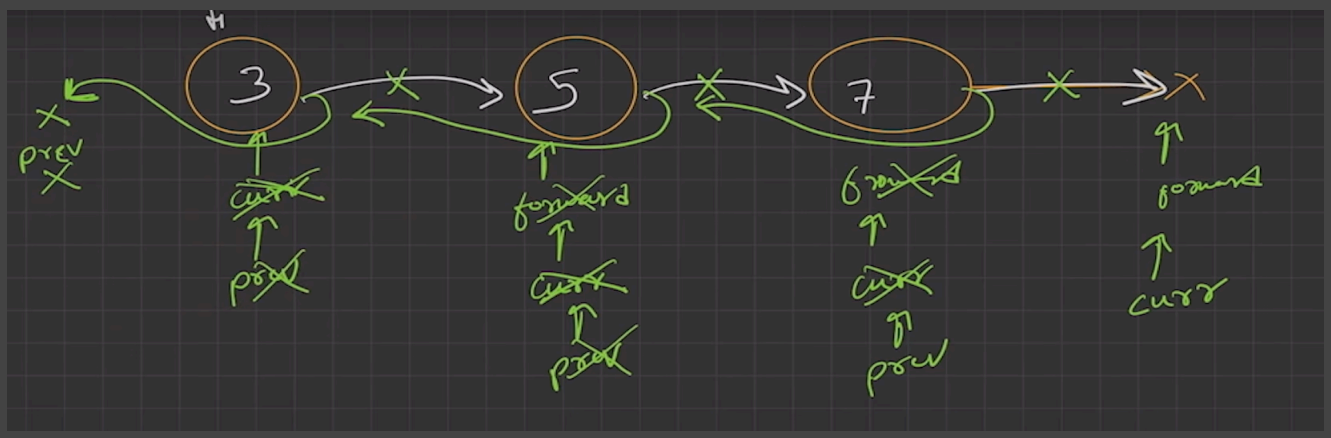
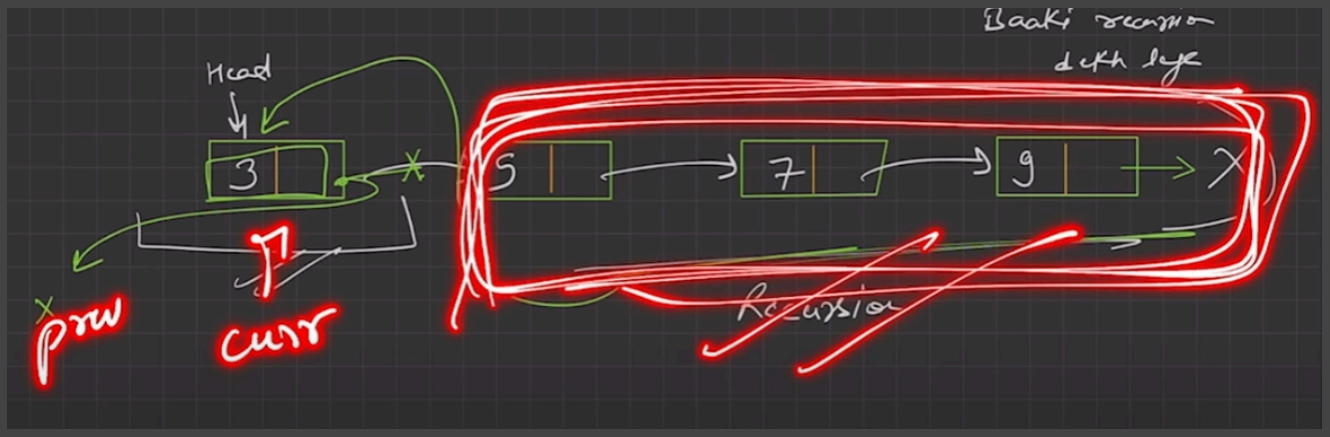
Linked-List

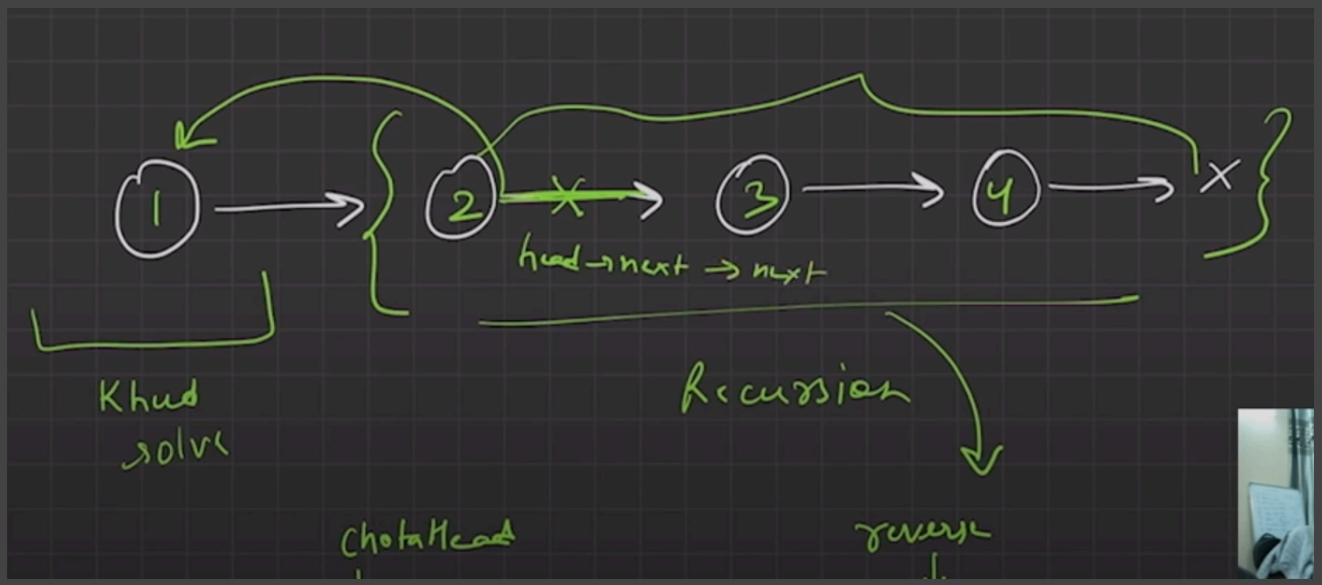
Reverse A Singly Linked List Approach-I:



Reverse A Singly Linked List Approach-II:

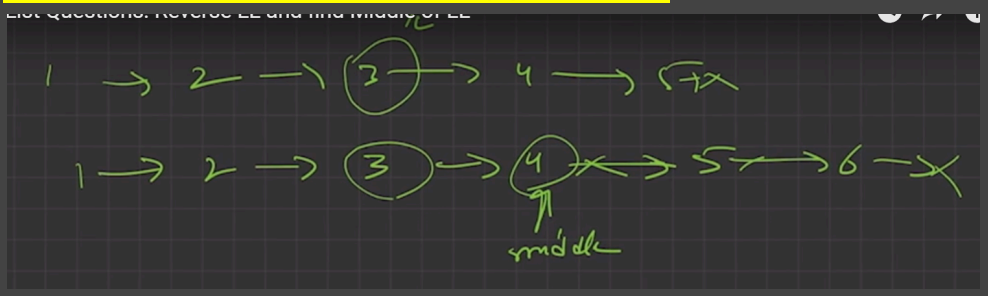


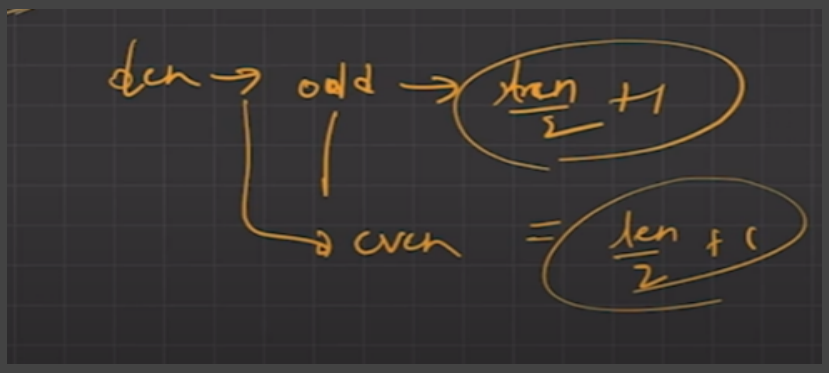
Reverse A Singly Linked List Approach-III:



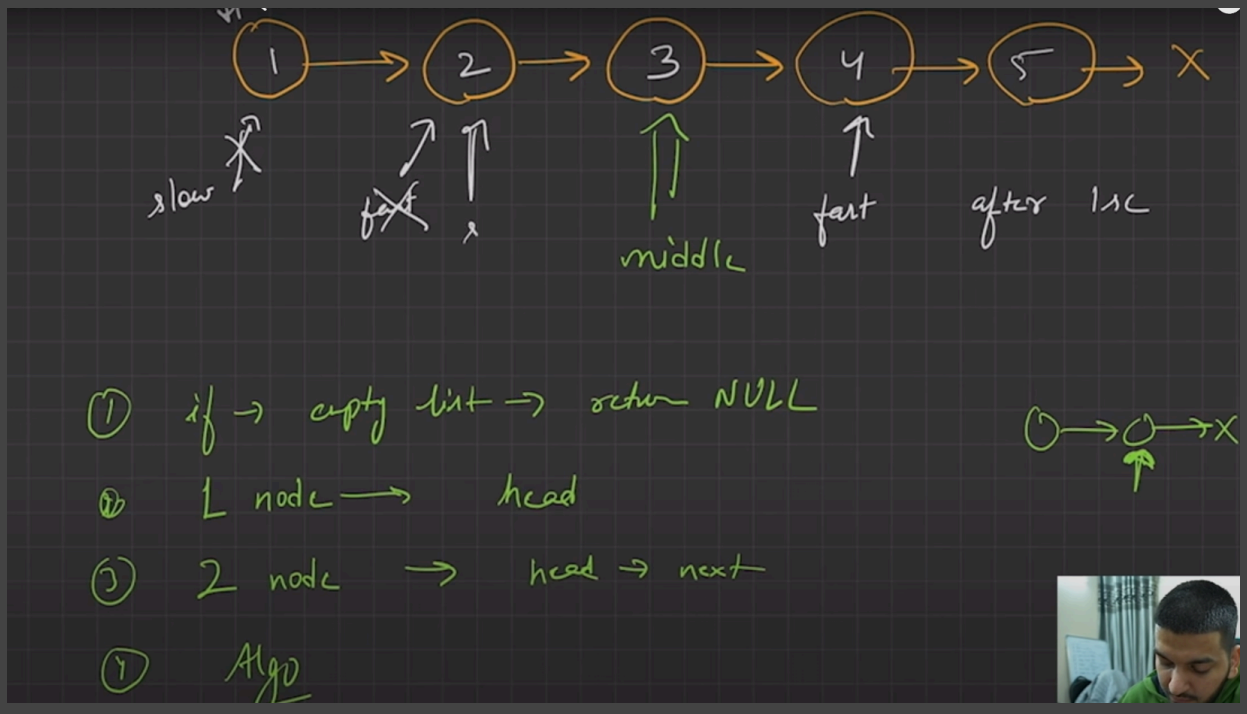
Reverse A Doubly Linked List

Find Middle In A Linked List Approach-I:

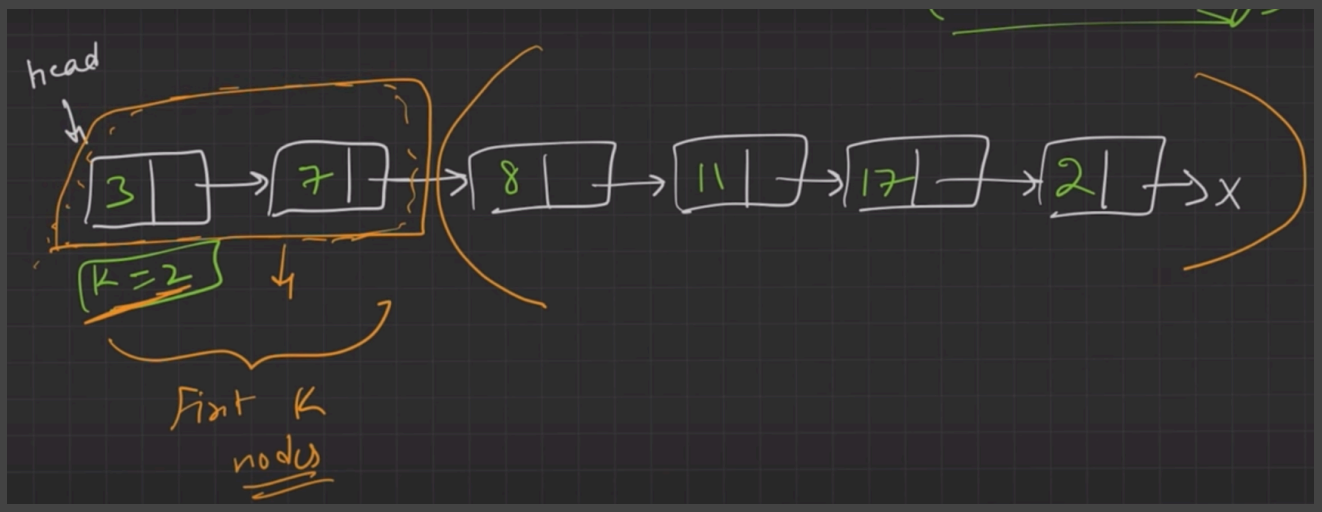


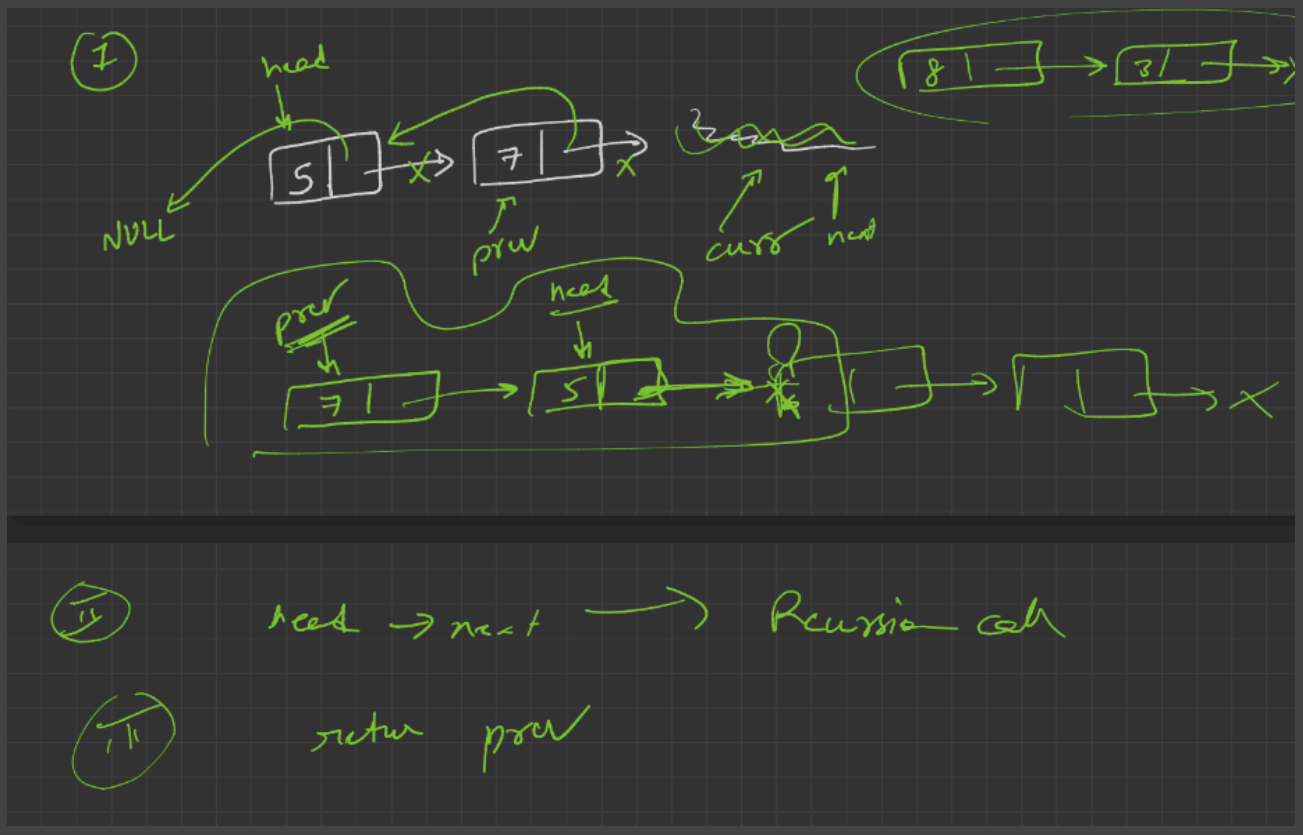


Find Middle In A Linked List Approach-II:

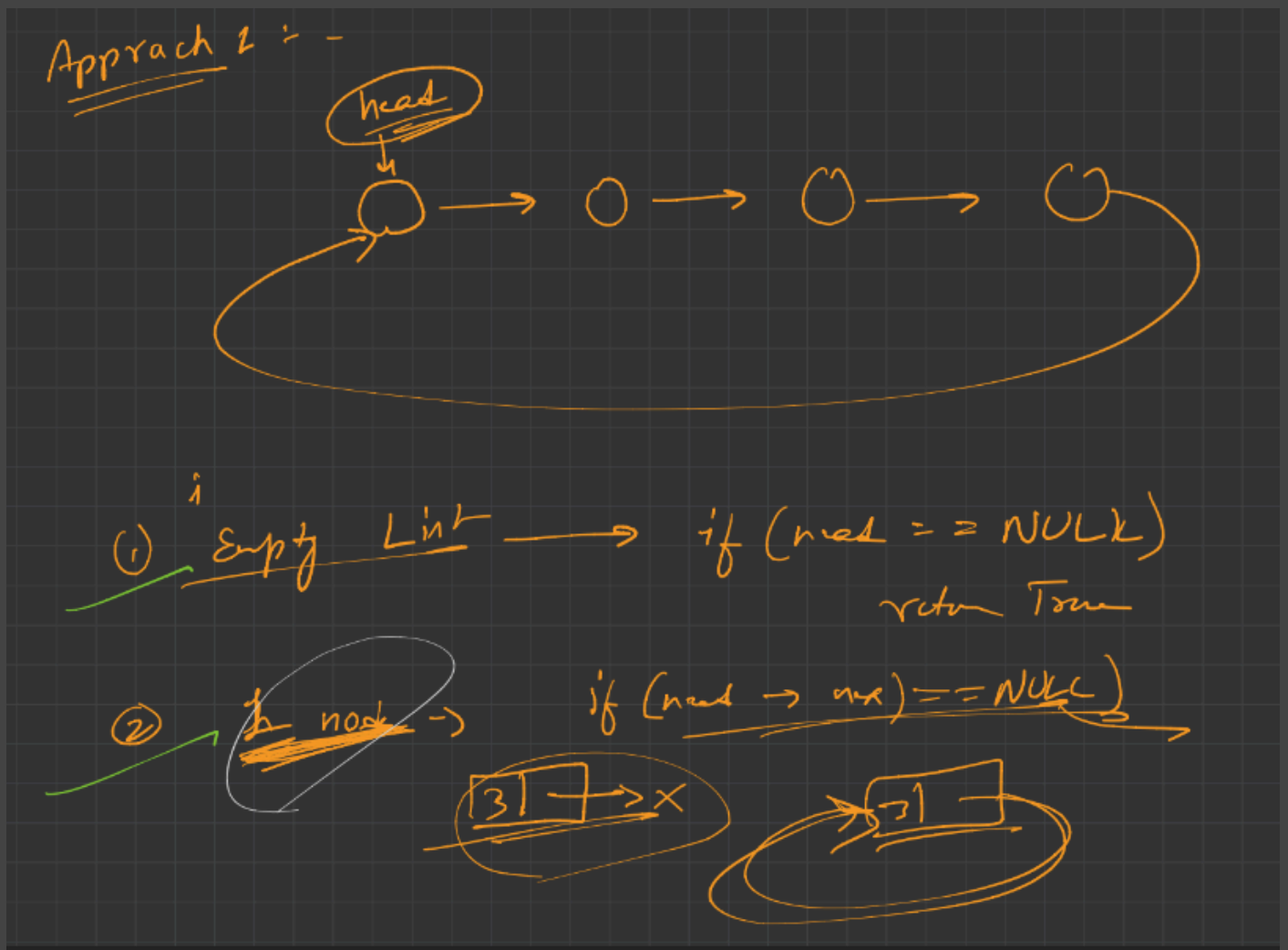


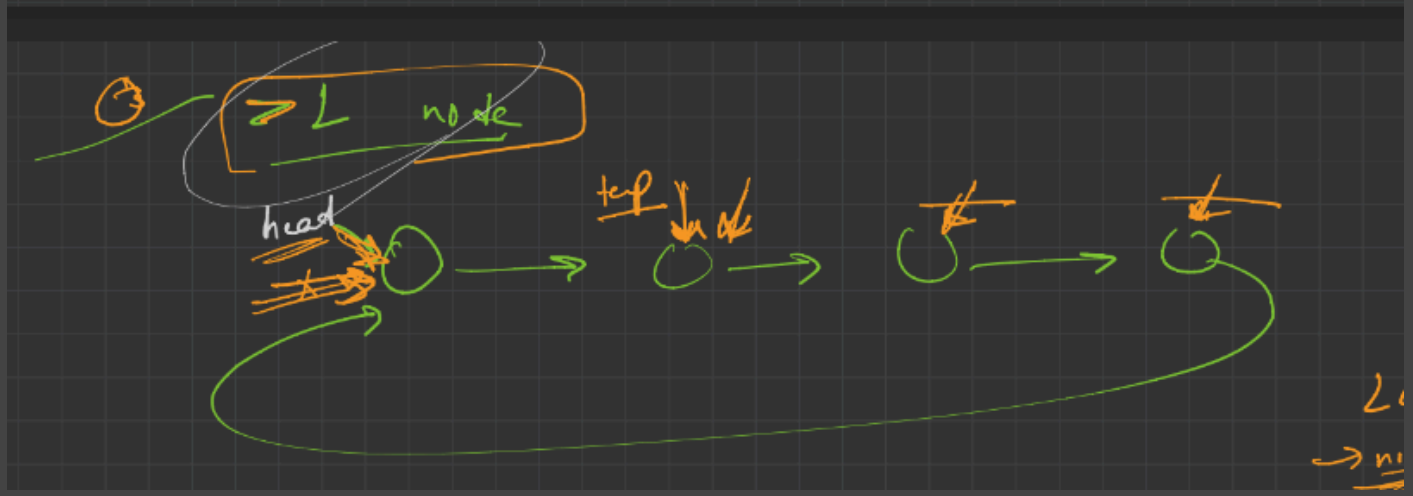
Reverse Linked List In 'K' groups:



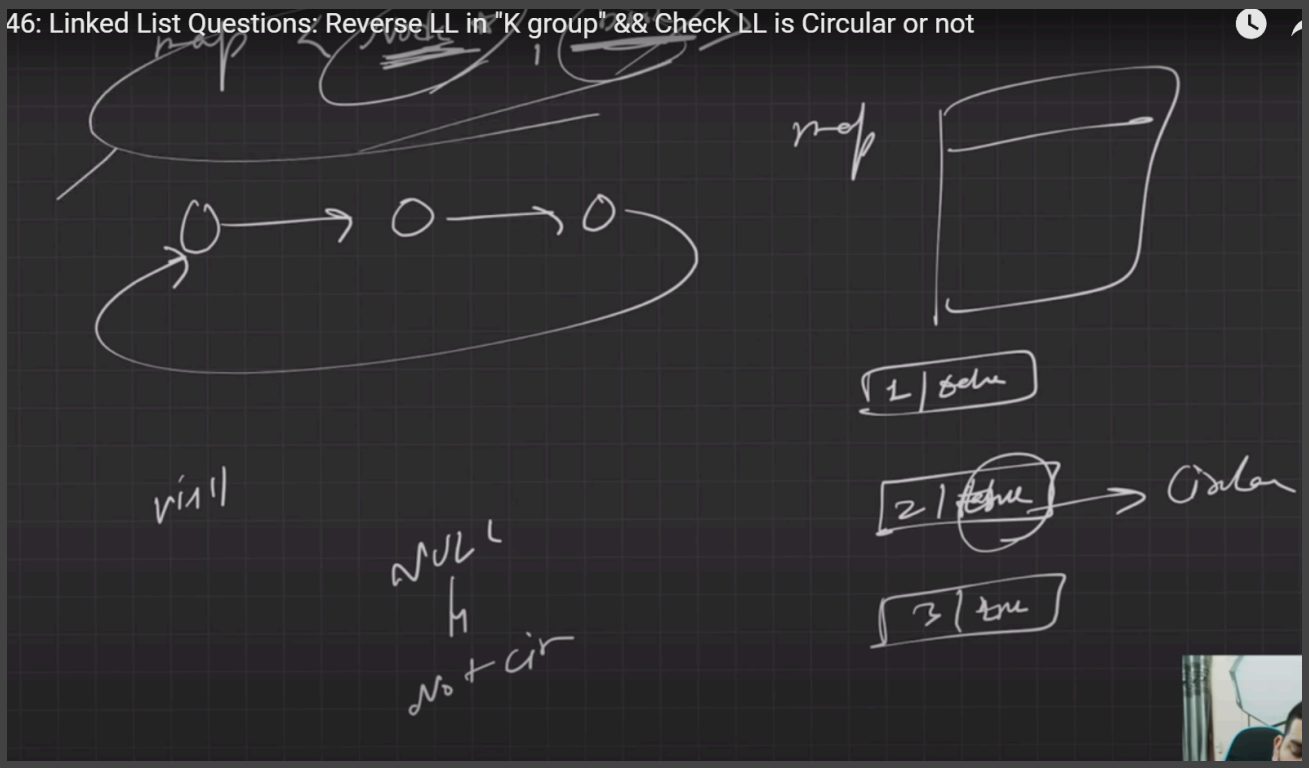


Check Whether A Linked List is Circular or Not Approach-I:

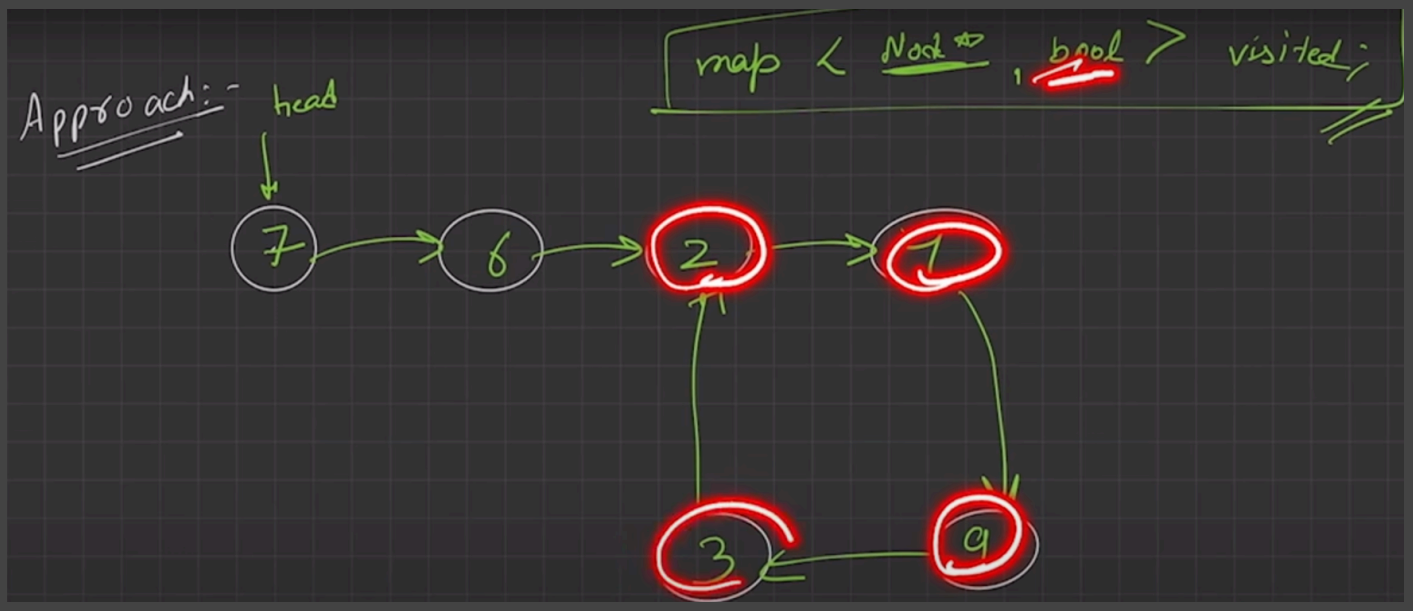




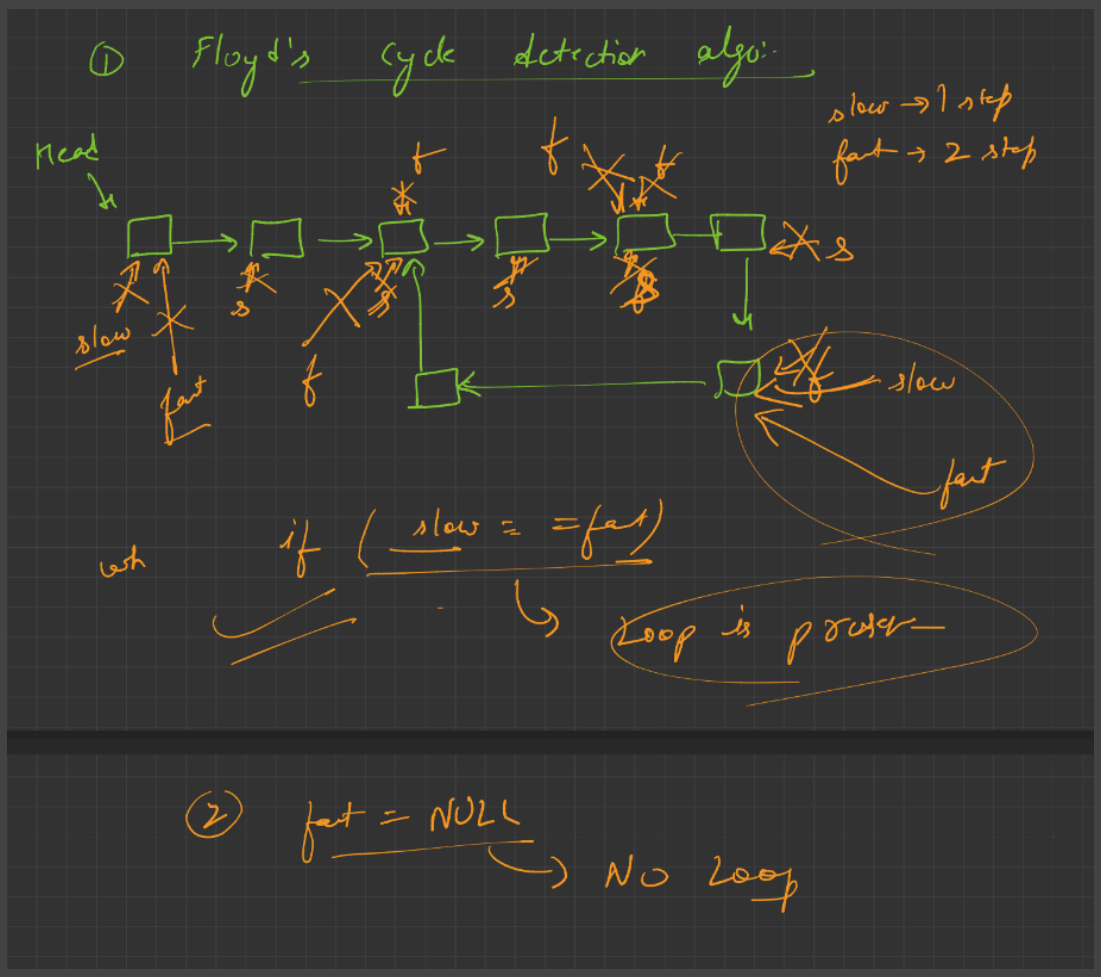
Check Whether A Linked List is Circular or Not Approach-II:

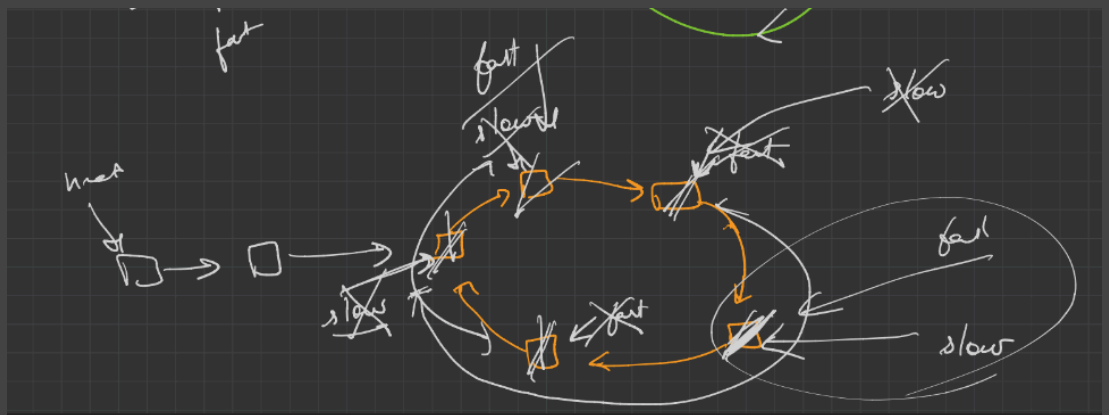


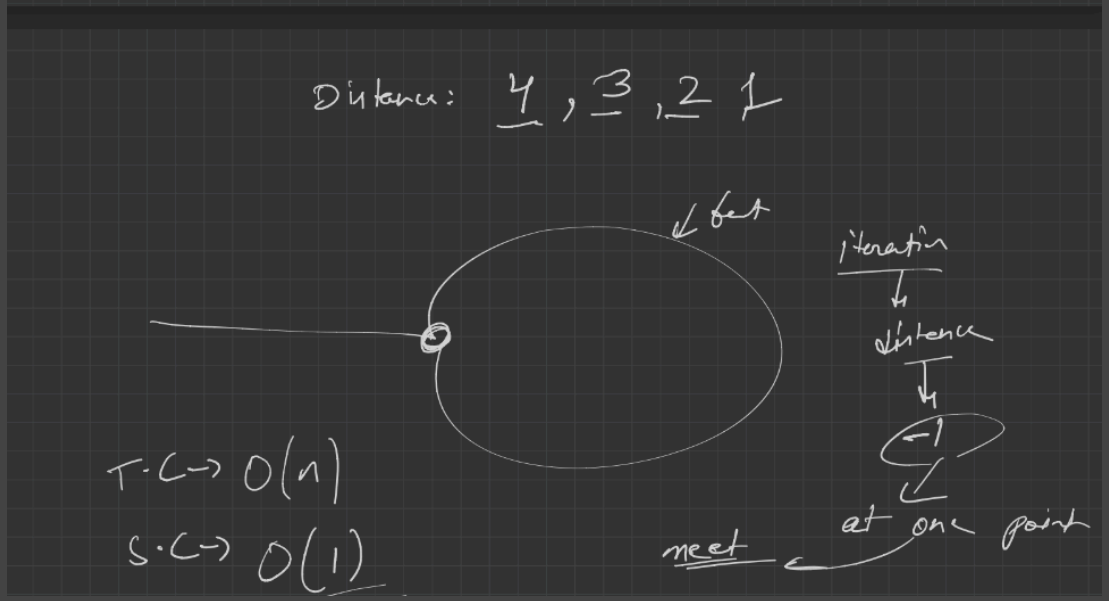
Detect Whether A Linked List Forms a Loop or Not:



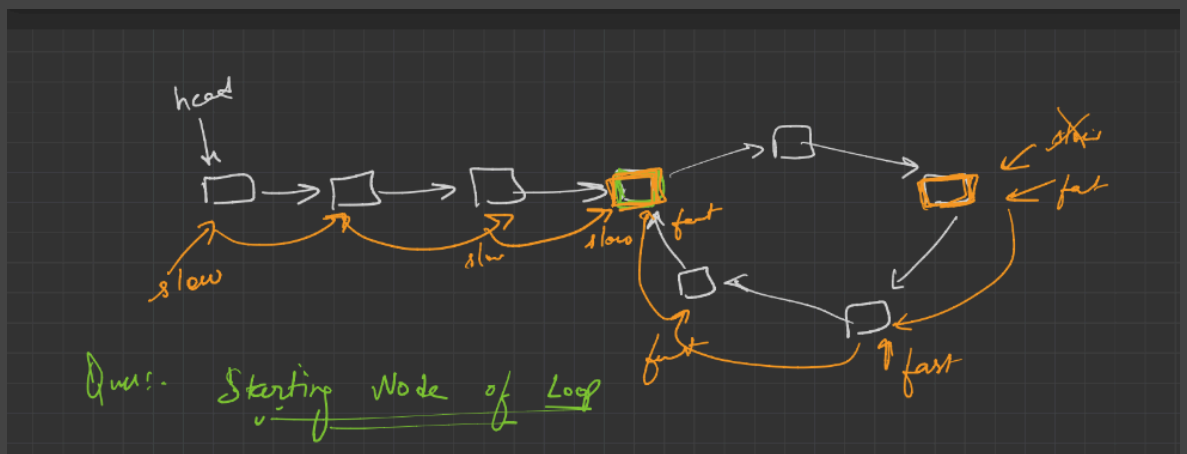
Floyd's Cycle Detection:







Find The Starting Node Of The Loop:



Ques: Starting Node of Loop

Approach:-

I → FCD Ifo → Point of Intersection

II → slow = head

↳ slow, fast → same pace
1, 1/1

↙

while(slow == fast)

↳ starting point of Loop

Distance by fast pointer = 2 * Distance by slow pointer

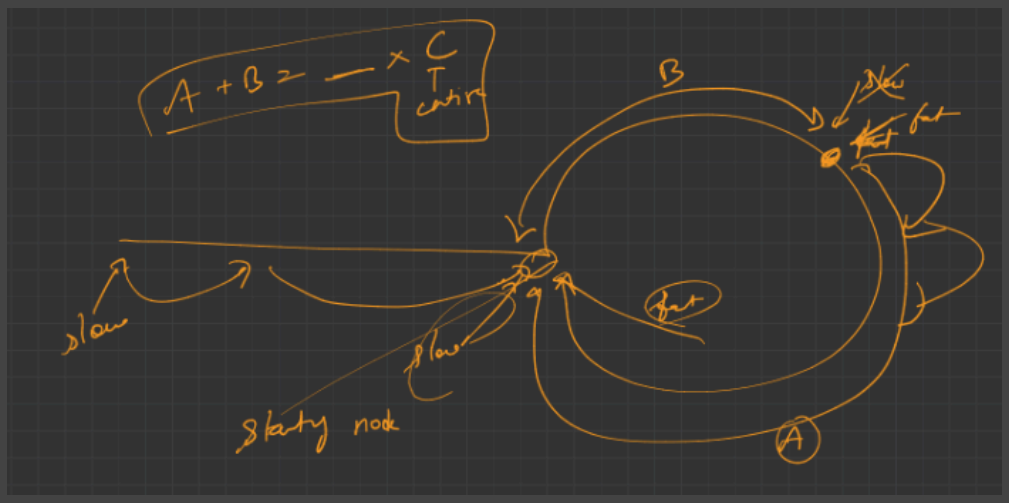
$$(A + x * C + B) = 2 * (A + y * C + B)$$

$$A + xC + B = 2A + 2yC + 2B$$

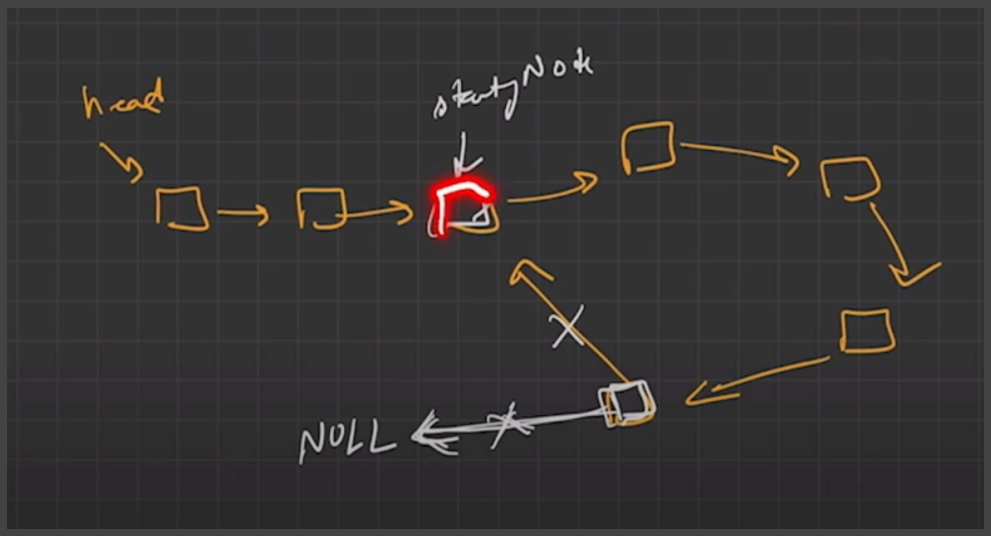
$$C(x - 2y) = A + B$$

$$\frac{C(x-2y)}{K} = A + B$$

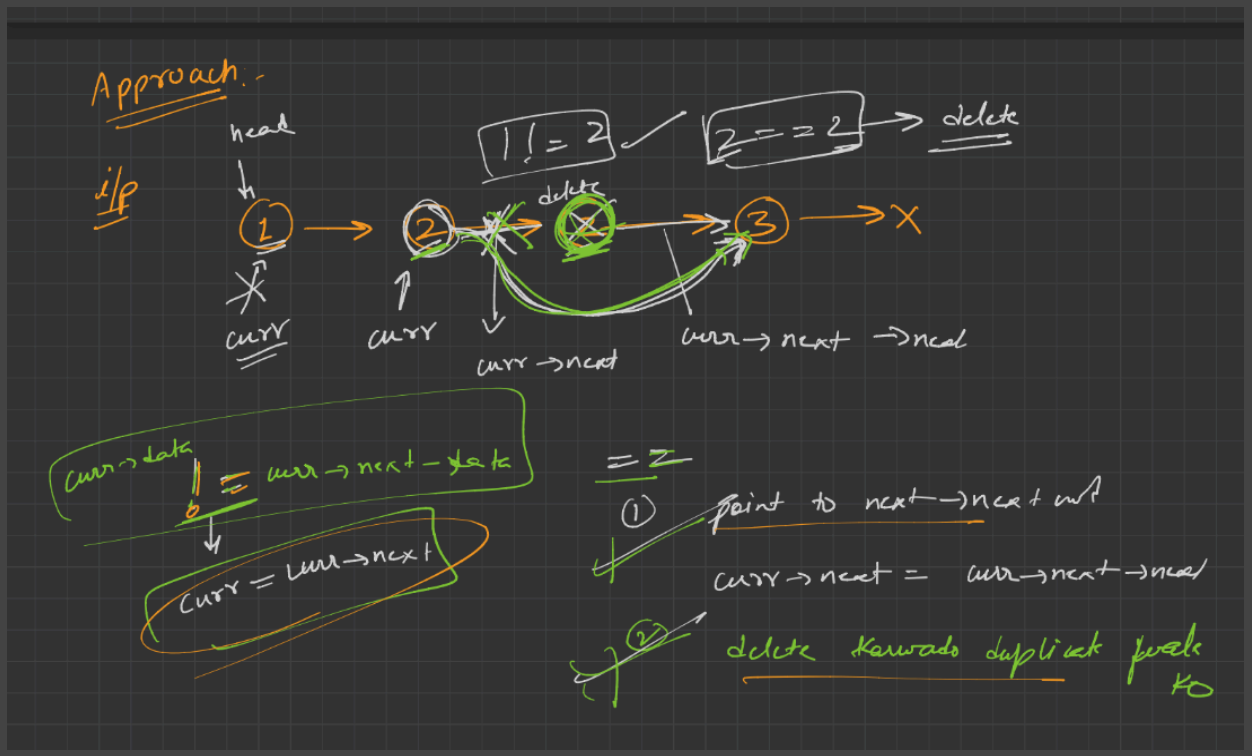
A + B = K times C



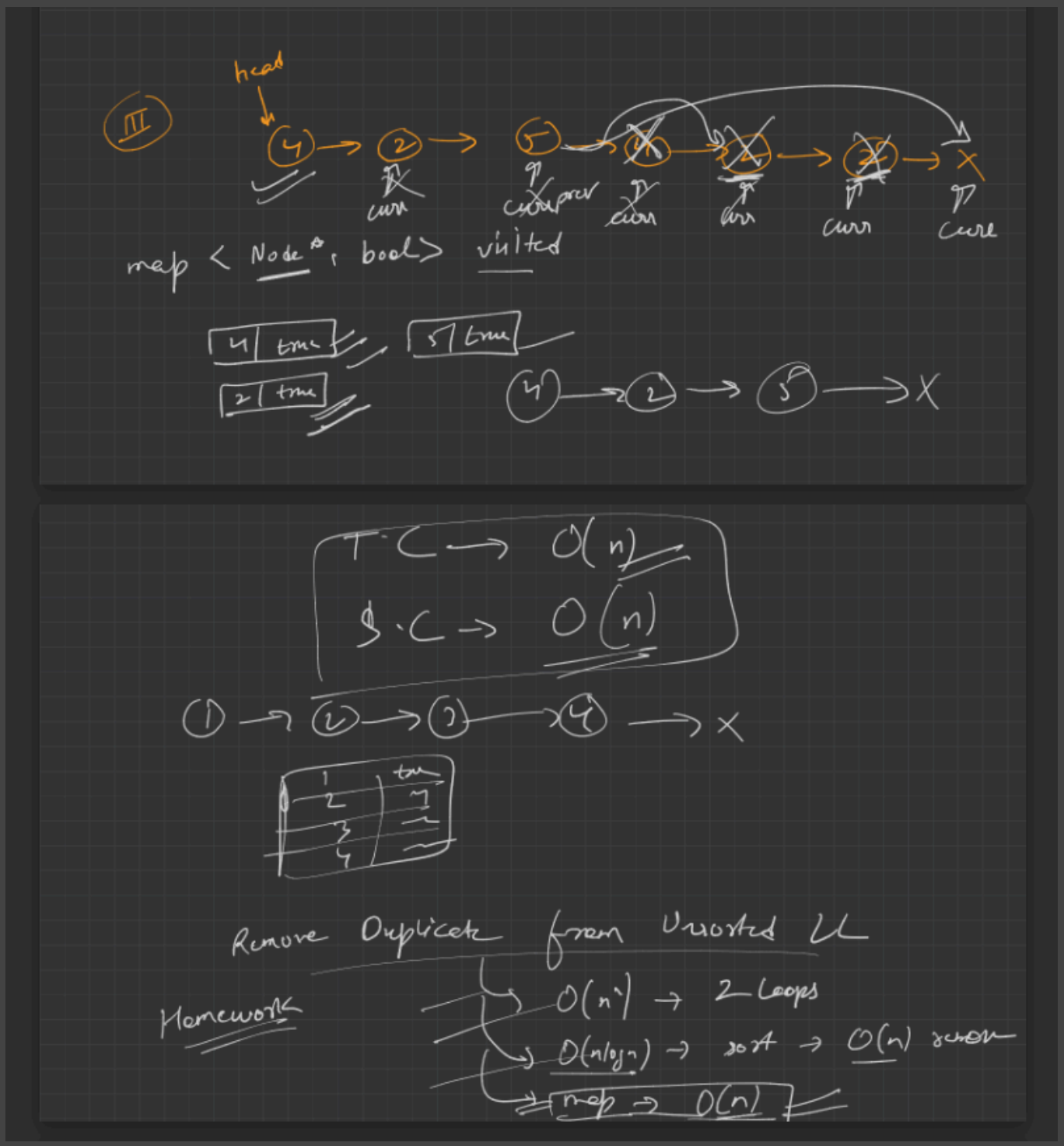
Remove The Loop:



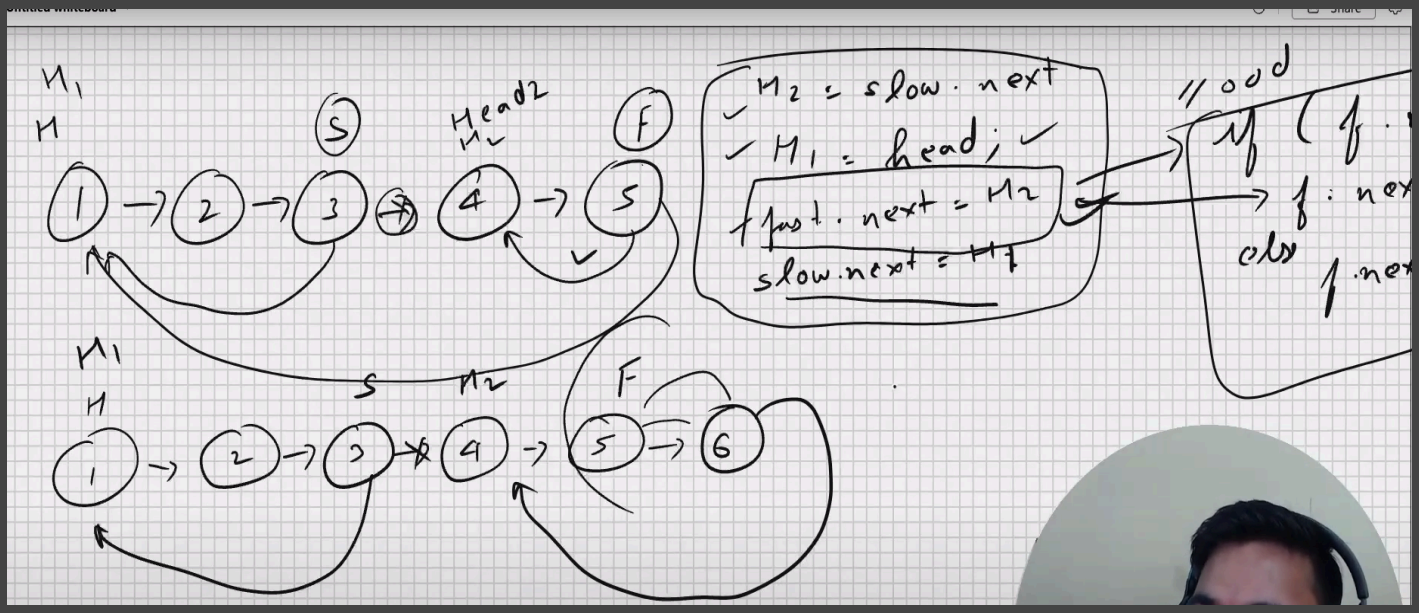
Remove Duplicates From A Sorted Linked List:



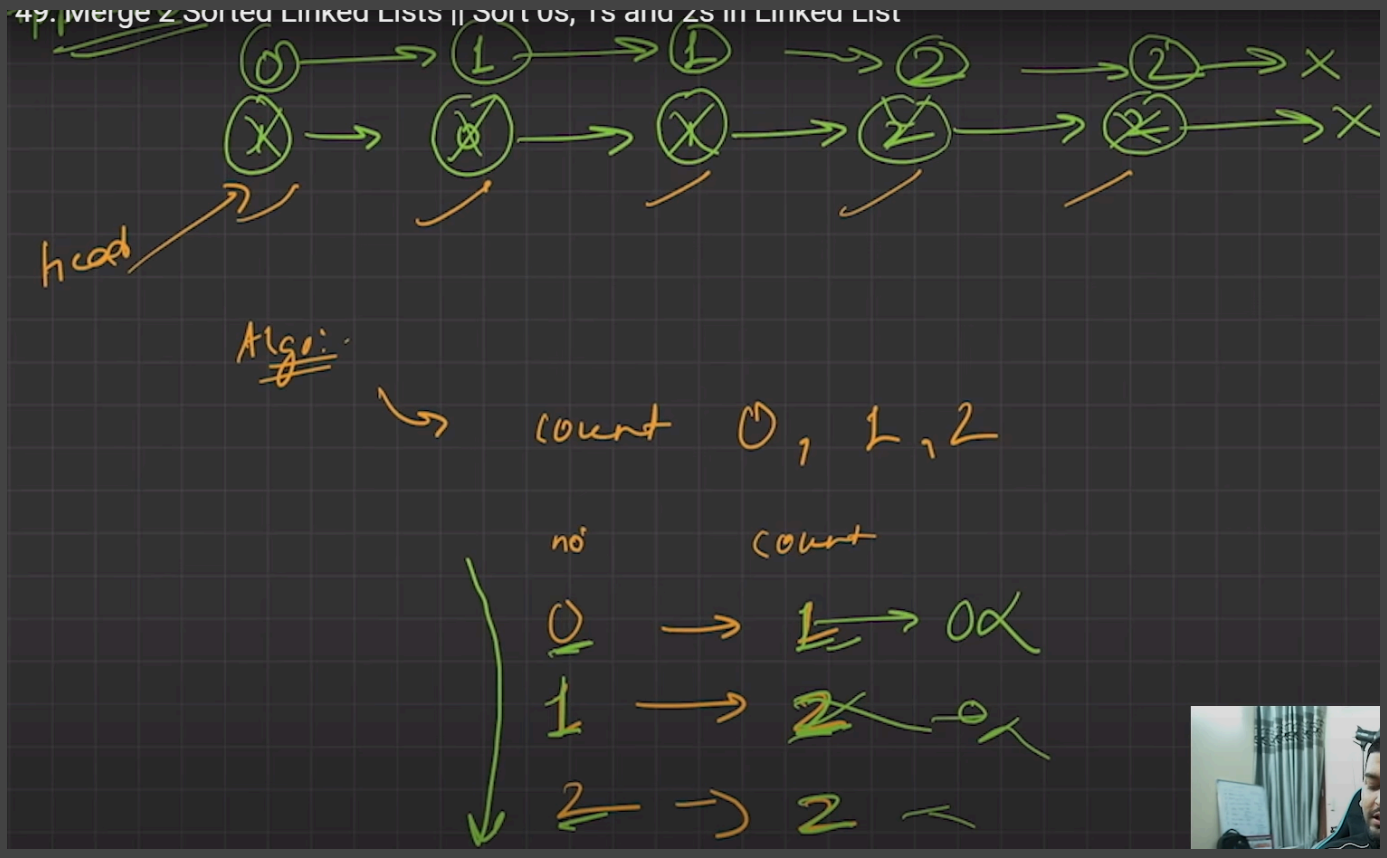
Remove Duplicates From A Unsorted Linked List:



Split A Circular LL Into Two Halves:



Sort 0s,1s & 2s In A LL Approach-I:



Sort 0s,1s & 2s In A LL Approach-II: