return prev


head
5
7
3
8
X
prev = NULL
curr
K=2
next

II
3
8
X

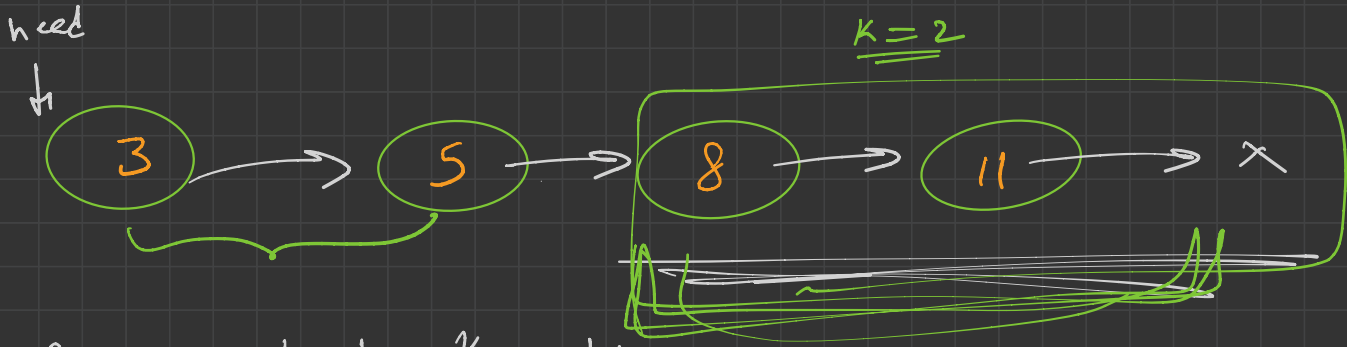

8
8
X


I
head
5
7
X
X
NULL
prev
curr
next

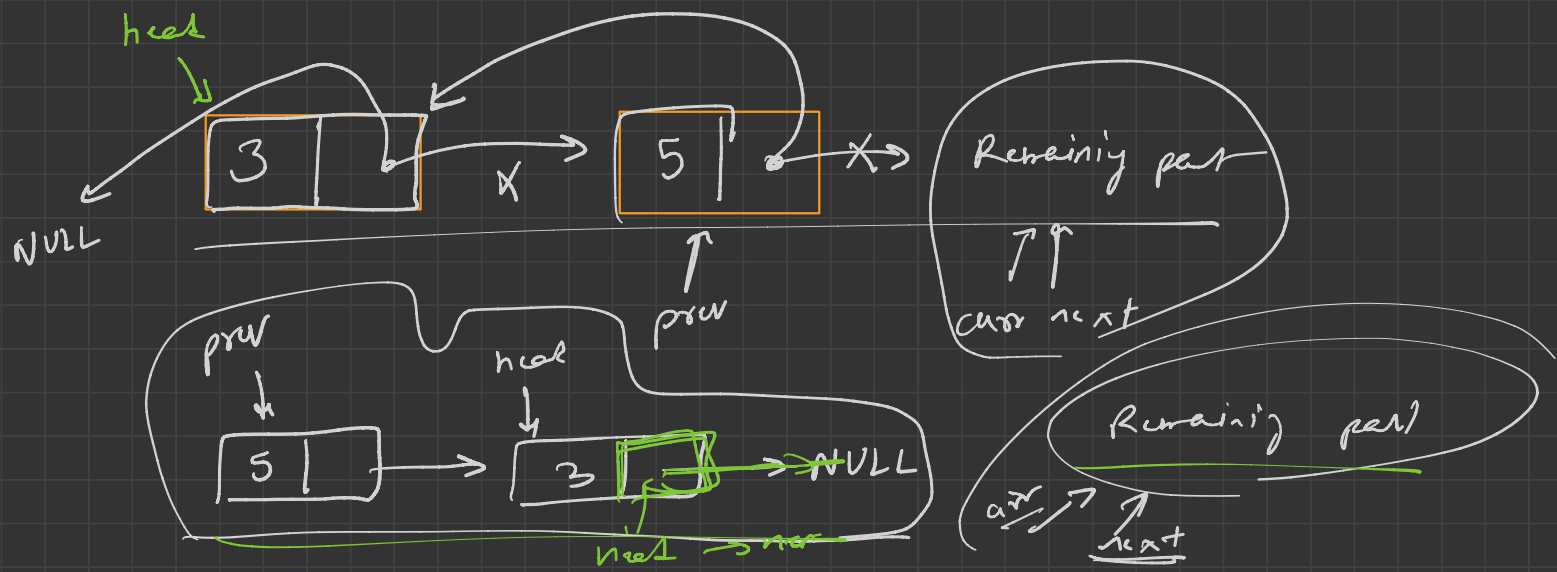

prev
next
7
5
1
X

head → next → Recursion call

vector push



① Reverse first K nodes



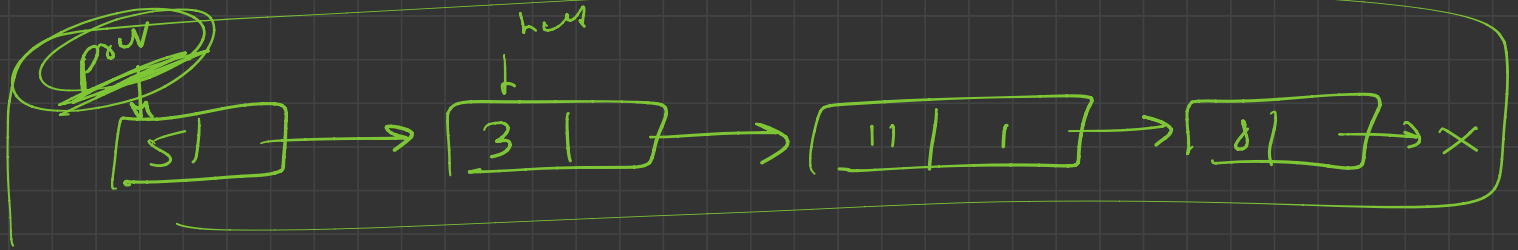
Step II

Remaining part → reverse → recursion

o/p → head of reverse list

head of reversed remaining part

```
head -> next = kReverse(next, k);
```



Step 3

return prev;

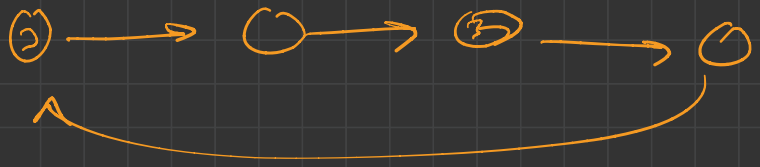
T.C → O(n)

S.C → O(n)

k nodes → reverse

(2)

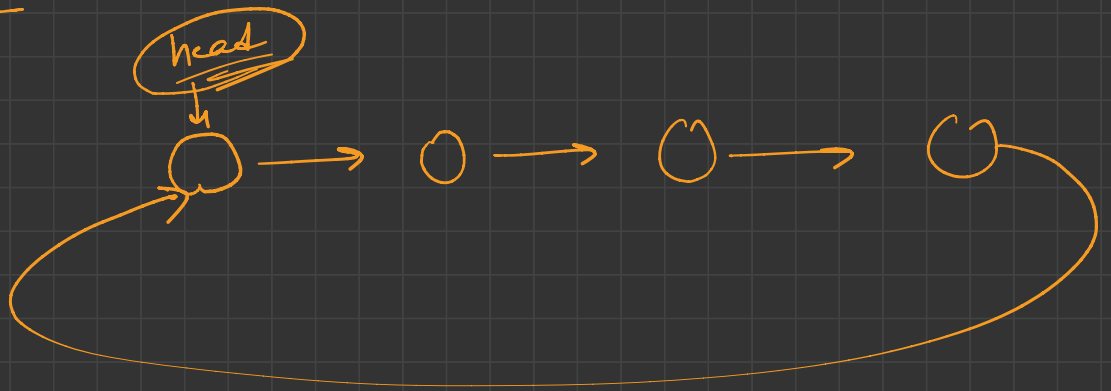
i/p ->



o/p → Circular or Not

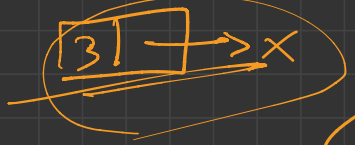
T or False

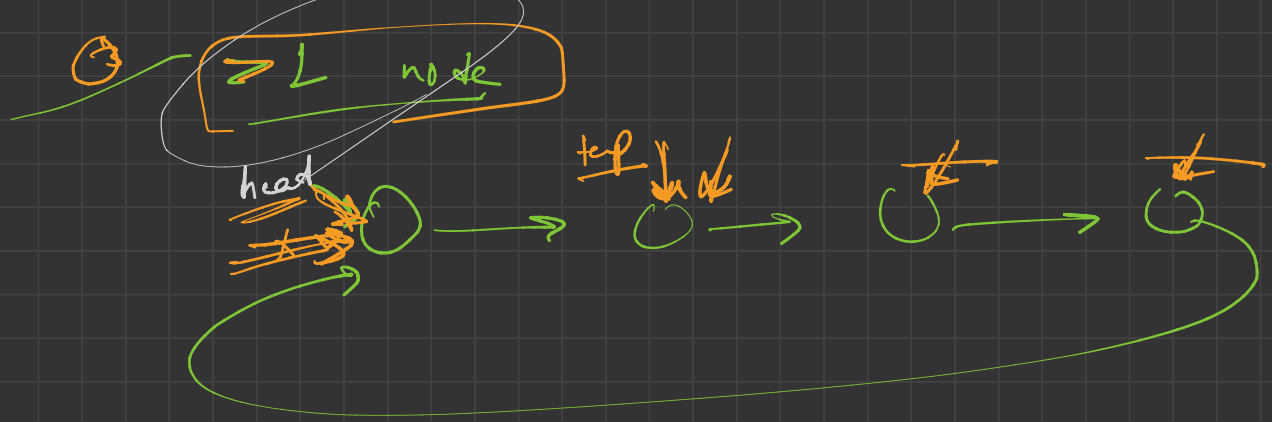
Approach 2 :-



① Empty List ⟶ if (head == NULL)
return True

② ⟶ if (head → next) == NULL



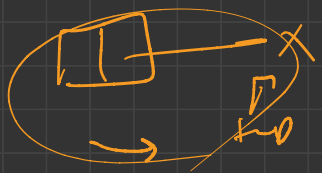


LL → NULL

→ non-circular

→ circle

same node → 2 ptr



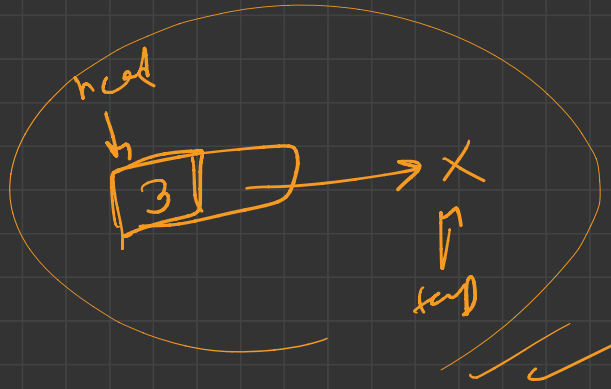

```
Node *temp = head -> next;

while ( temp != NULL && temp != head )
{
    temp = temp -> next
}
```

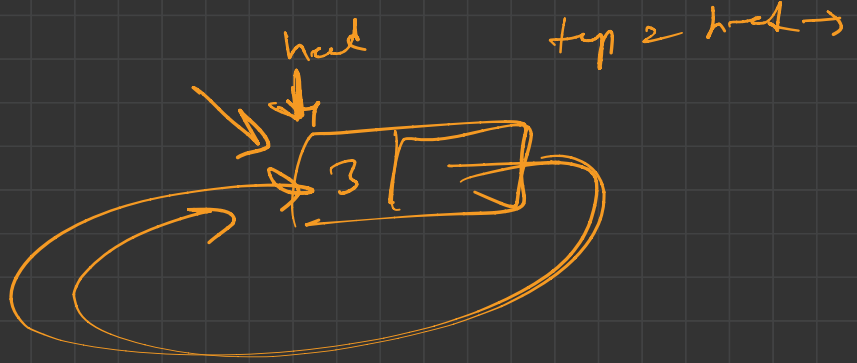
if (temp == NULL) (Non Circ

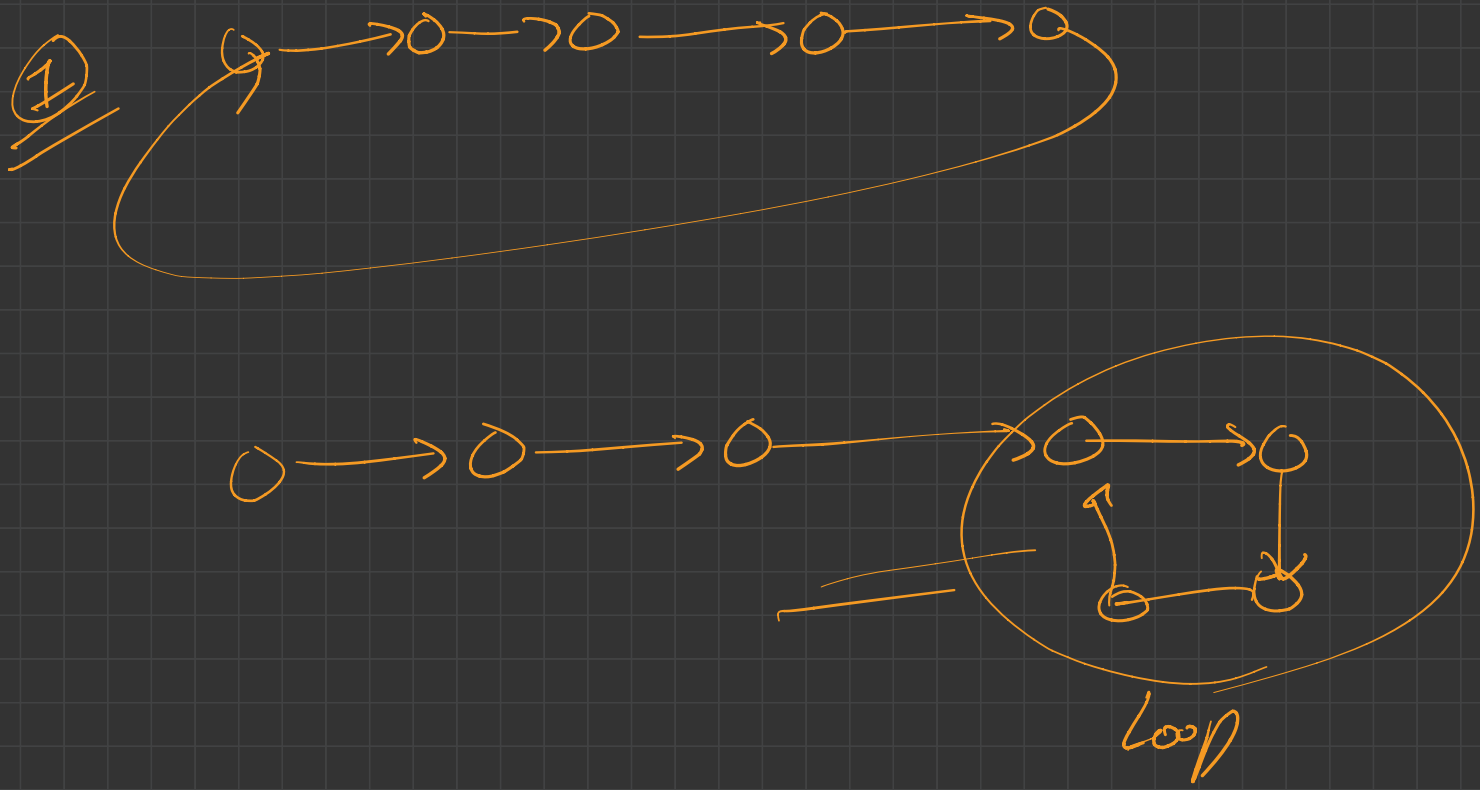
return false;

if (temp == head)

return true;






Loop

→ n nodes

T.C → O(n)

S.C → O(1)

Log n²

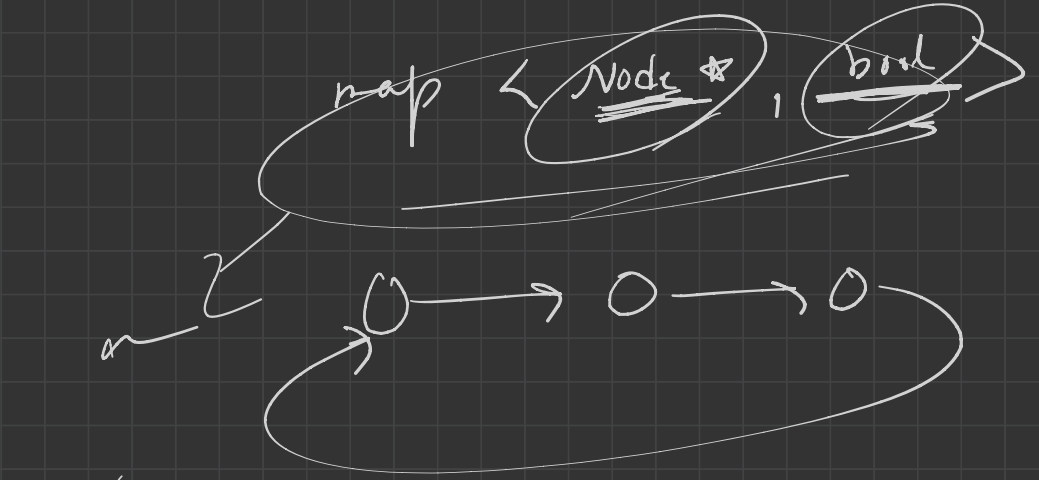
map →

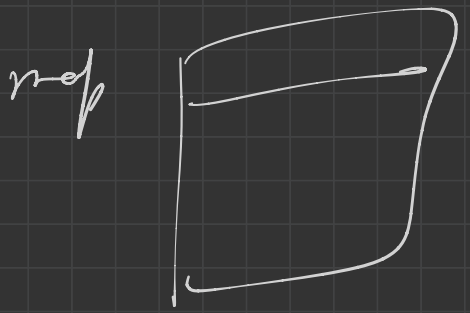
map → D.S

Key, Value

Key → value

Key2 → value



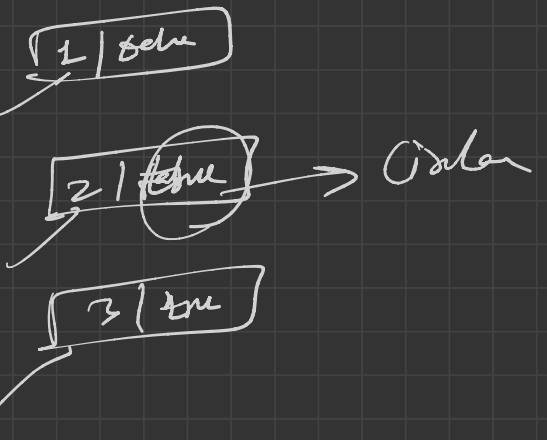


T.C → O(n)

visit

S.C → O(n)

NULL
↓
No + cir



→ Floyd's Cycle detection algo:-