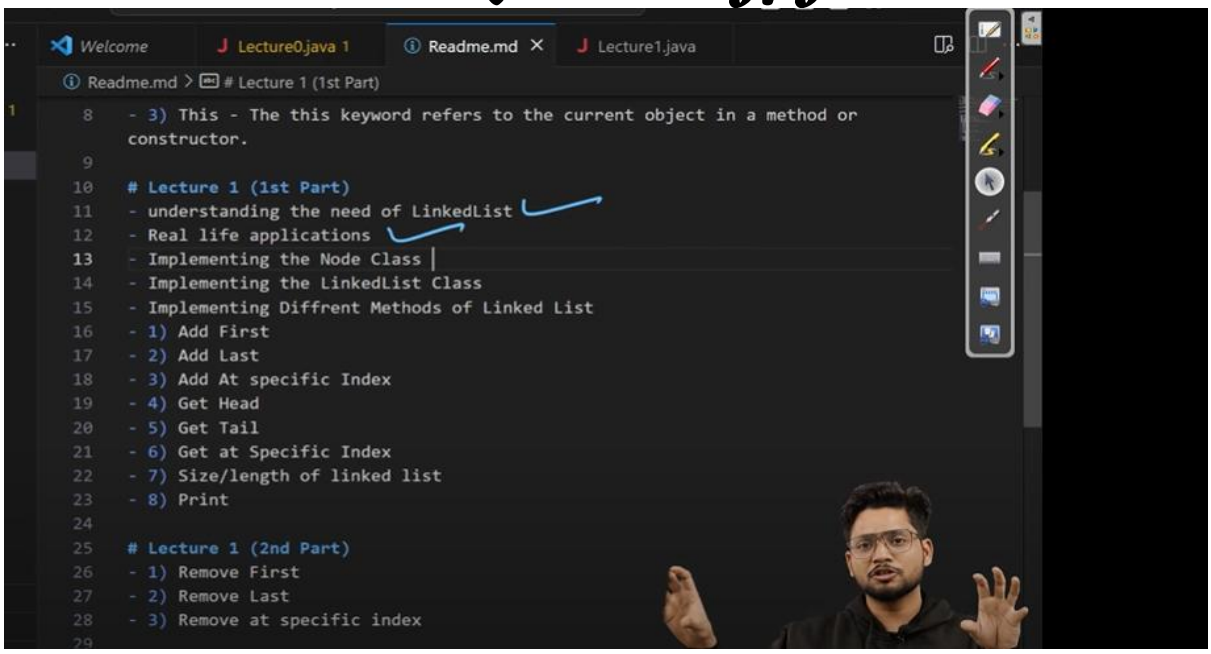
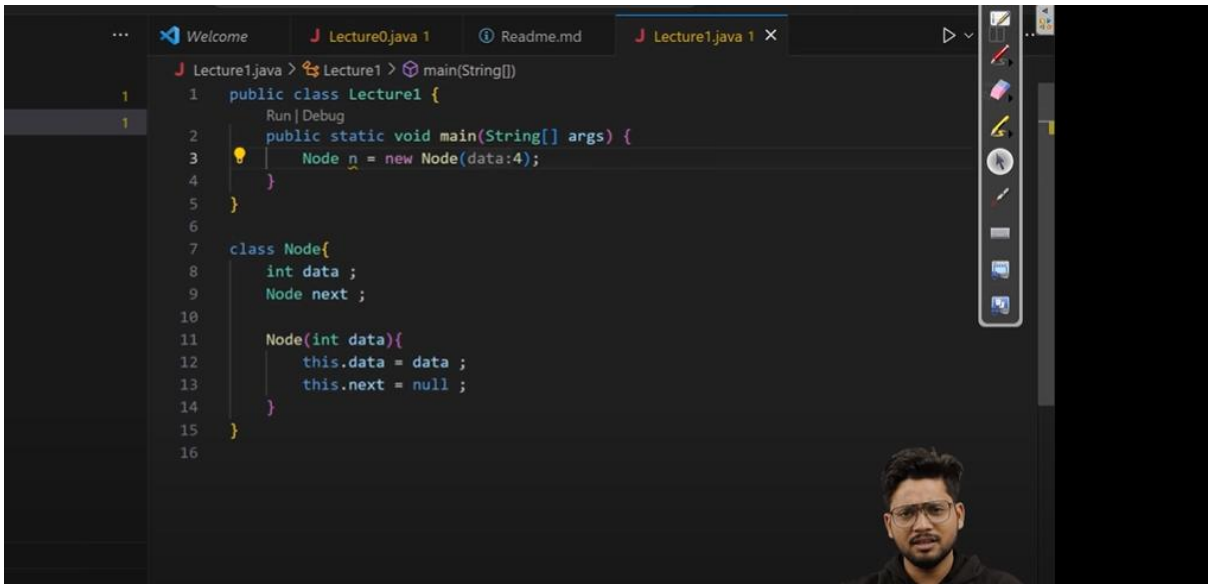
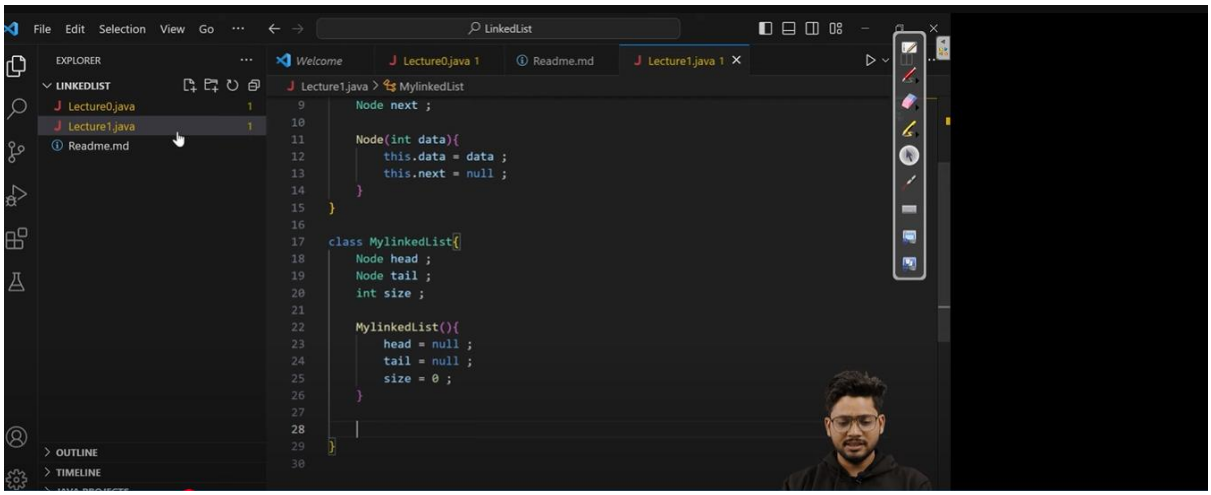
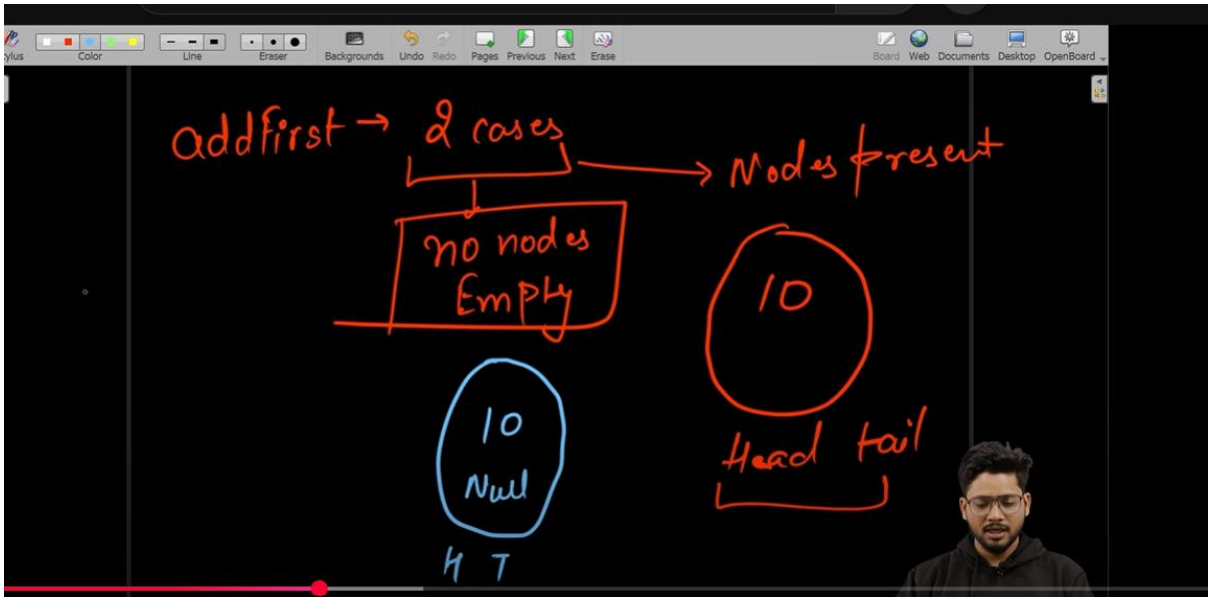
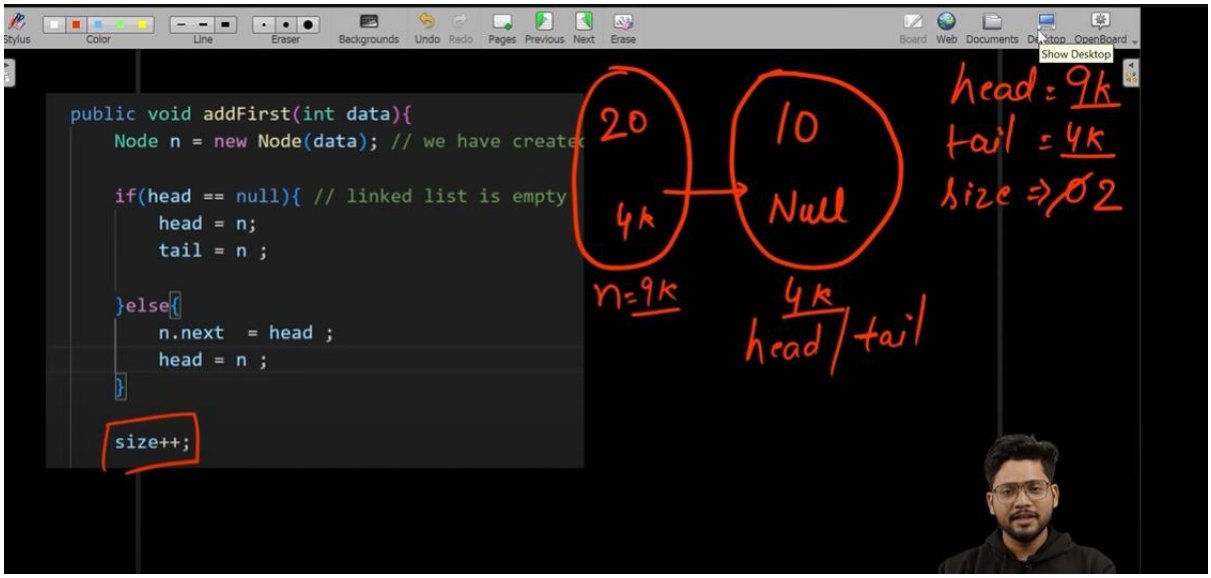
Linked List(gfg)

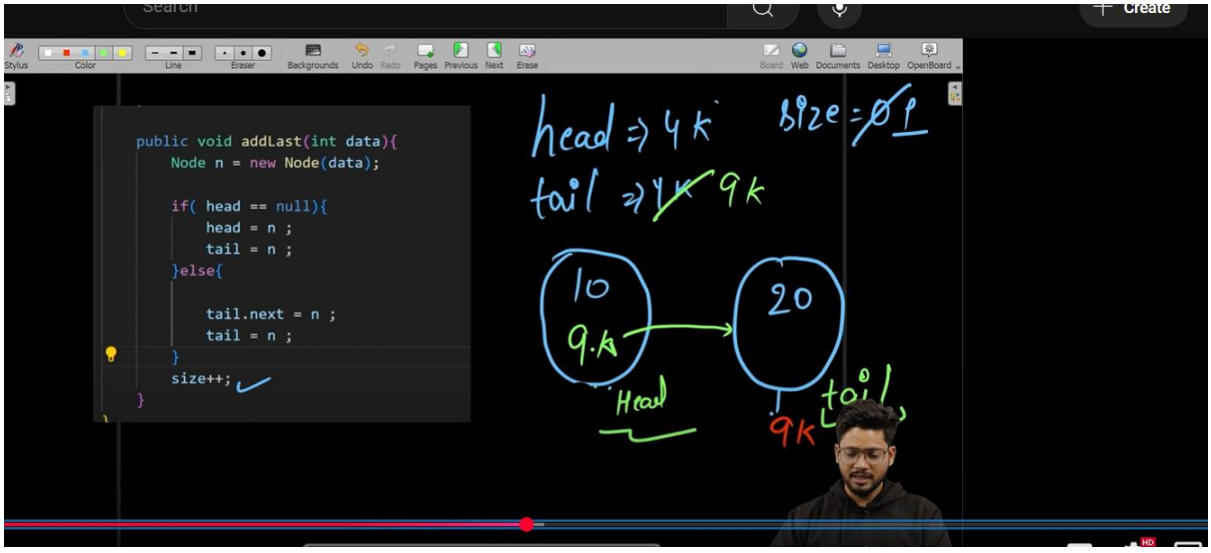







addFirst → 2 cases
Nodes present
no nodes
Empty
10
Null
H T
10
Head tail


public void addFirst(int data){
Node n = new Node(data); // we have create
if(head == null){ // linked list is empty
head = n;
tail = n ;
}else{
n.next = head ;
head = n ;
}
size++;
20
4k
10
Null
n=9k
4k
head/tail
head = 9k
tail = 4k
size ⇒ 02


public void addLast(int data){
Node n = new Node(data);
if(head == null){
head = n ;
tail = n ;
}else{
tail.next = n ;
tail = n ;
}
size++;
}
head ⇒ 4k
size = 01
tail ⇒ 9k
10
9.k
20
Head
9k
tail

Code 1 Code 1

```java
public class Lecture1 {
    public static void main(String[] args) {
        MylinkedList list = new MylinkedList();

        list.addFirst(10);
        list.addFirst(20);
        list.addLast(100);
        list.addLast(190);

        list.addAtSpecificIndex(110, 2);

        list.addAtSpecificIndex(99, 1);

        System.out.println(" before deleteion " + list);

        System.out.println(list.removeAtSpecificIndex(2));
        System.out.println(list.removeAtSpecificIndex(3));

        System.out.println("After deletion " + list);
    }
}

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

```java
public void addAtSpecificIndex(int data, int idx) {

    if (idx < 0 || idx > size) {
        System.out.println(" Index is not valid ");
        return;
    } else if (idx == 0) {
        addFirst(data);
    } else if (idx == size) {
        addLast(data);
    } else {
        Node n = new Node(data);

        Node pre = head;

        while (idx - 1 > 0) {
            pre = pre.next;
            idx--;
        }

        Node nbr = pre.next;

        pre.next = n;
        n.next = nbr;

        size++;
    }
}
```

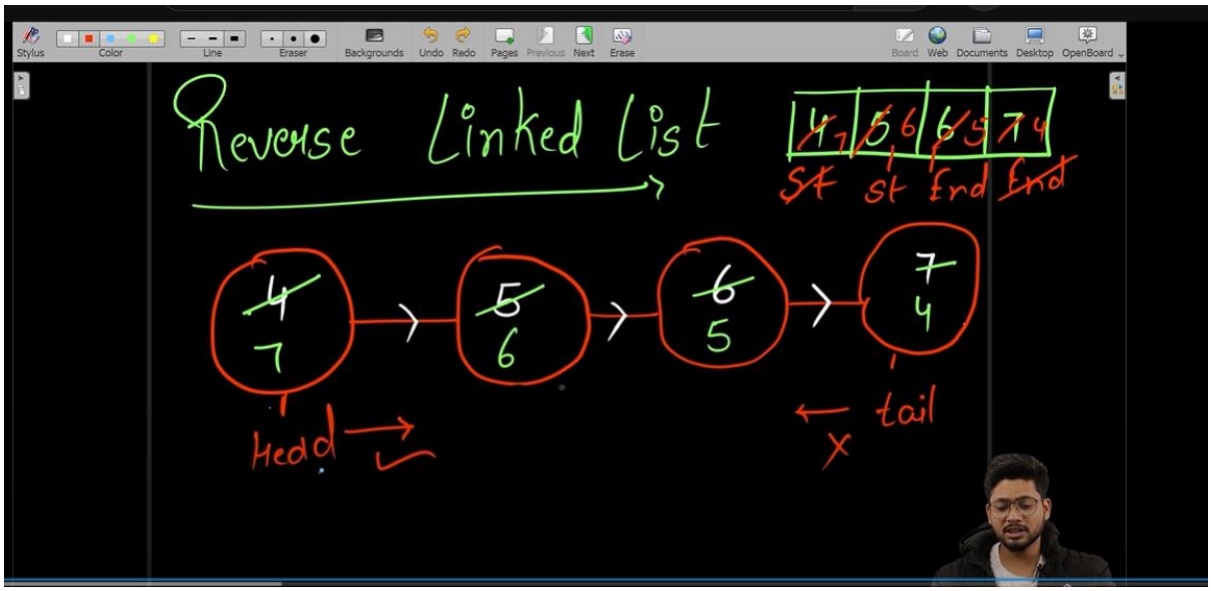
```java
}

public int getFirst() {

    if (head == null) {
        System.out.println("LinkedList is empty");
        return -1;
    } else {
        return head.data;
    }
}

public int getLast() {

    if (tail == null) {
        System.out.println("LinkedList is empty");
        return -1;
    } else {
        return tail.data;
    }
}

public int getAtSpecificIndex(int idx) {
    if (idx < 0 || idx >= size) {
```

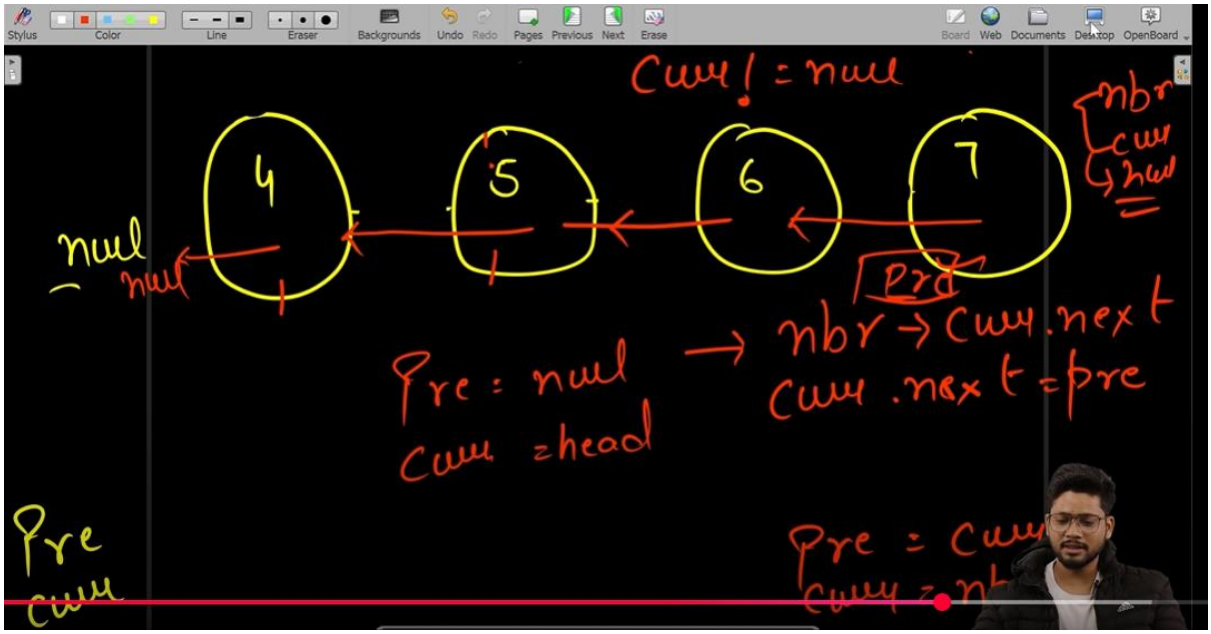
```java
public int getAtSpecificIndex(int idx) {
    if (idx < 0 || idx >= size) {
        System.out.println("Invalid index ");
        return -1;
    } else if (idx == 0) {
        return getFirst();
    } else if (idx == size - 1) {
        return getLast();
    } else {
        Node curr = head;

        while (idx > 0) {
            curr = curr.next;
            idx--;
        }

        return curr.data;
    }
}
```

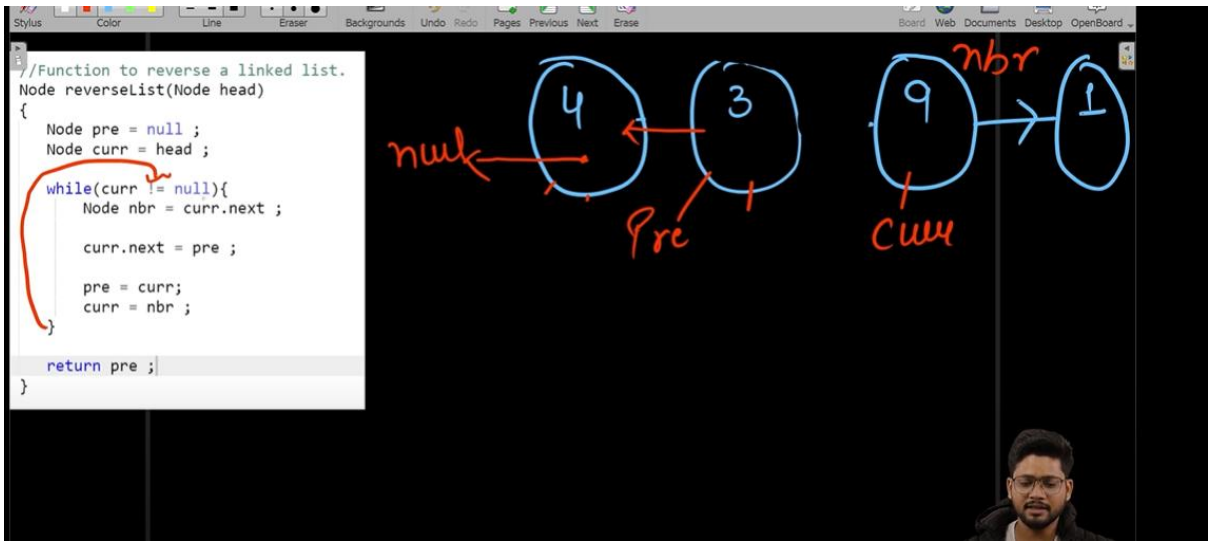
```java
public int removeFirst() {
    if (head == null) {
        System.out.println("LinkedList is empty");
        return -1;
    } else if (head.next == null) {

        int data = head.data;
        head = null;
        tail = null;

        size--;
        return data;
    } else {

        int data = head.data;

        head = head.next;

        size--;

        return data;
    }
}

public int removeLast() {
    if (head == null) { // empty LL
        System.out.println(" Linked list is empty ");
        return -1;
    } else if (head.next == null) { // one Node in LL
        int data = head.data;

        head = null;
        tail = null;
        size--;
        return data;
    } else {
        Node curr = head;

        int data = tail.data;

        while (curr.next != tail) {
            curr = curr.next;
        }

        curr.next = null;

        size--;

        tail = curr;

        return data;
    }
}
```

```java

    public int removeAtSpecificIndex(int idx){

        if ( idx<0 || idx>= size ){
            System.out.println("Invalid idx");
            return -1 ;
        }else if ( idx == 0 ){
            return removeFirst();
        }else if ( idx == size-1 ){
            return removeLast();
        }else {
            Node curr = head ;

            while( idx-1 > 0){
                curr = curr.next ;

                idx-- ;
            }
            int data = curr.next.data ;
            curr.next = curr.next.next ;

            size-- ;

            return data ;
        }
    }

    public String toString() {

        String str = "";

        Node curr = head;

        while (curr != null) {

            str = str + curr.data + " ";

            curr = curr.next;
        }

        return str;
    }

    public int length() {
        return size:
```

1. ***Reverse LinkedList***

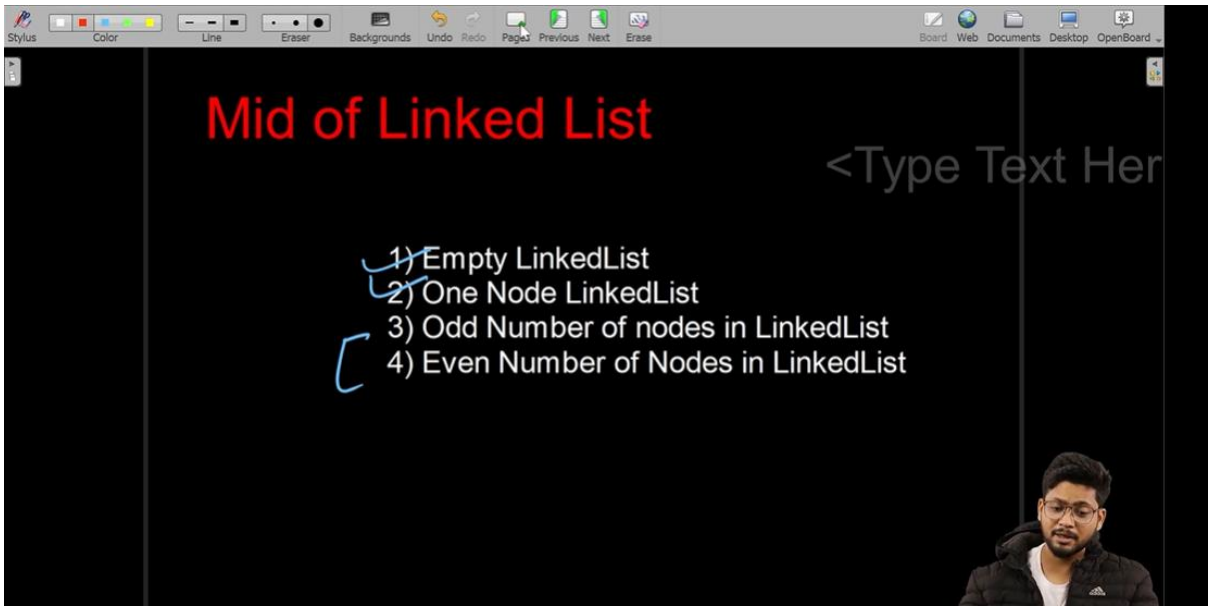


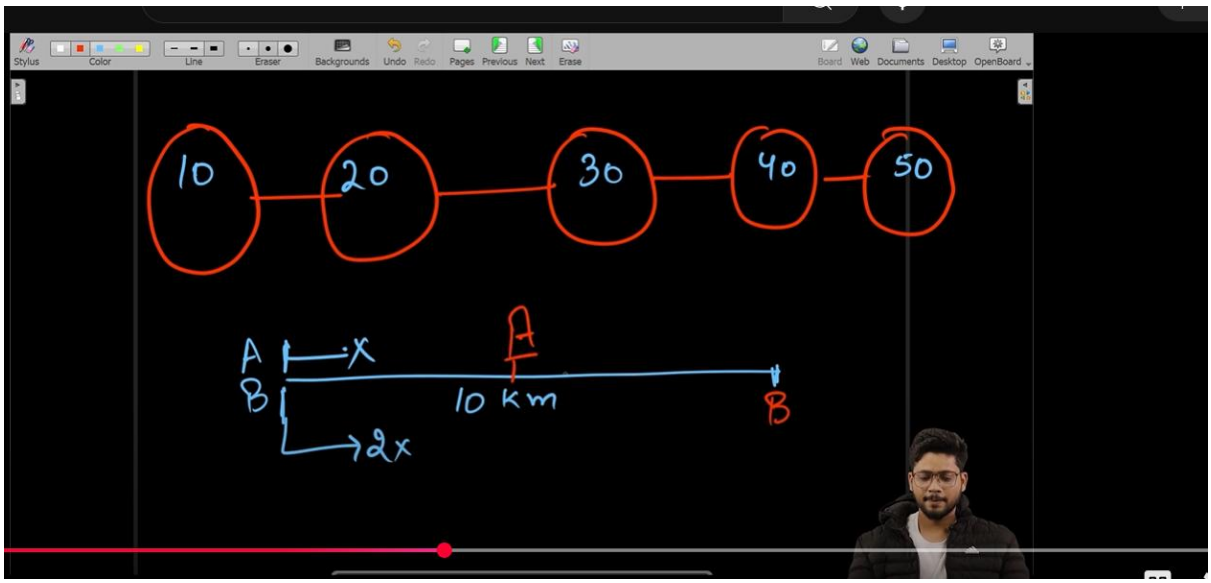


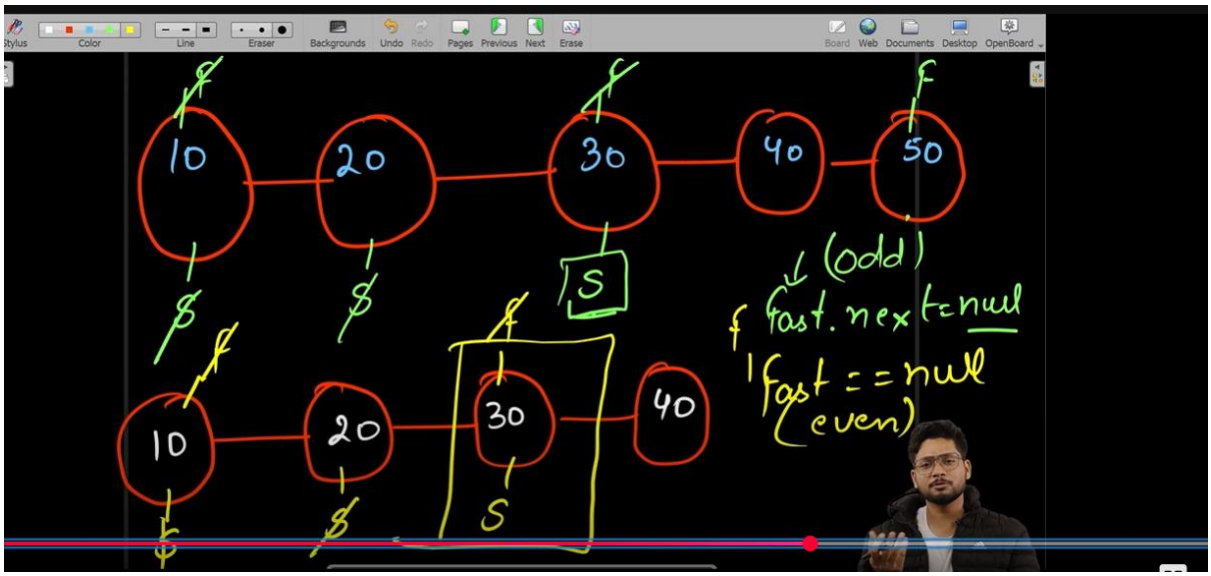


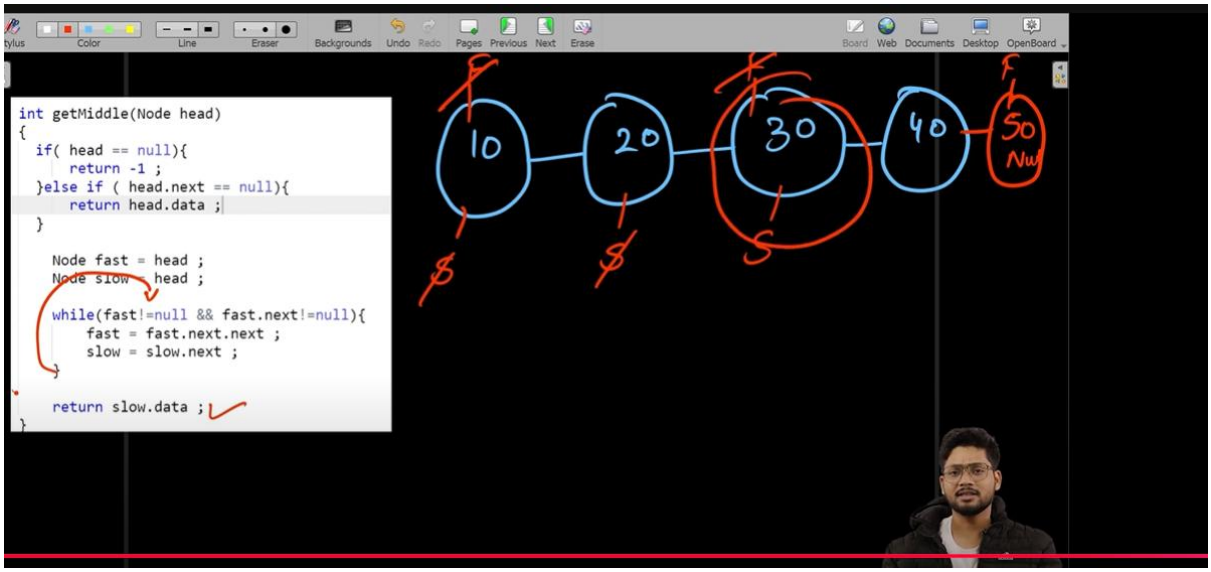
```
//Function to reverse a linked list.
Node reverseList(Node head)
{
    Node pre = null ;
    Node curr = head ;

    while(curr != null){
        Node nbr = curr.next ;

        curr.next = pre ;

        pre = curr;
        curr = nbr ;
    }

    return pre ;
}
```

2.Middle Element in LinkedList

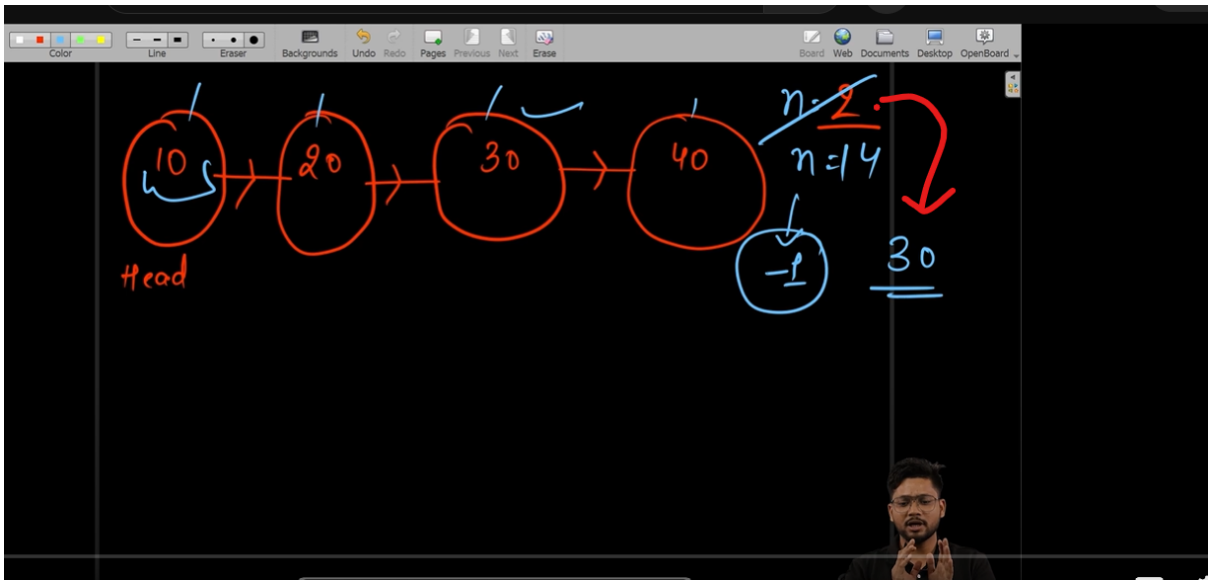


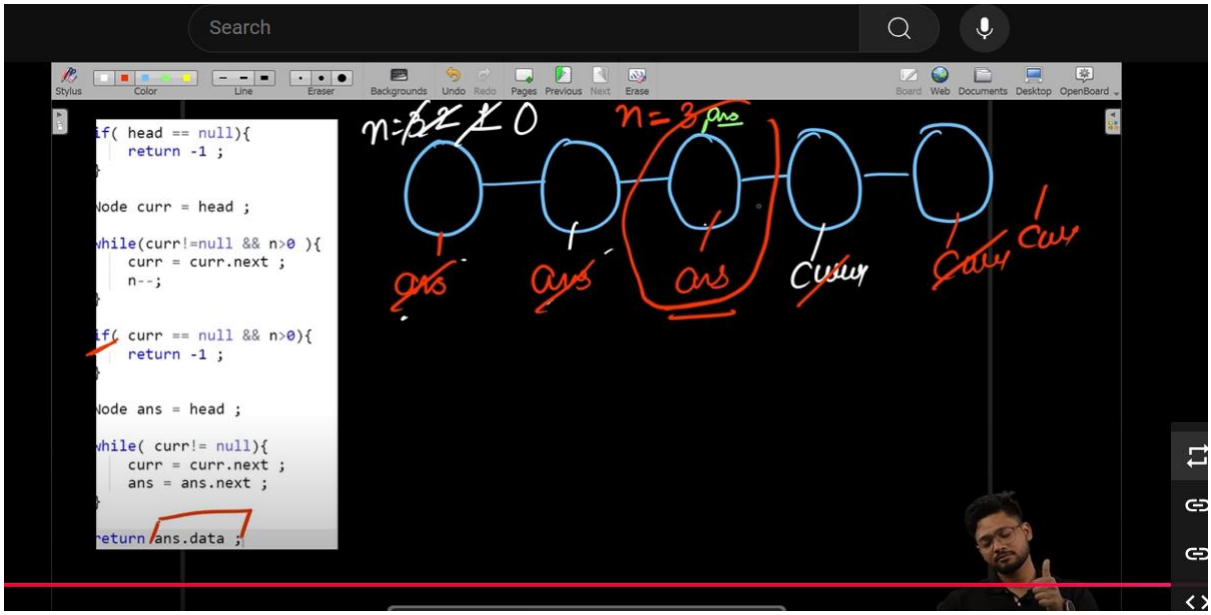




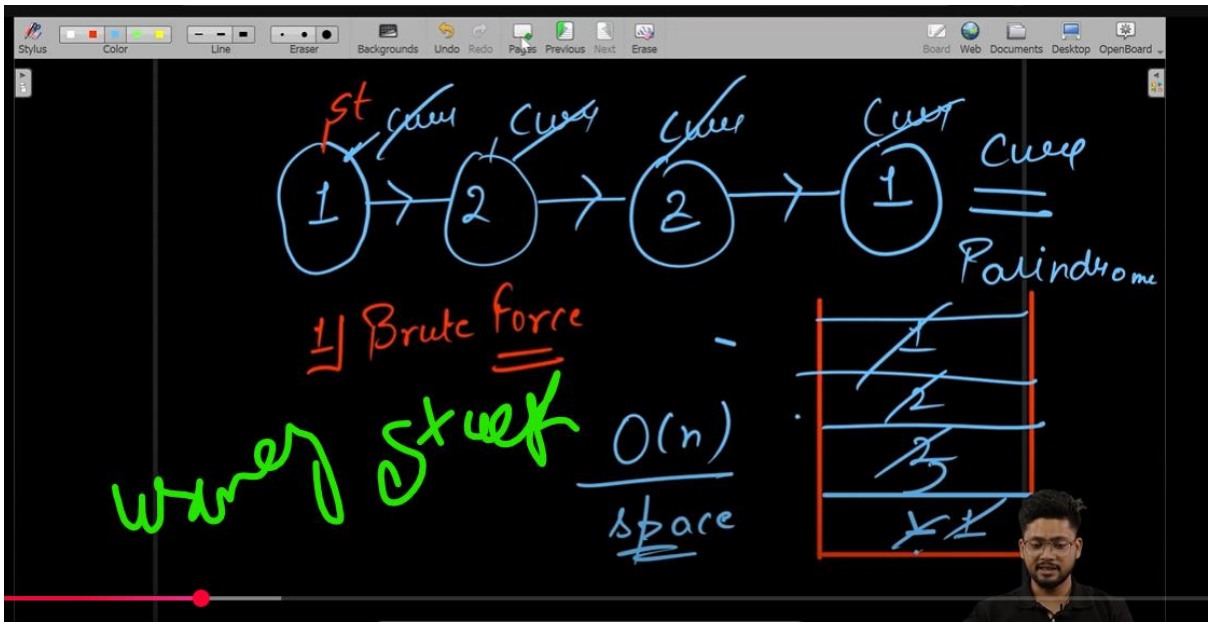


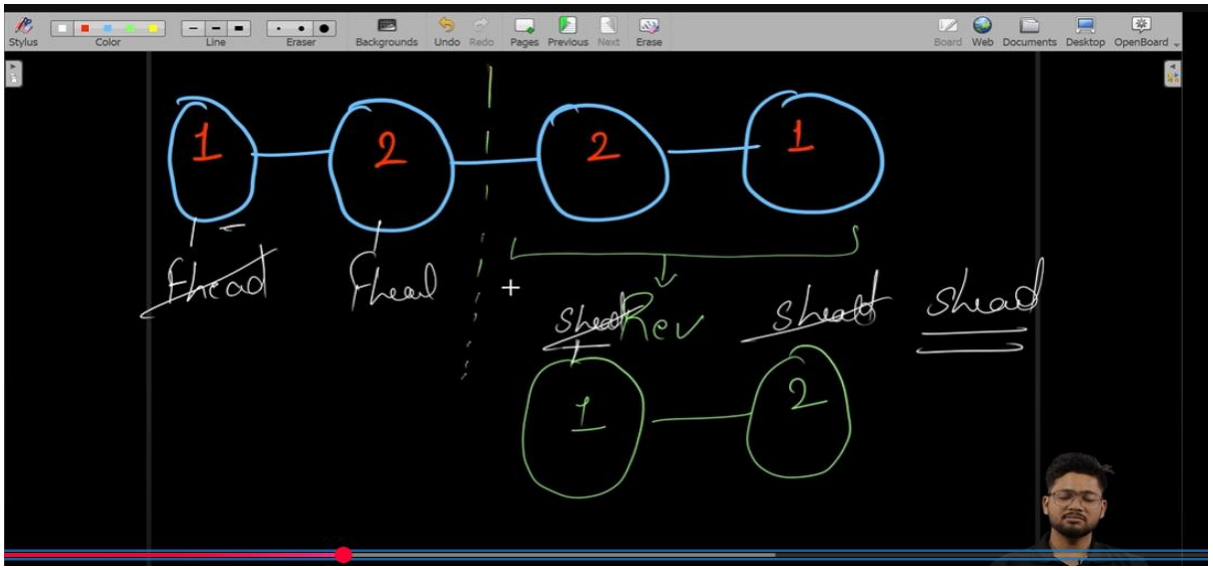
3.Nth Node from End of LinkedList

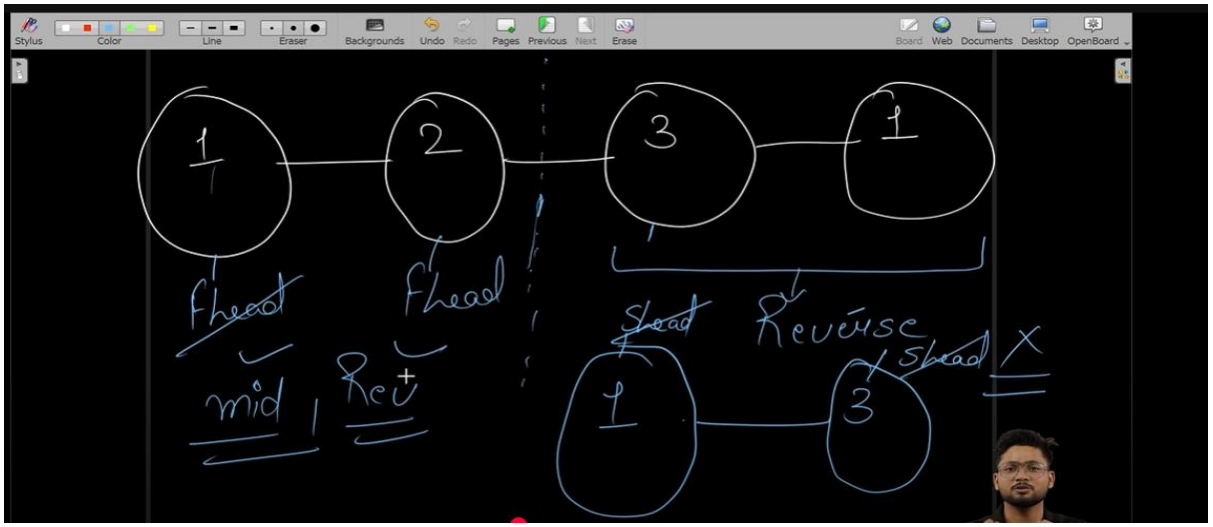


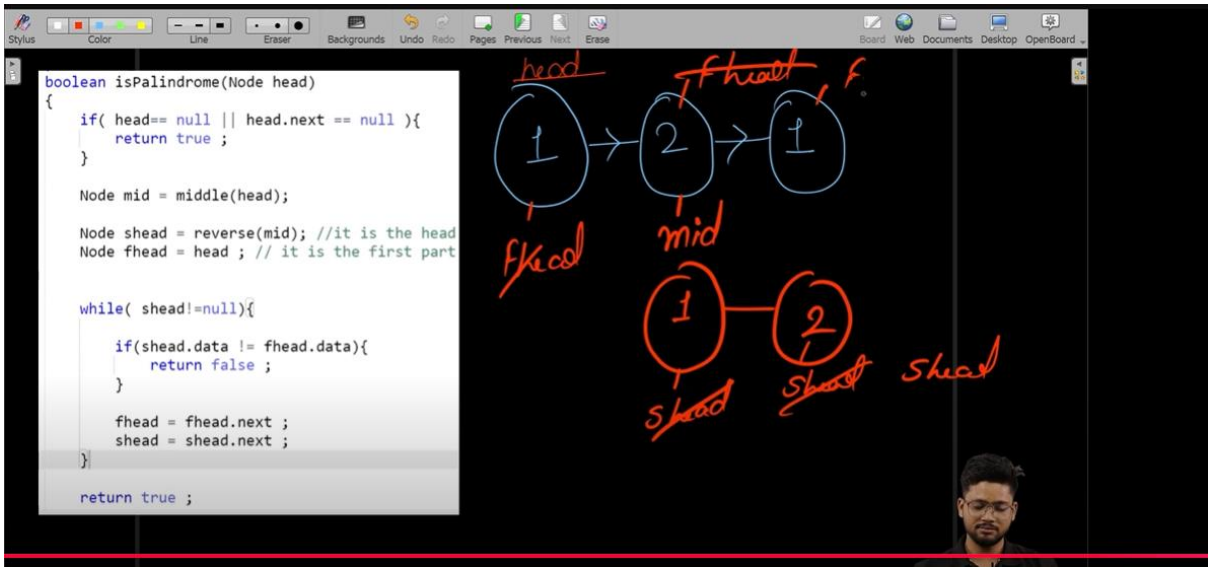


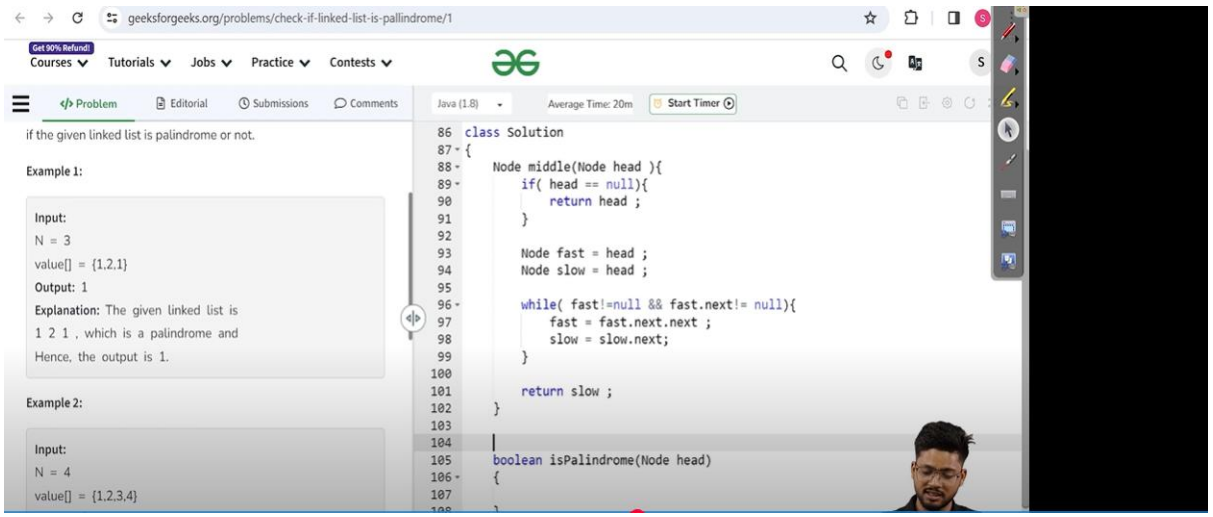
4. Check if linked list is palindrome

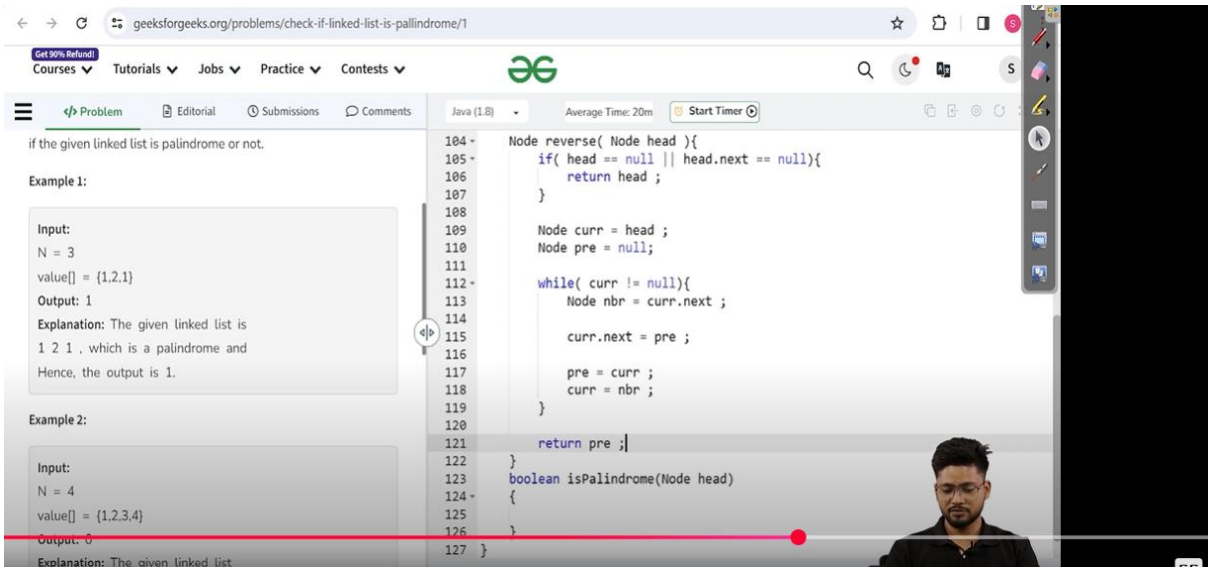




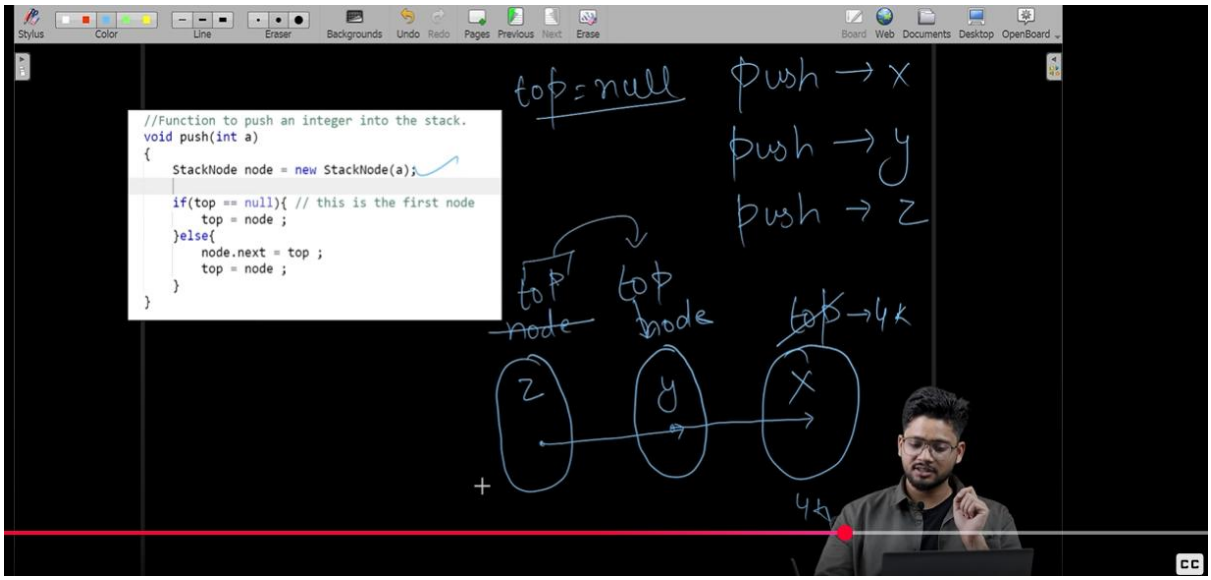


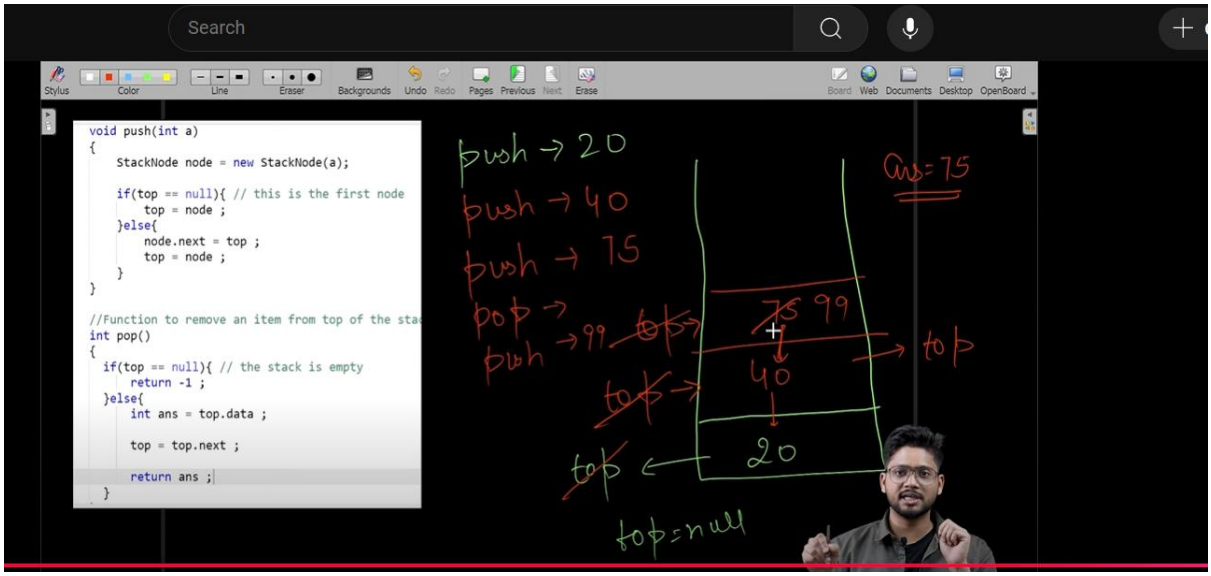

Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Desktop
OpenBoard
boolean isPalindrome(Node head)
{
if(head== null || head.next == null){
return true ;
}
Node mid = middle(head);
Node shead = reverse(mid); //it is the head
Node fhead = head ; // it is the first part
while(shead!=null){
if(shead.data != fhead.data){
return false ;
}
fhead = fhead.next ;
shead = shead.next ;
}
return true ;
head
fhead
mid
shead
shead
shead


geeksforgeeks.org/problems/check-if-linked-list-is-pallindrome/1
Get 90% Refund!
Courses
Tutorials
Jobs
Practice
Contests
Problem
Editorial
Submissions
Comments
Java (1.8)
Average Time: 20m
Start Timer
if the given linked list is palindrome or not.
Example 1:
Input:
N = 3
value[] = {1,2,1}
Output: 1
Explanation: The given linked list is
1 2 1 , which is a palindrome and
Hence, the output is 1.
Example 2:
Input:
N = 4
value[] = {1,2,3,4}
class Solution
{
Node middle(Node head){
if(head == null){
return head ;
}
Node fast = head ;
Node slow = head ;
while(fast!=null && fast.next!= null){
fast = fast.next.next ;
slow = slow.next;
}
return slow ;
}
boolean isPalindrome(Node head)
{

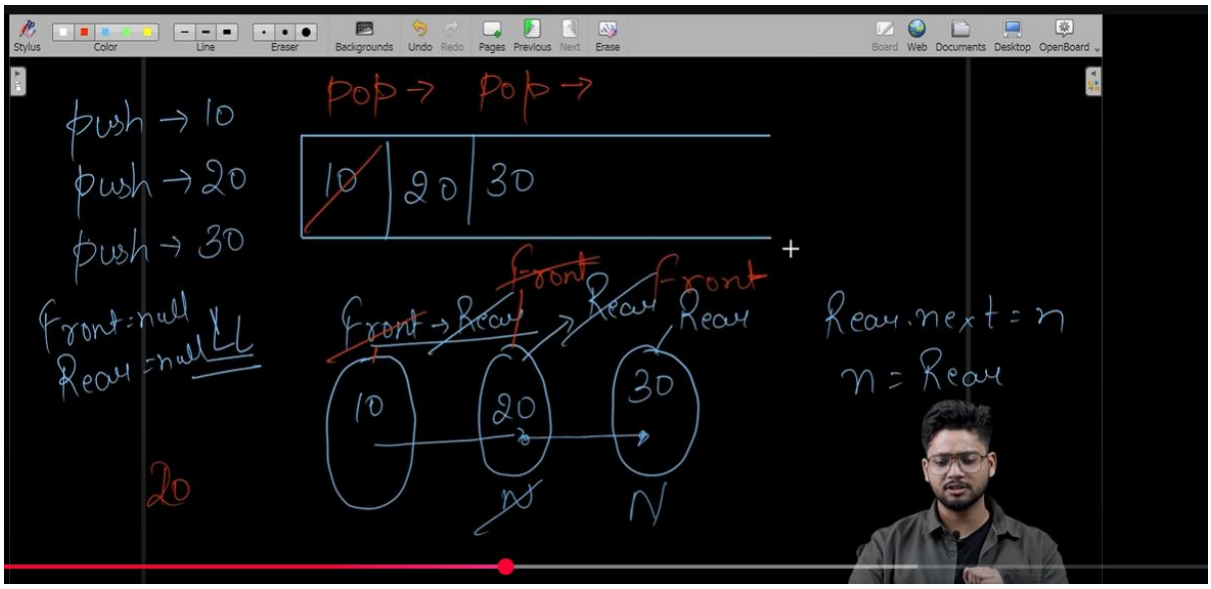

geeksforgeeks.org/problems/check-if-linked-list-is-pallindrome/1
Get 90% Refund!
Courses
Tutorials
Jobs
Practice
Contests
Problem
Editorial
Submissions
Comments
Java (1.8)
Average Time: 20m
Start Timer
if the given linked list is palindrome or not.
Example 1:
Input:
N = 3
value[] = {1,2,1}
Output: 1
Explanation: The given linked list is
1 2 1 , which is a palindrome and
Hence, the output is 1.
Example 2:
Input:
N = 4
value[] = {1,2,3,4}
Node reverse(Node head){
if(head == null || head.next == null){
return head ;
}
Node curr = head ;
Node pre = null;
while(curr != null){
Node nbr = curr.next ;
curr.next = pre ;
pre = curr ;
curr = nbr ;
}
return pre ;
}
boolean isPalindrome(Node head)
{
}
}

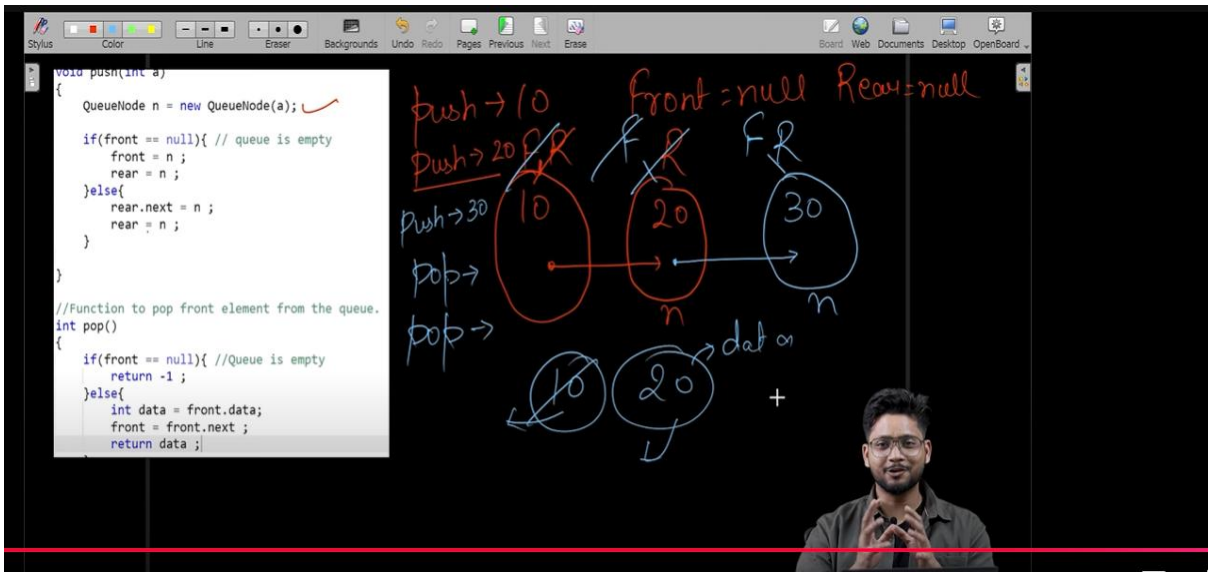
5. Implimenet stack using Linked list



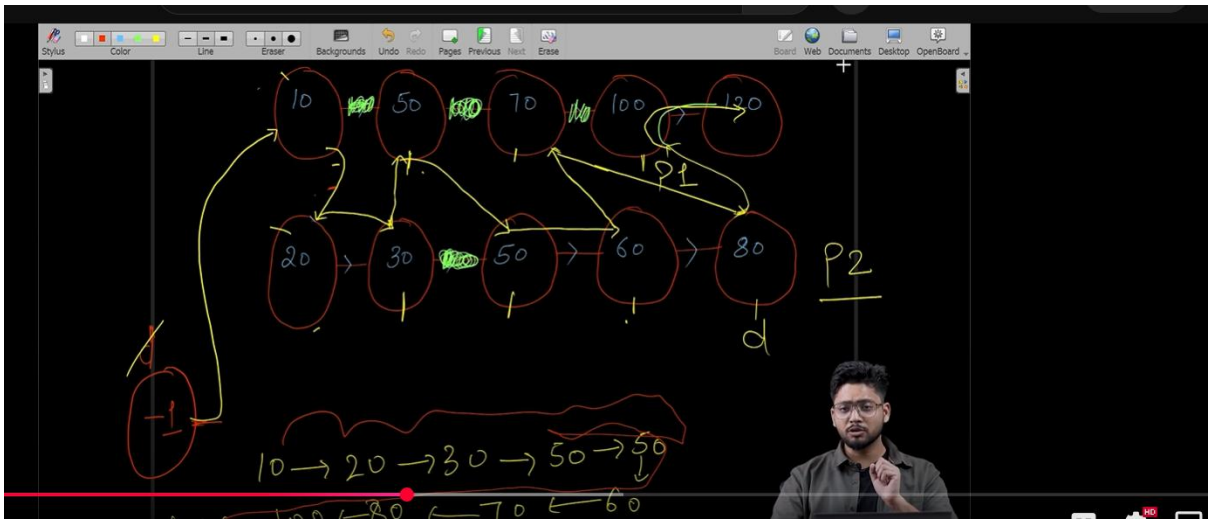


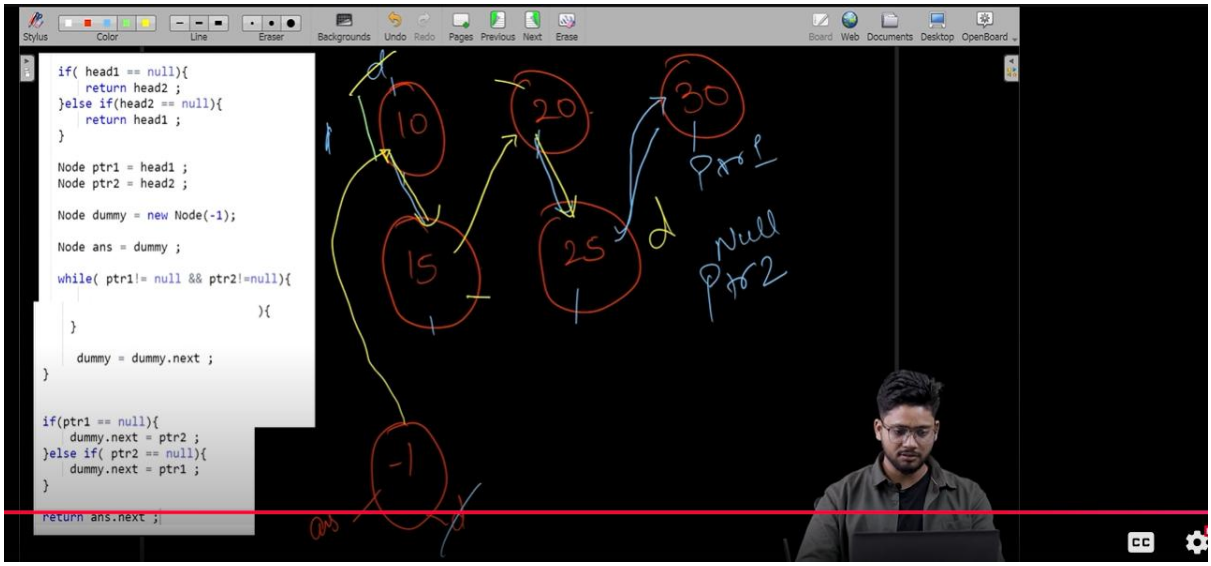
6.Implimenet Queue using LinkedList



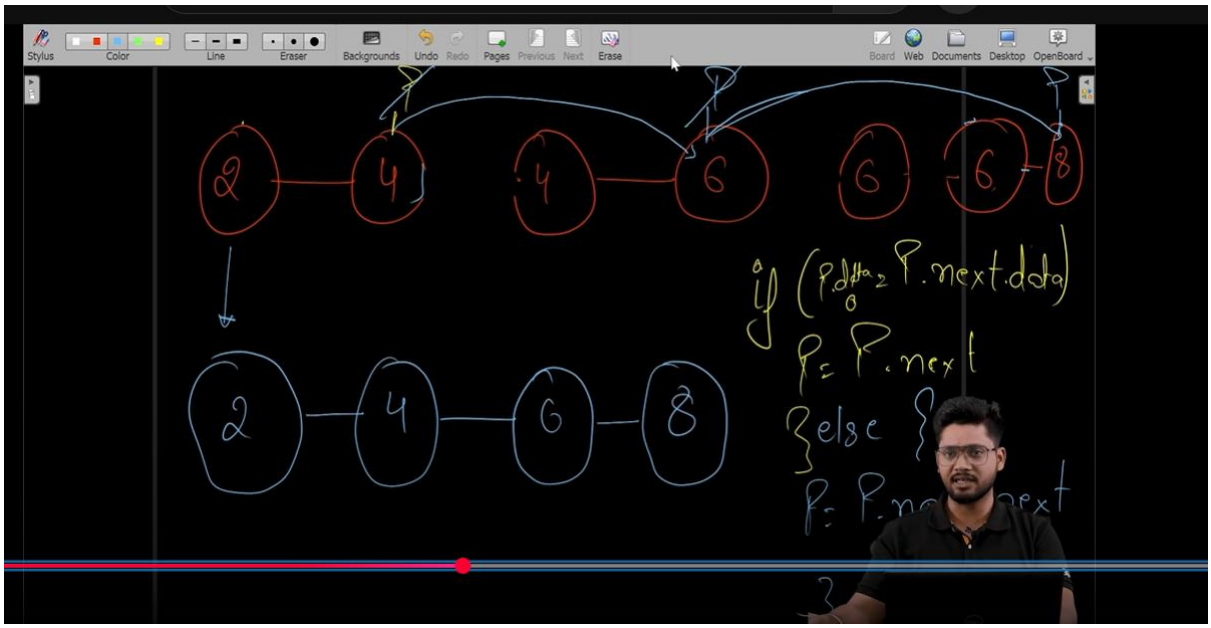


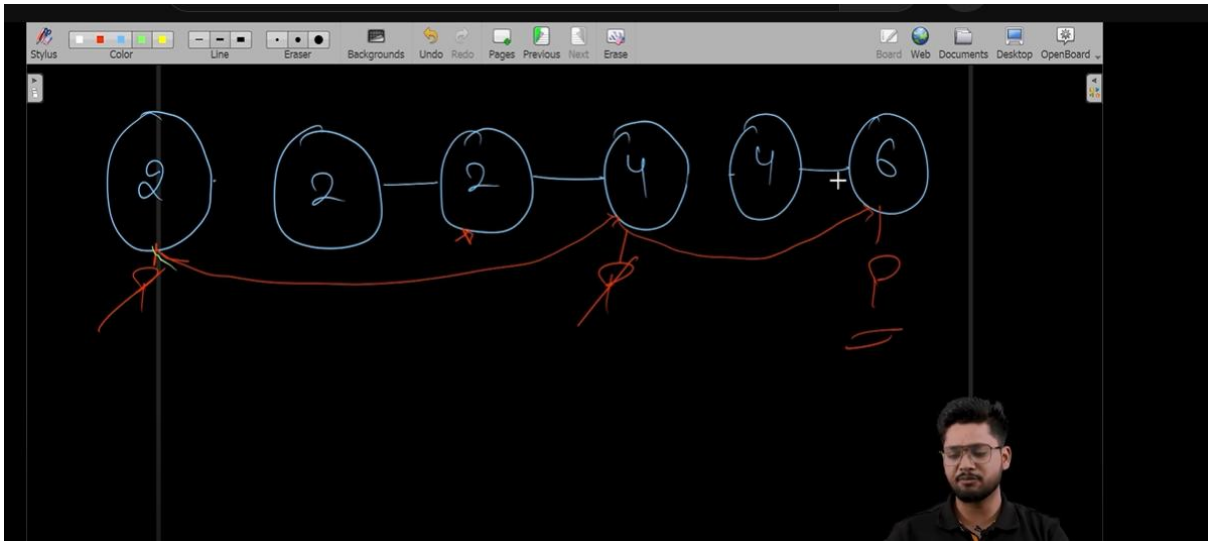
7.merge Two Sorted LinkedList

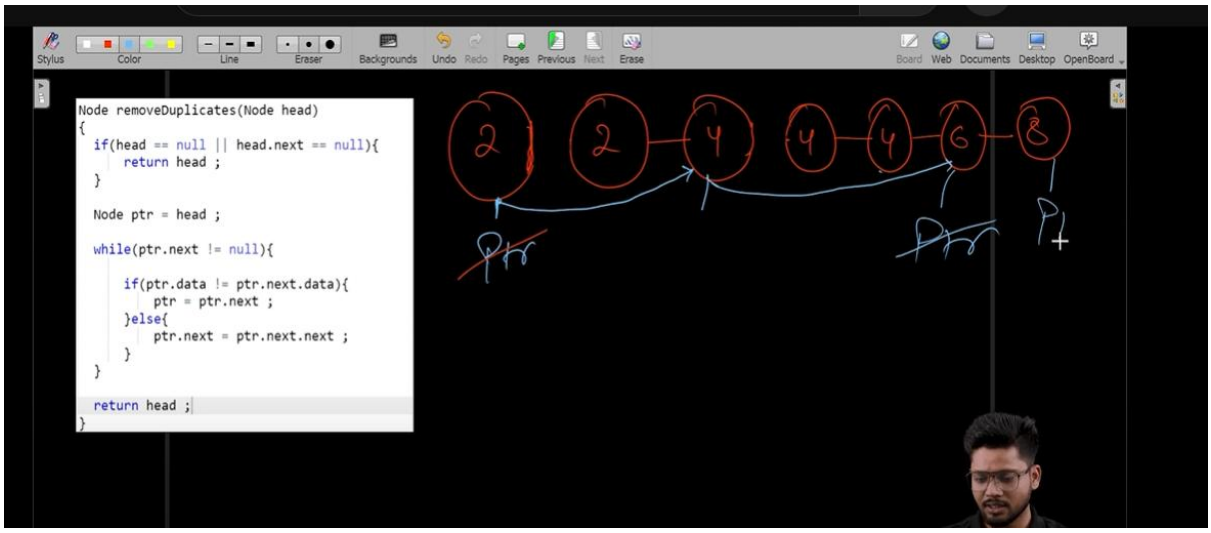




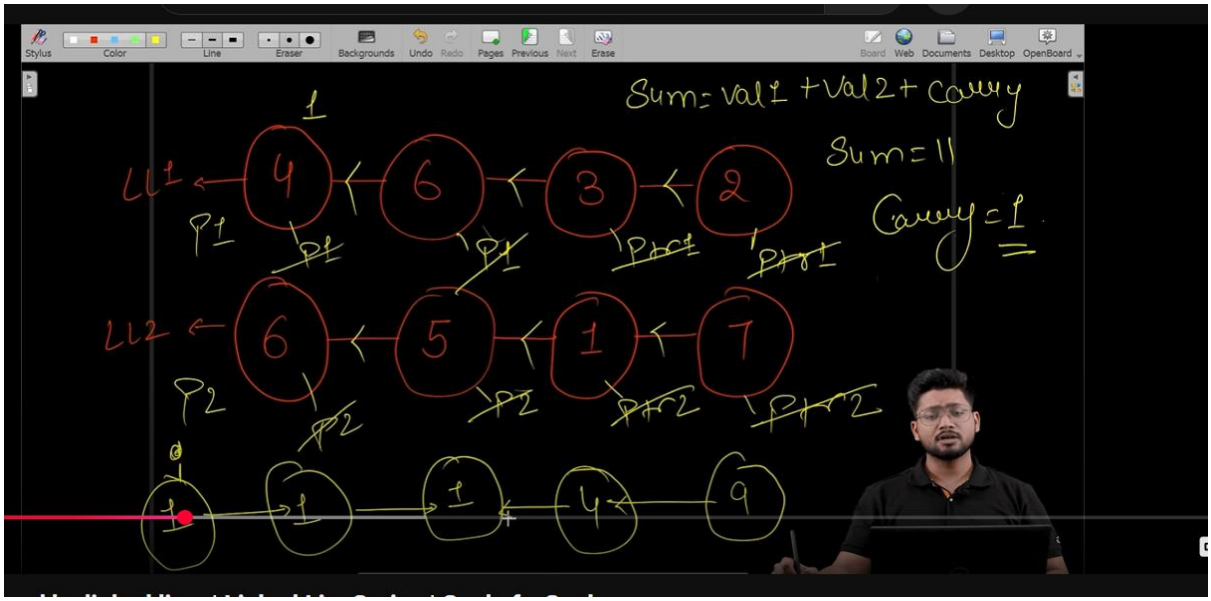
8.Remove duplicate element from sorted linked list

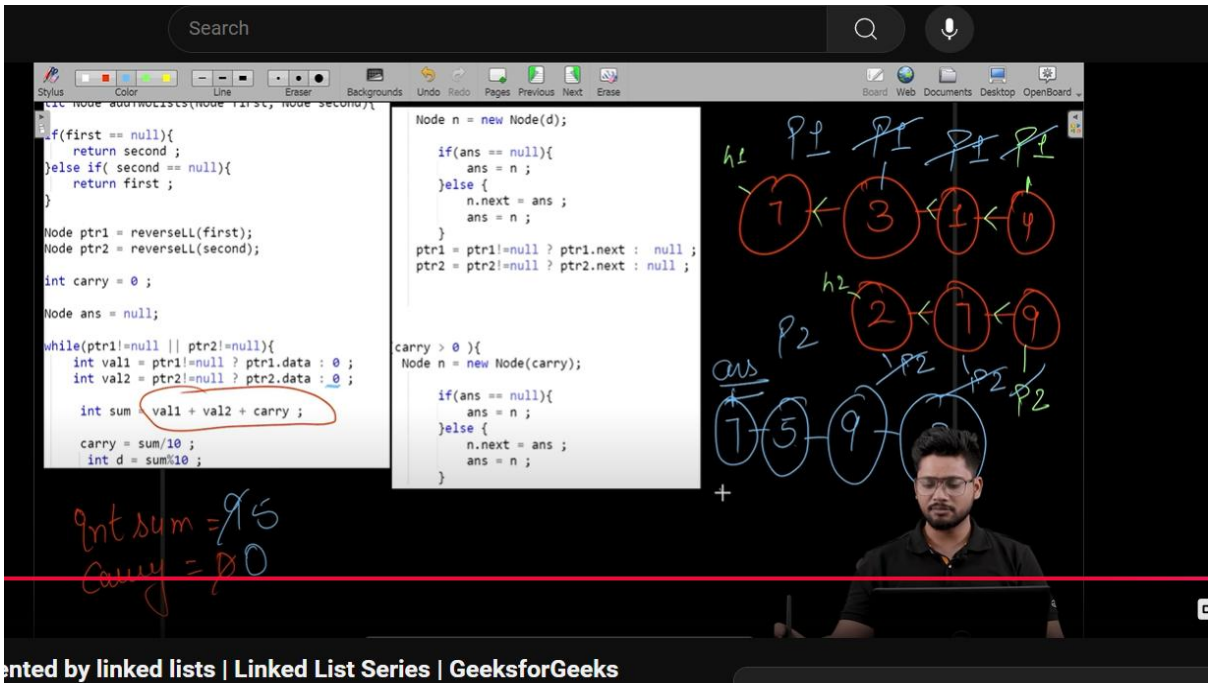


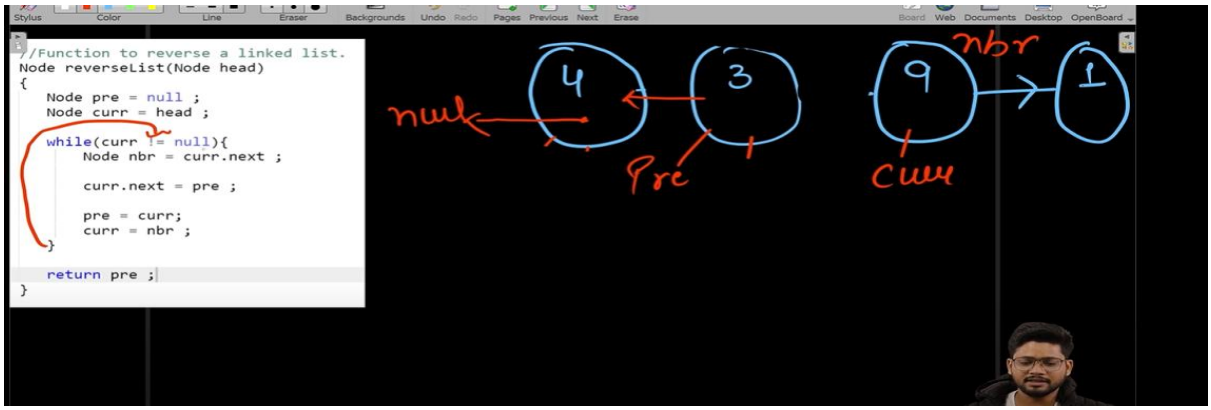




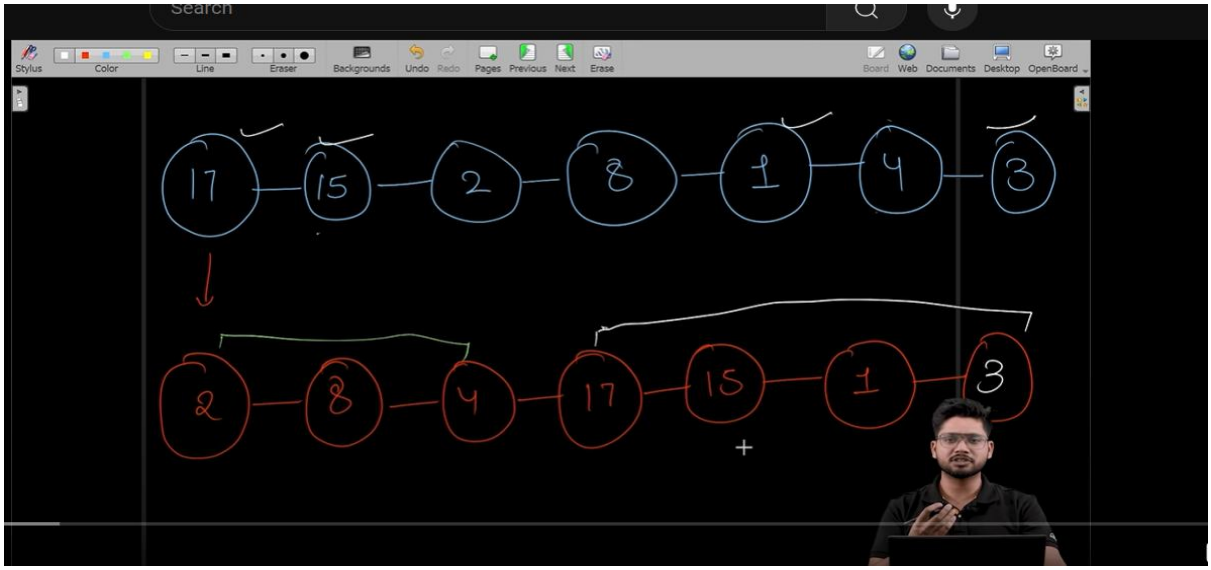
9.Add Two Number represented by Linked List

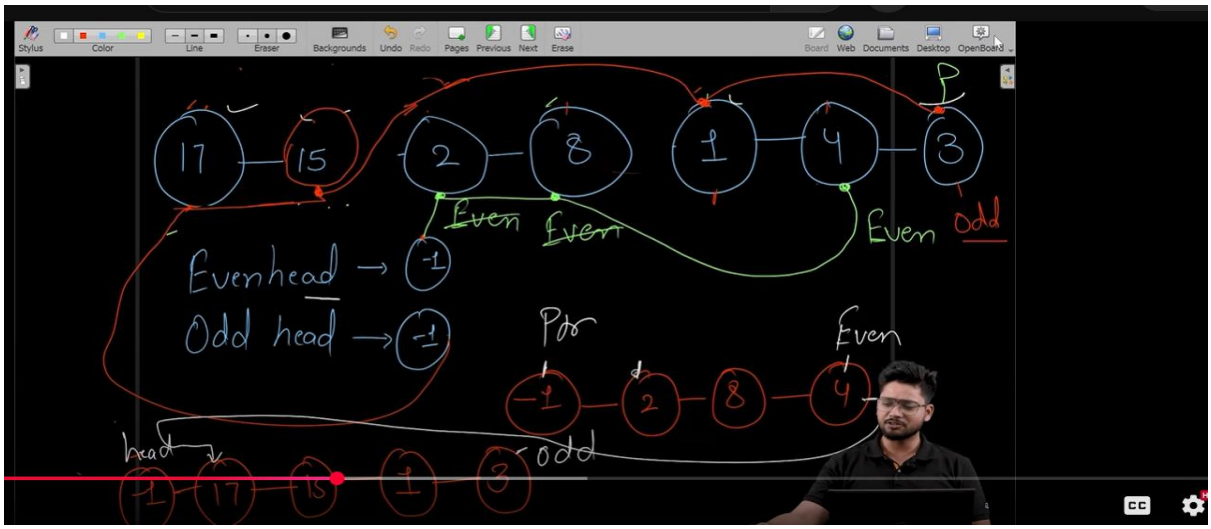


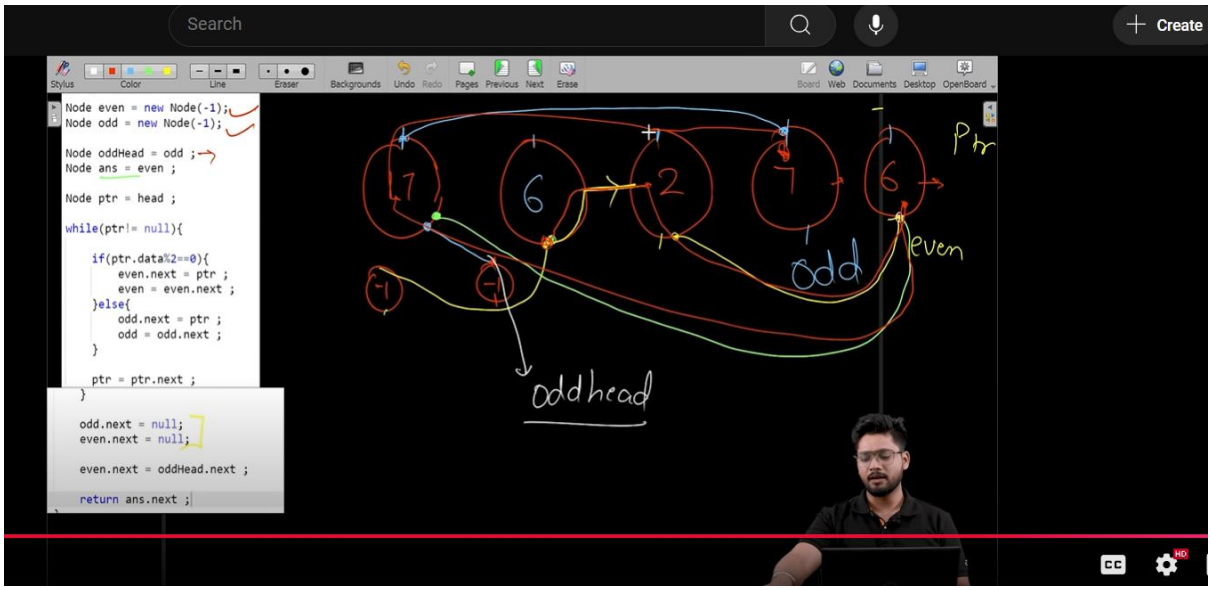




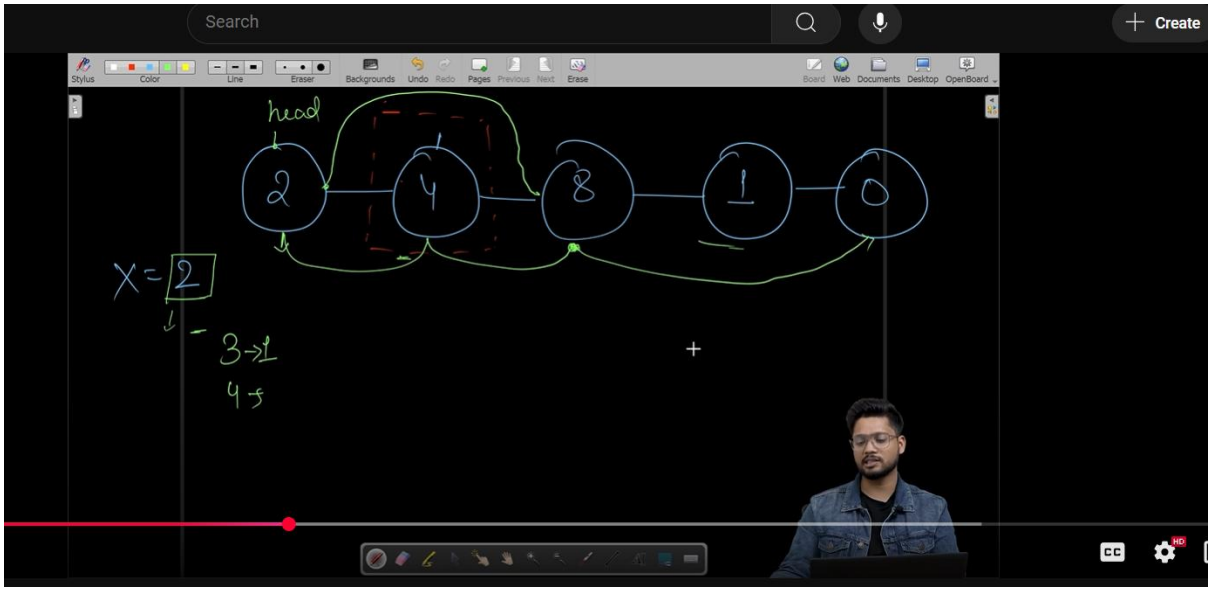
10.Segregate even ans odd in Linked List

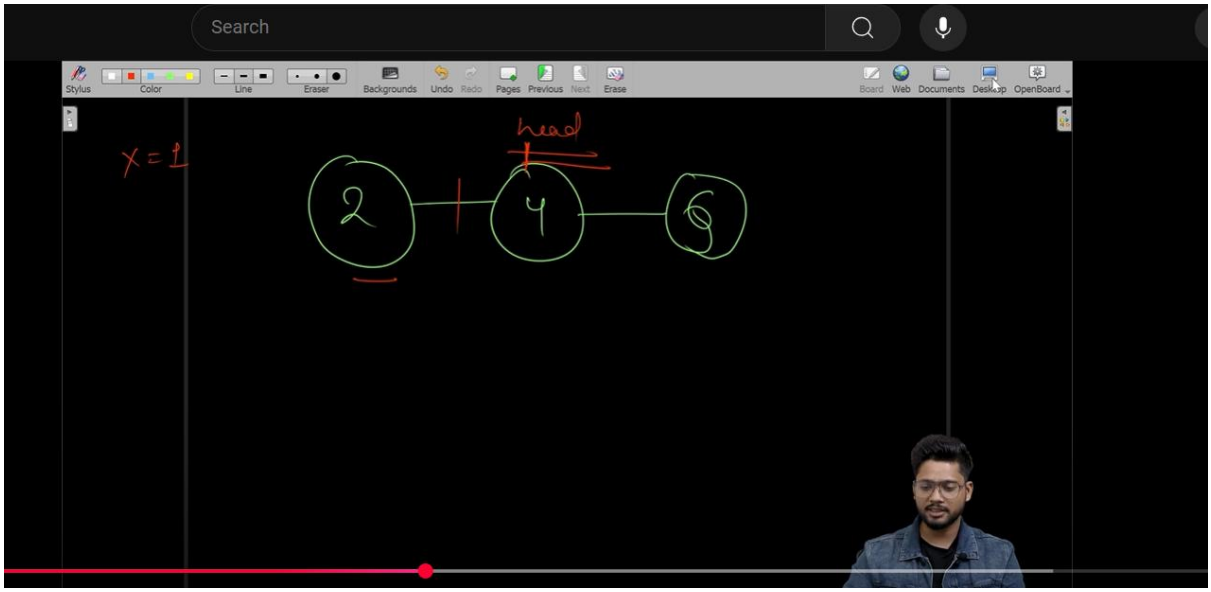


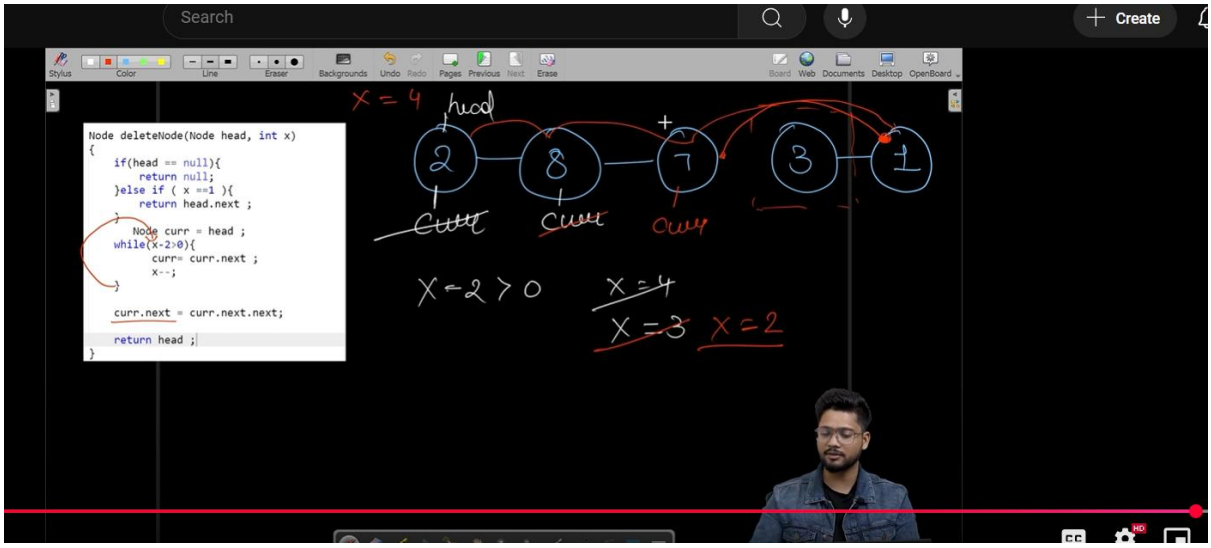




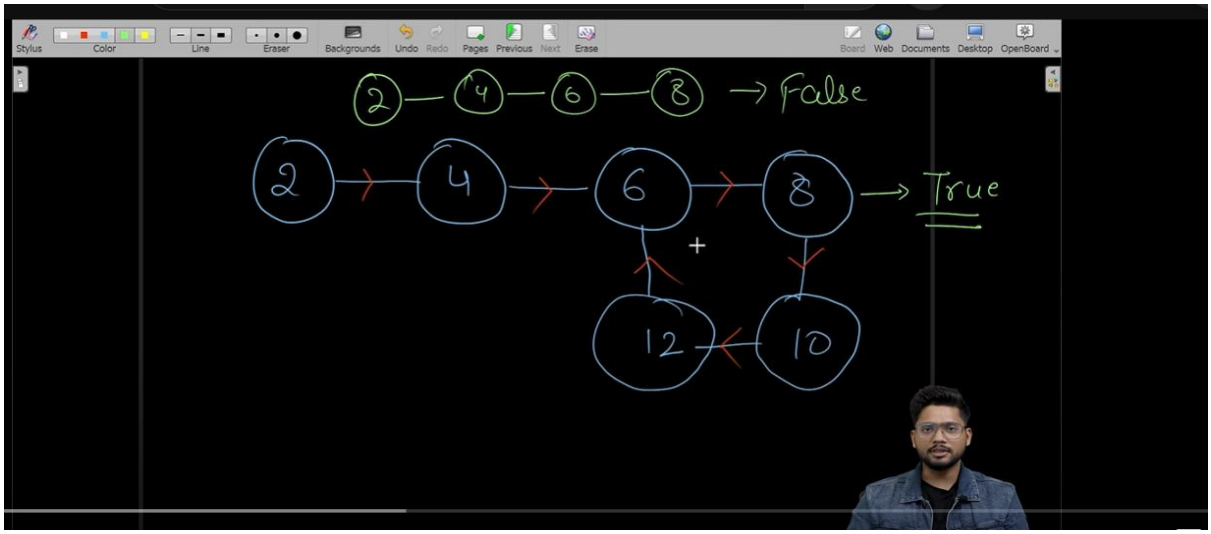
11.Delete a node in single linked list

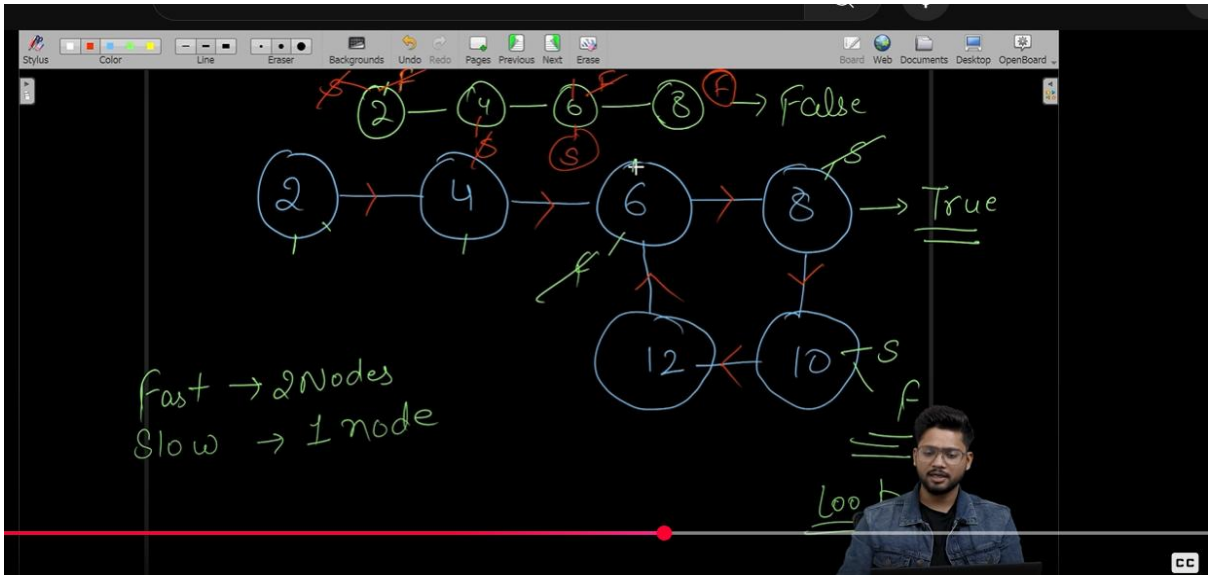


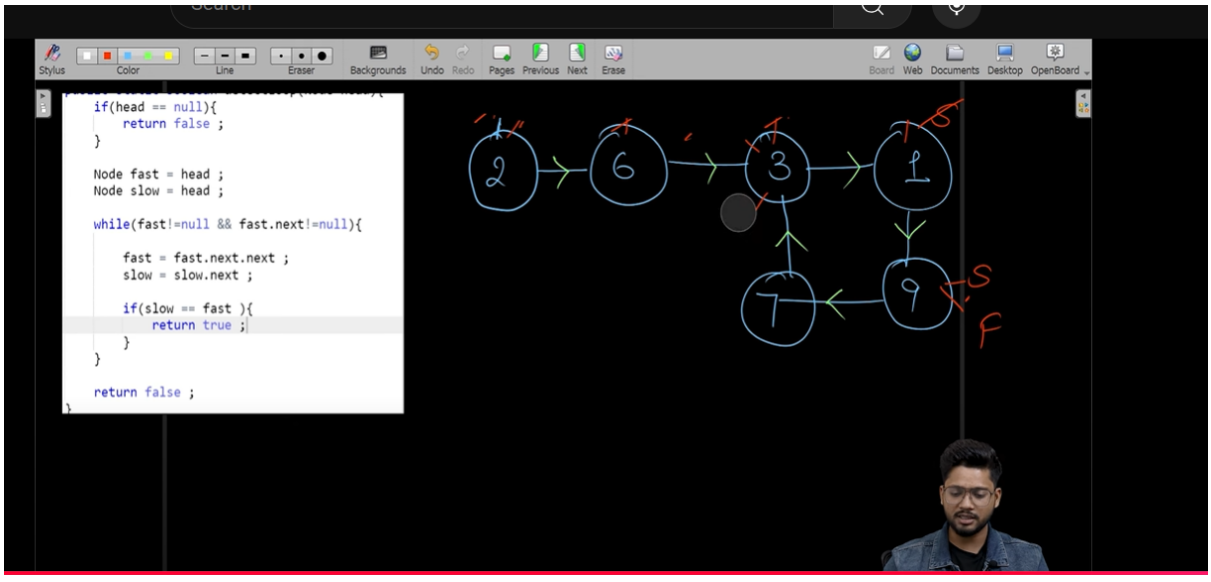




12. Detect Loop in Linked List

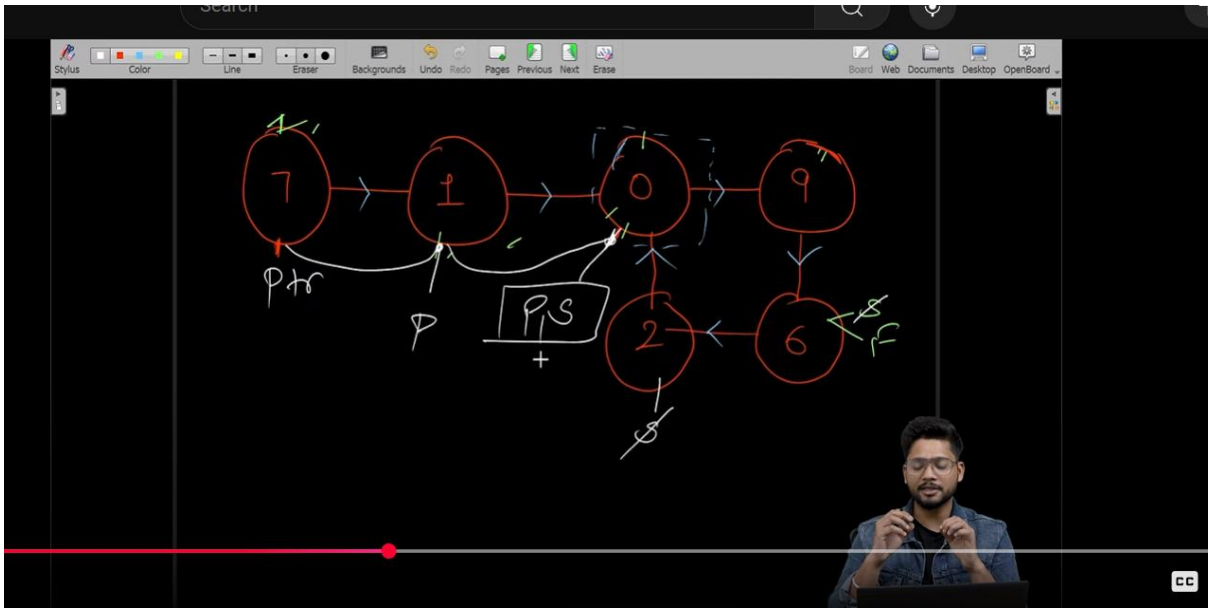


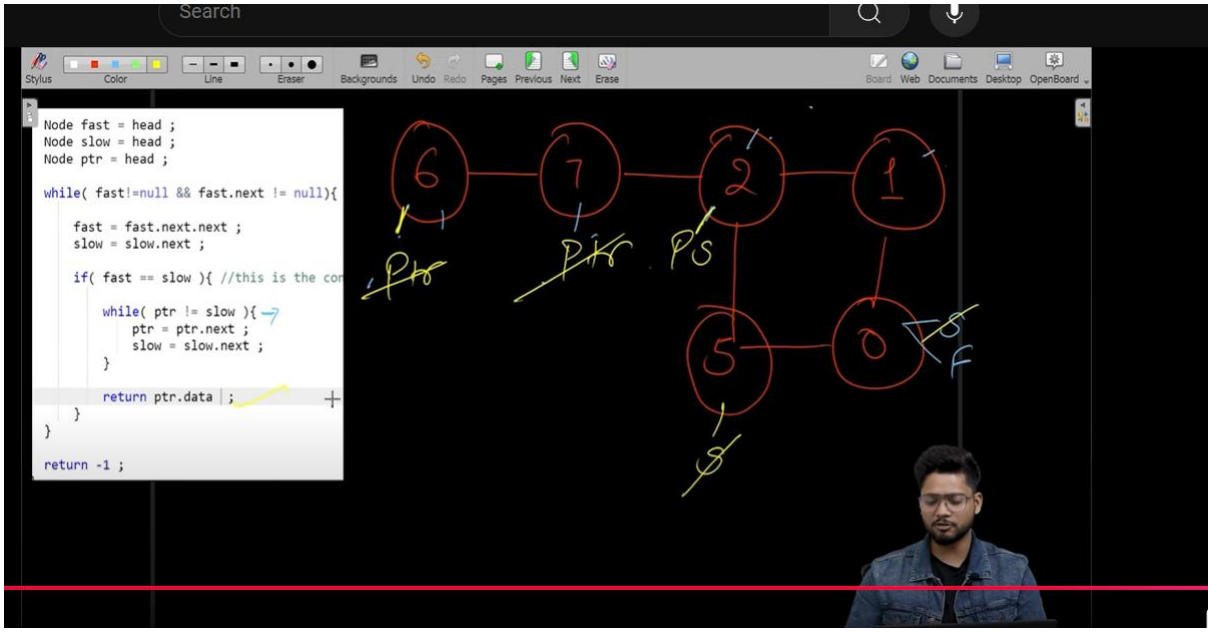




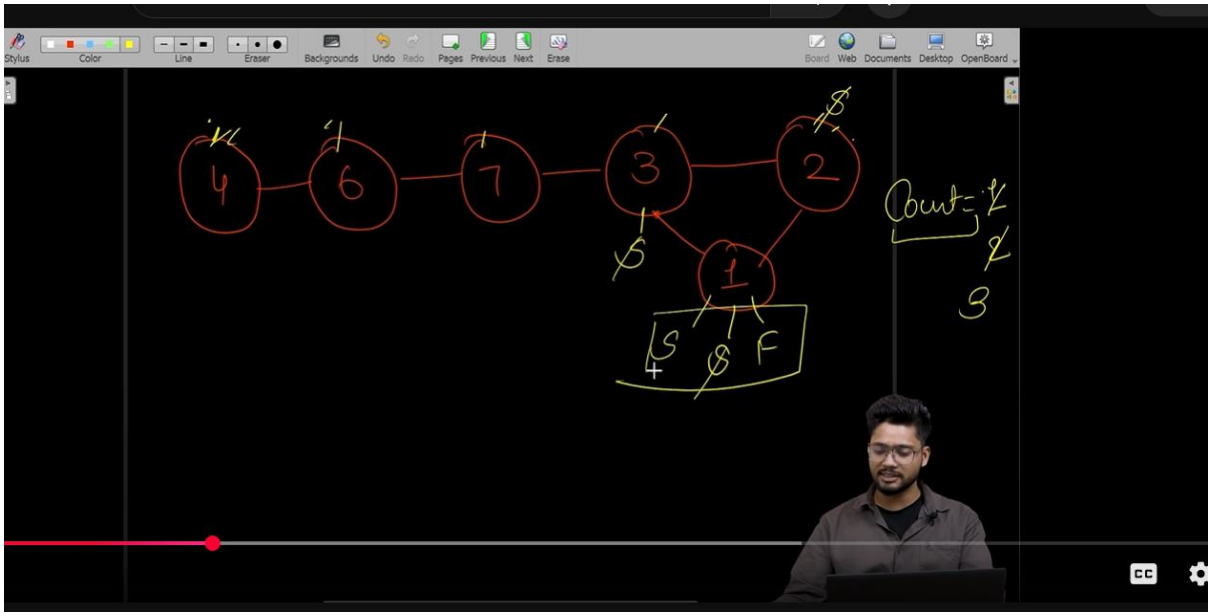
```
if(head == null){
    return false ;
}

Node fast = head ;
Node slow = head ;

while(fast!=null && fast.next!=null){

    fast = fast.next.next ;
    slow = slow.next ;

    if(slow == fast ){
        return true ;
    }
}

return false ;
```

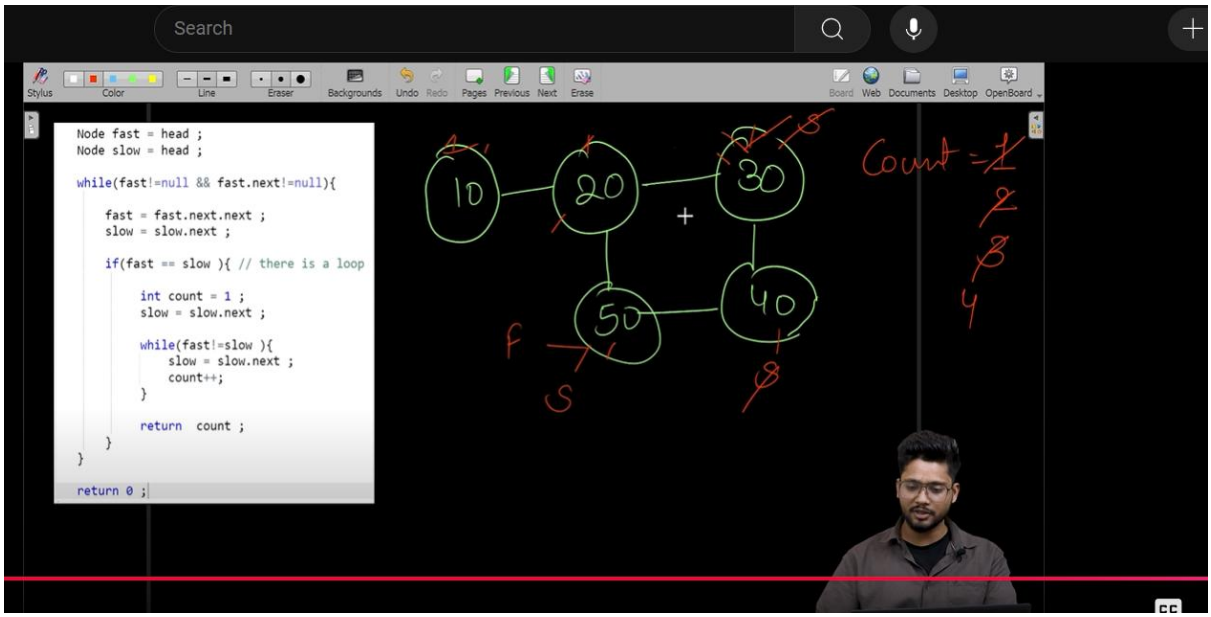
13.find the first node of loop in linked list



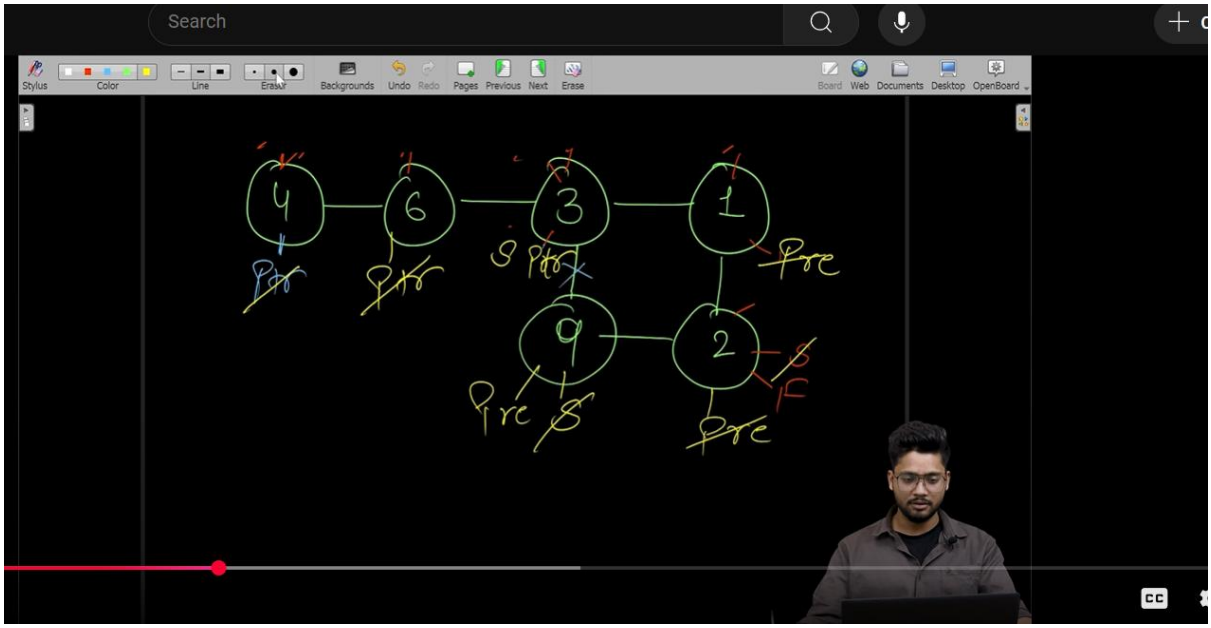


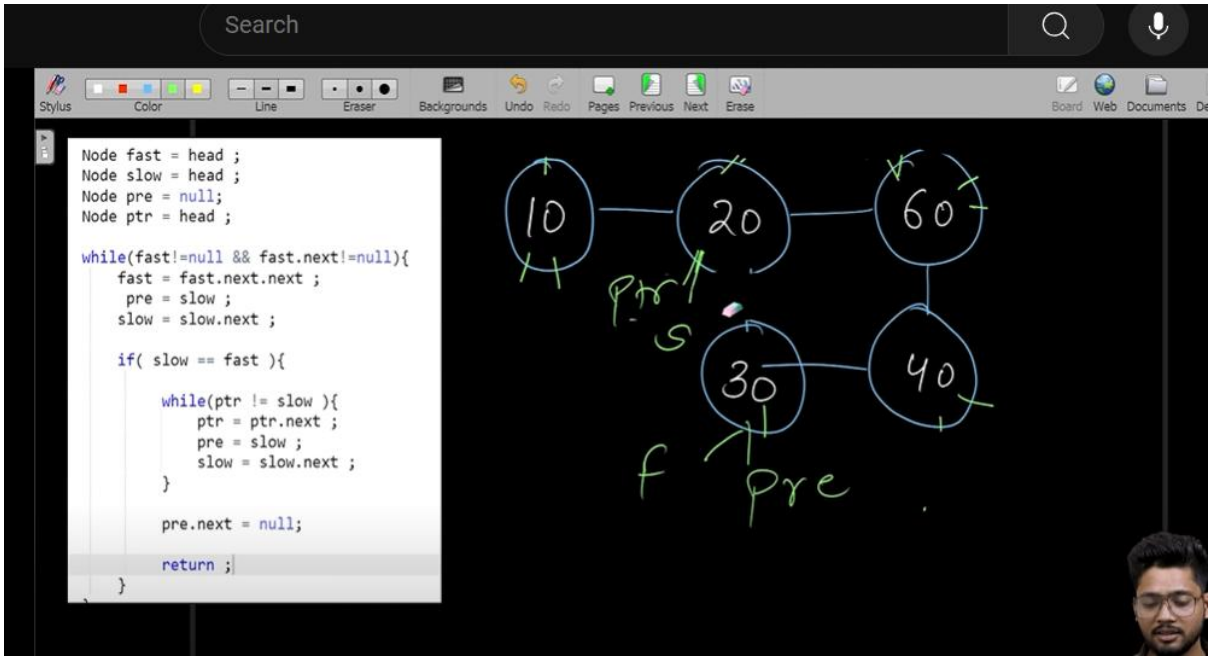
14 .Find the Length of the Loop in Linked List



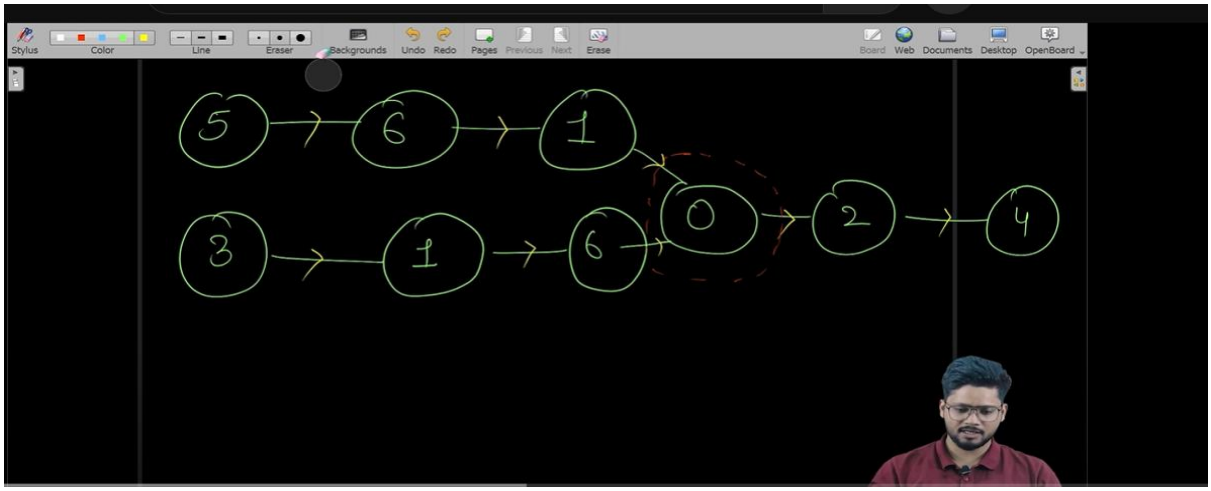


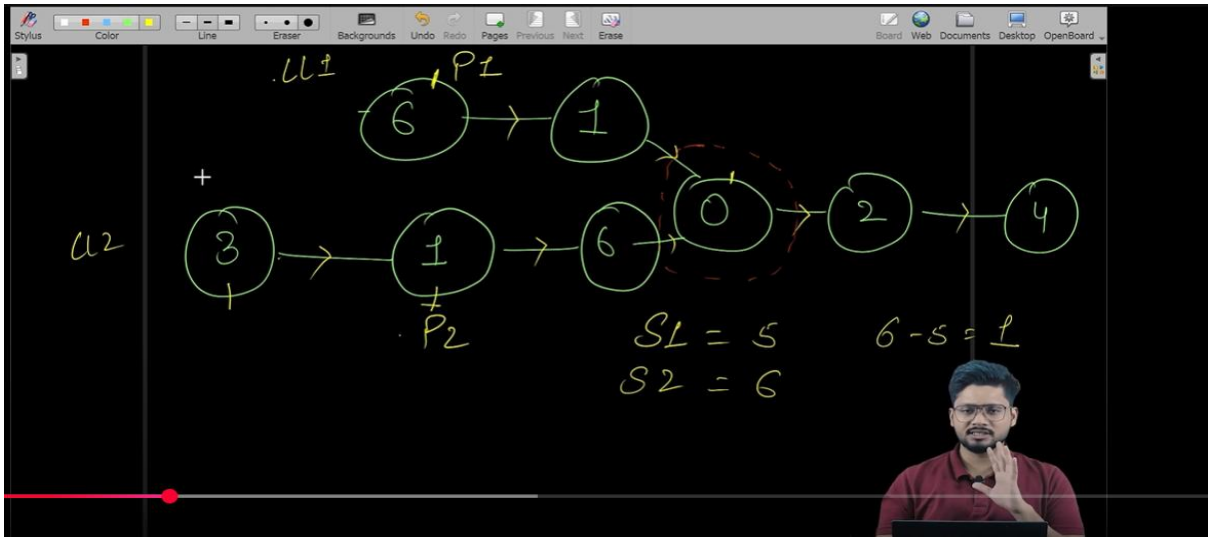
15. Remove Loop in Linked List

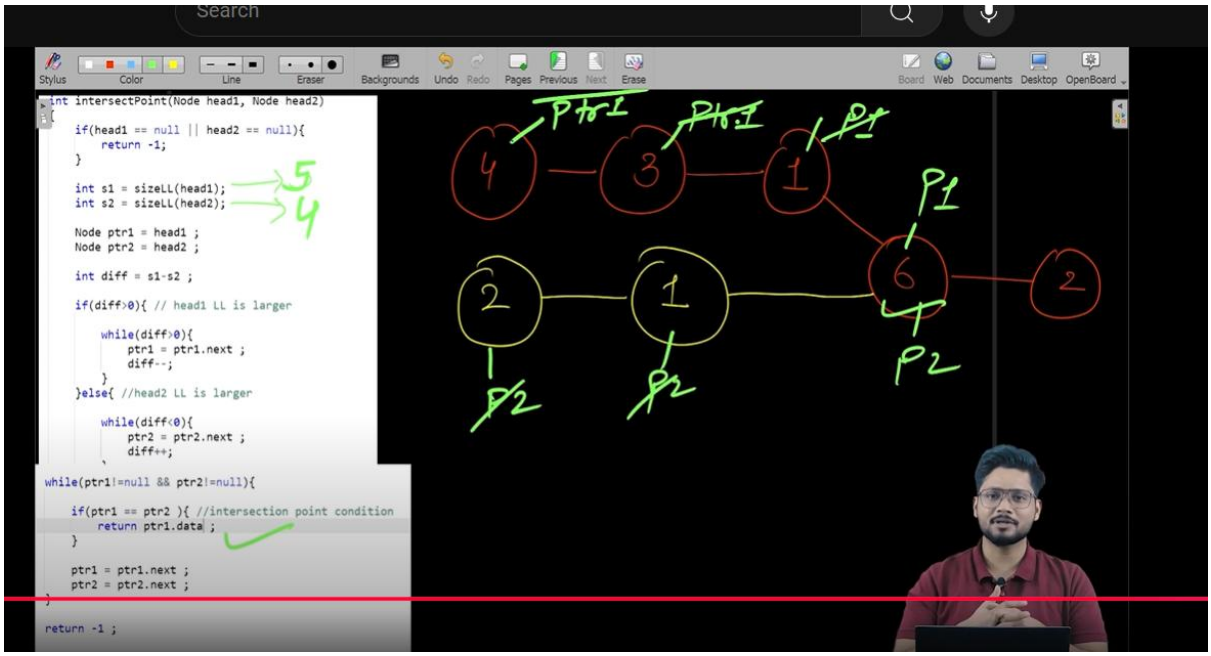




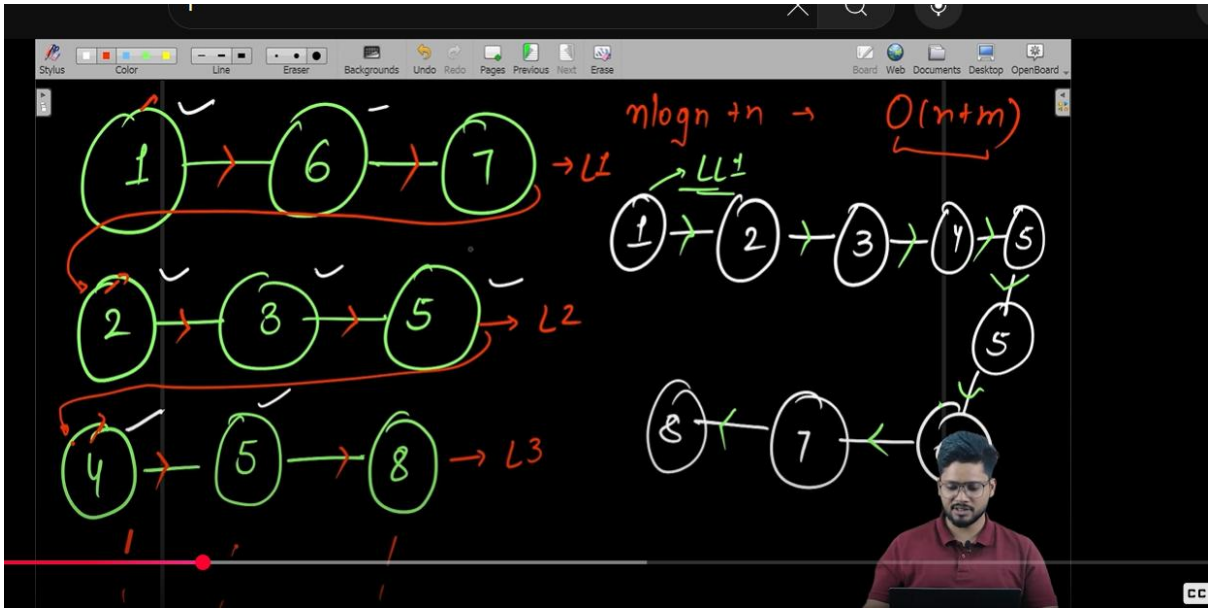
16.Intesection point in Y shaped linked list



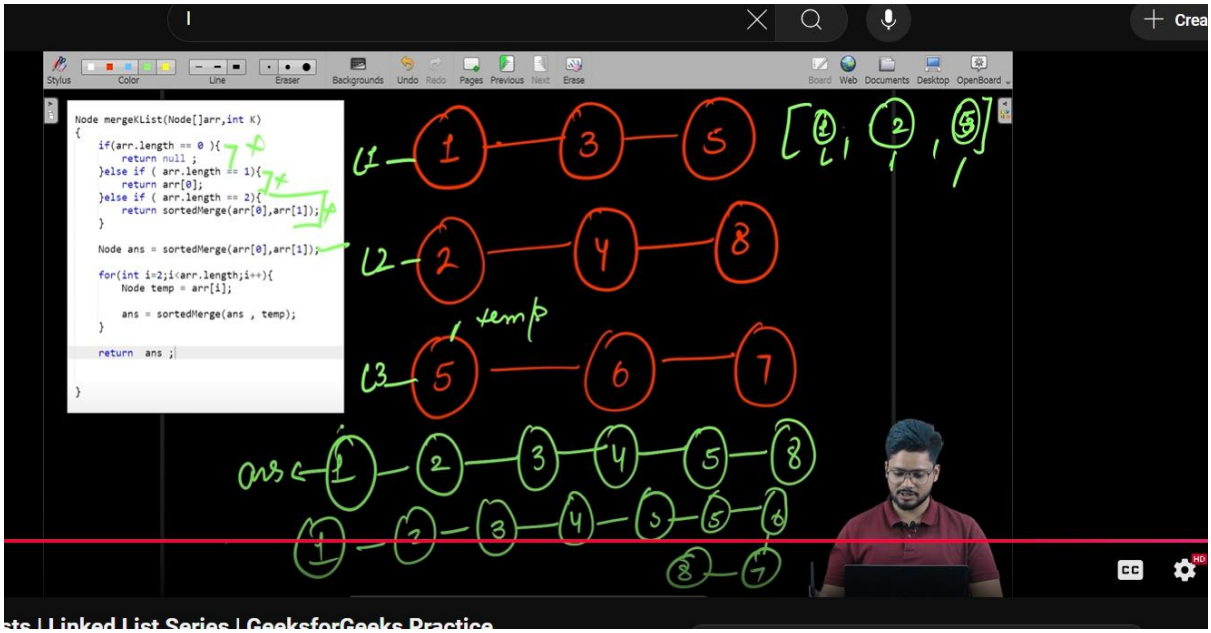




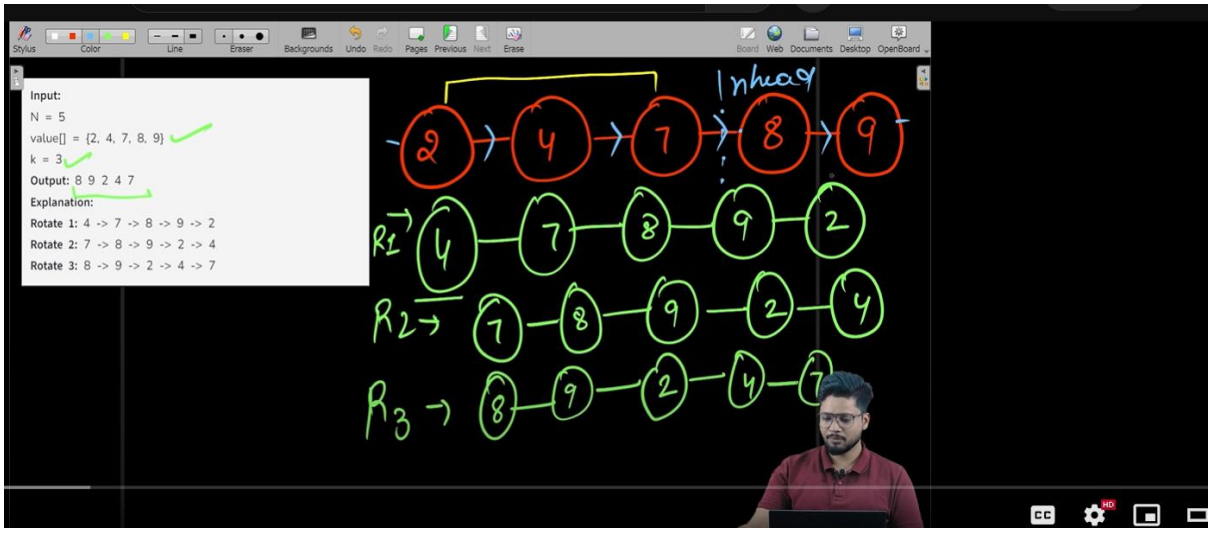
17.Merge k Sorted Linked List

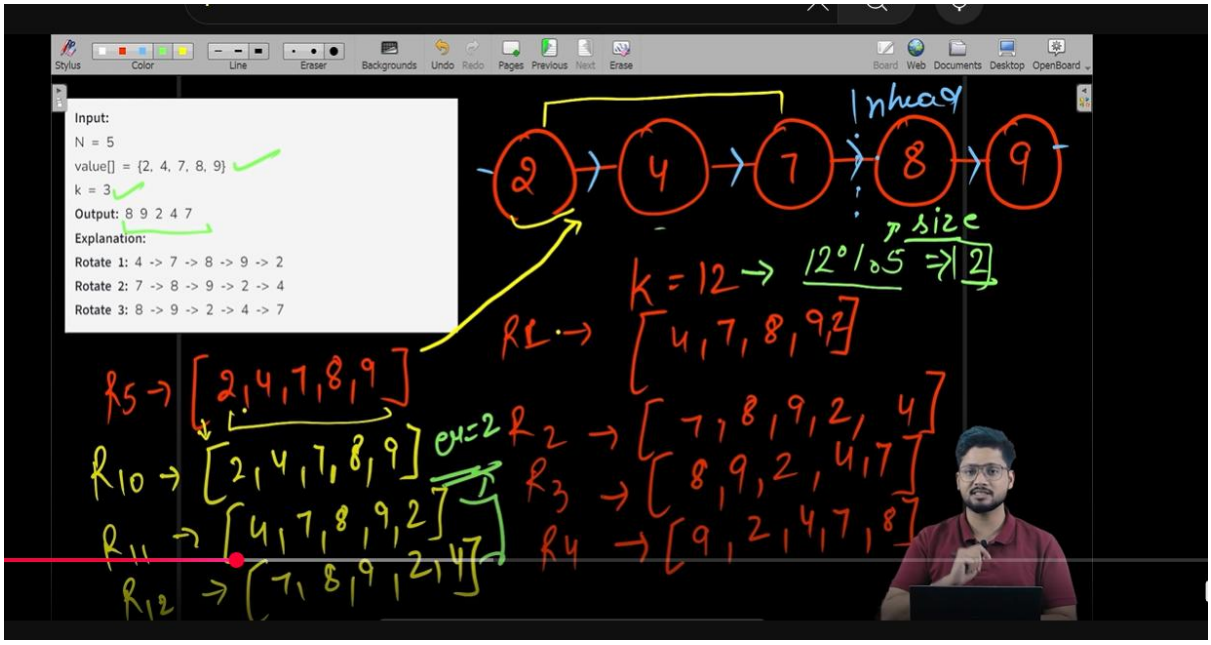


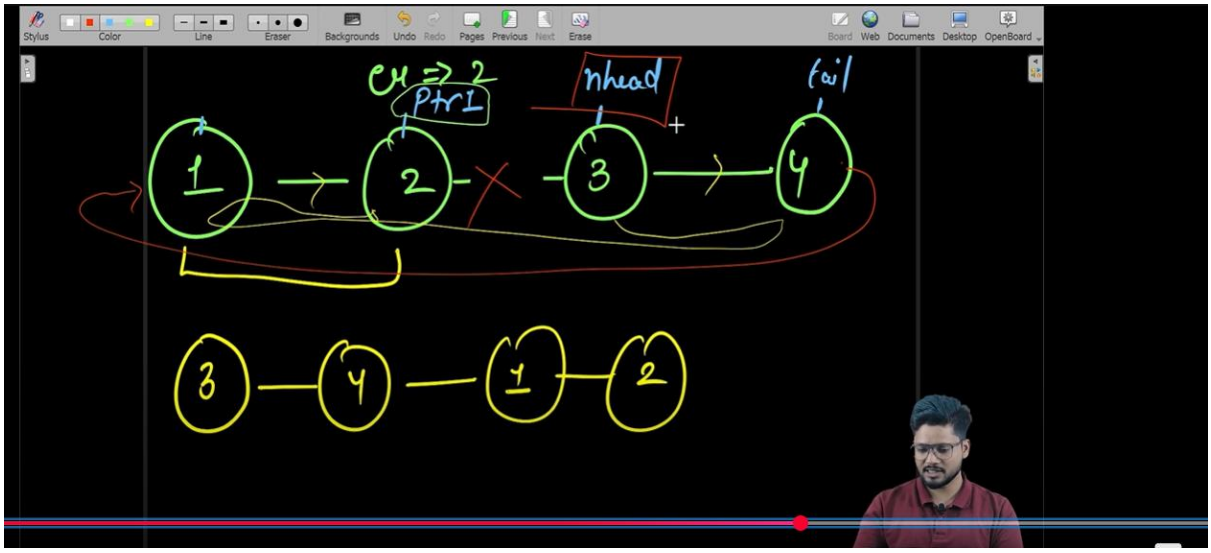
Approach : In this,first we merge the first two arr into linked list and then for upcoming arr we use for loop from i=2,and most important is to use the exact code for (merge two sorted linked list) ans we call that function(sortedMerge) when converted arr to linked node.

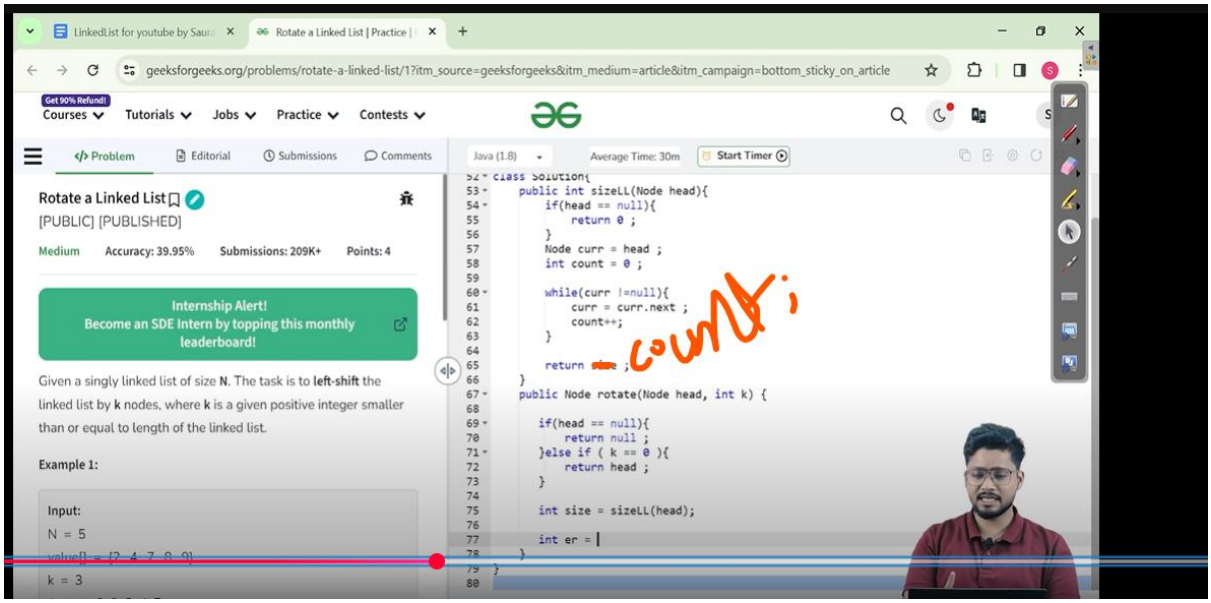


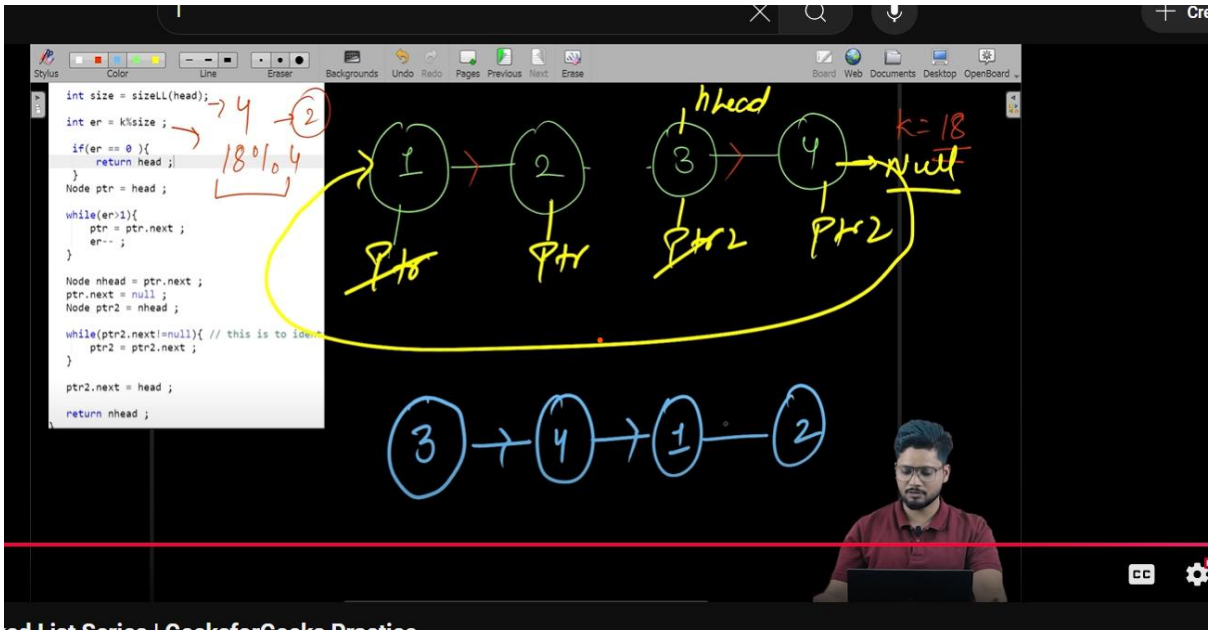
18.Rotate a Linked List



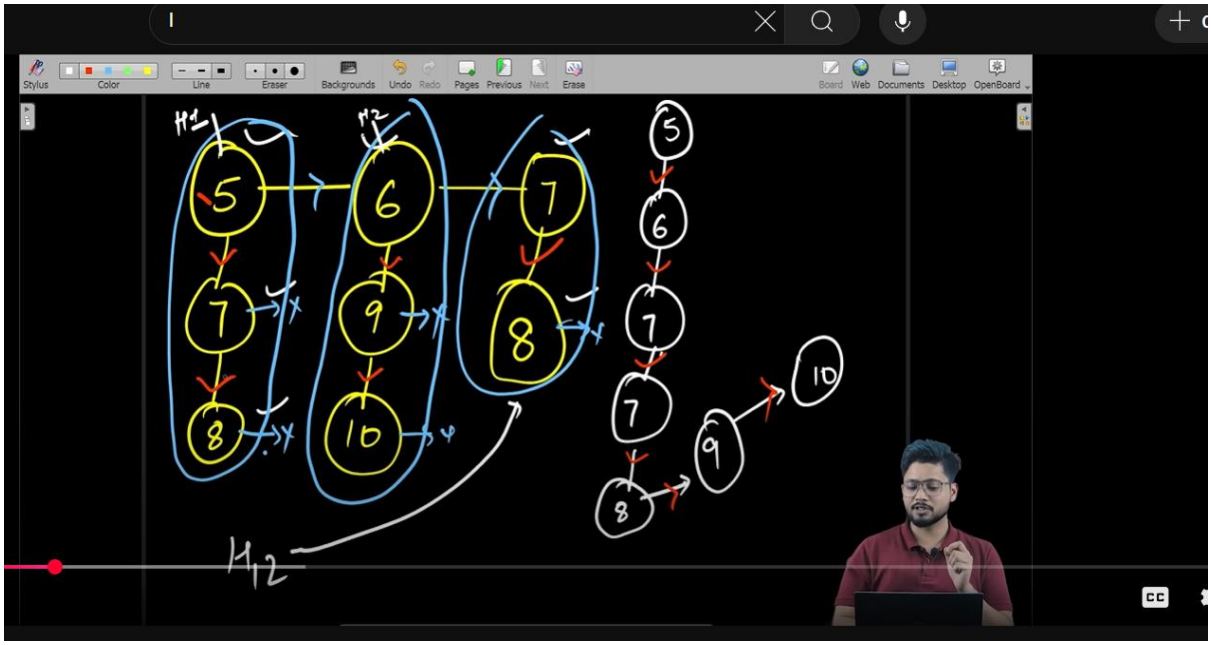


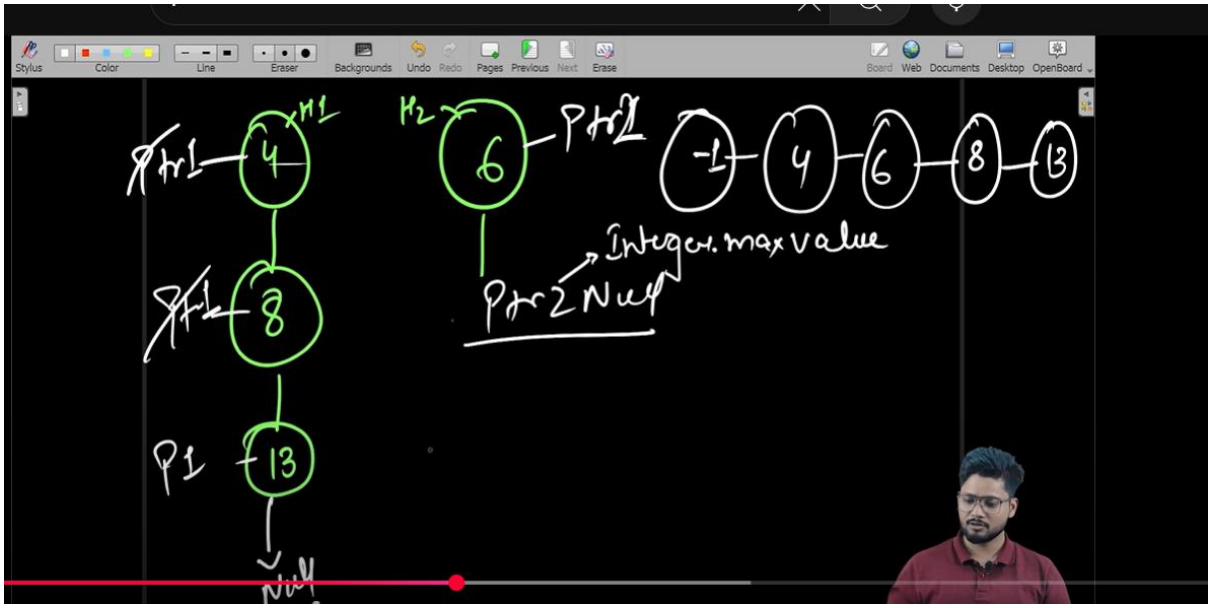



Rotate a Linked List | Practice
geeksforgeeks.org/problems/rotate-a-linked-list/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article
Courses
Tutorials
Jobs
Practice
Contests
Problem
Editorial
Submissions
Comments
Java (1.8)
Average Time: 30m
Start Timer
Rotate a Linked List
[PUBLIC] [PUBLISHED]
Medium
Accuracy: 39.95%
Submissions: 209K+
Points: 4
Internship Alert!
Become an SDE Intern by topping this monthly leaderboard!
Given a singly linked list of size N. The task is to left-shift the linked list by k nodes, where k is a given positive integer smaller than or equal to length of the linked list.
Example 1:
Input:
N = 5
k = 3
class Solution{
public int sizeLL(Node head){
if(head == null){
return 0 ;
}
Node curr = head ;
int count = 0 ;
while(curr !=null){
curr = curr.next ;
count++;
}
return count;
}
public Node rotate(Node head, int k) {
if(head == null){
return null ;
}else if (k == 0){
return head ;
}
int size = sizeLL(head);
int er =

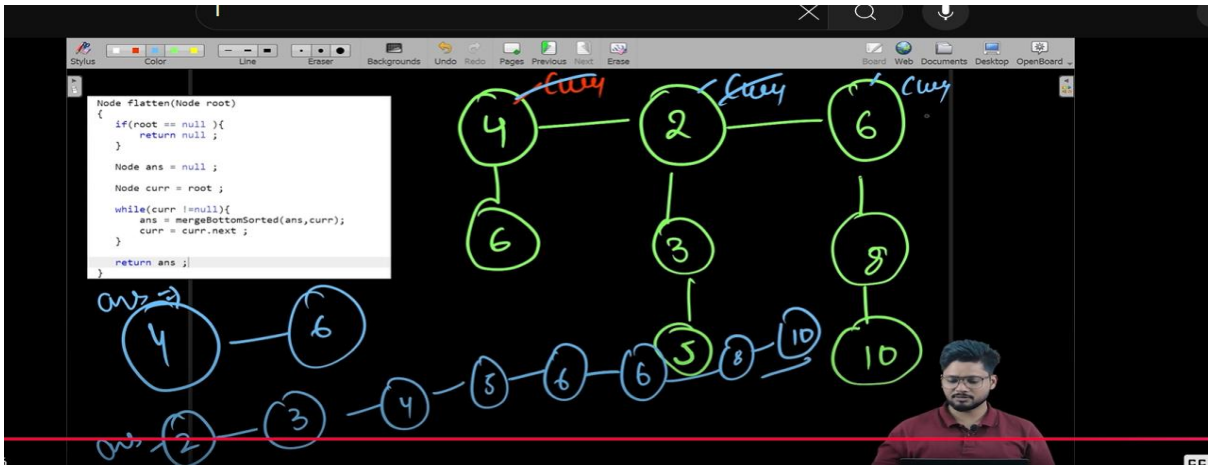

Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Desktop
OpenBoard
int size = sizeLL(head);
int er = k%size ;
if(er == 0){
return head ;
}
Node ptr = head ;
while(er>1){
ptr = ptr.next ;
er-- ;
}
Node nhead = ptr.next ;
ptr.next = null ;
Node ptr2 = nhead ;
while(ptr2.next!=null){ // this is to
ptr2 = ptr2.next ;
}
ptr2.next = head ;
return nhead ;
nHead
k=18
null
Ptr
Ptr
Ptr2
Ptr2

19.Flattening a Linked List

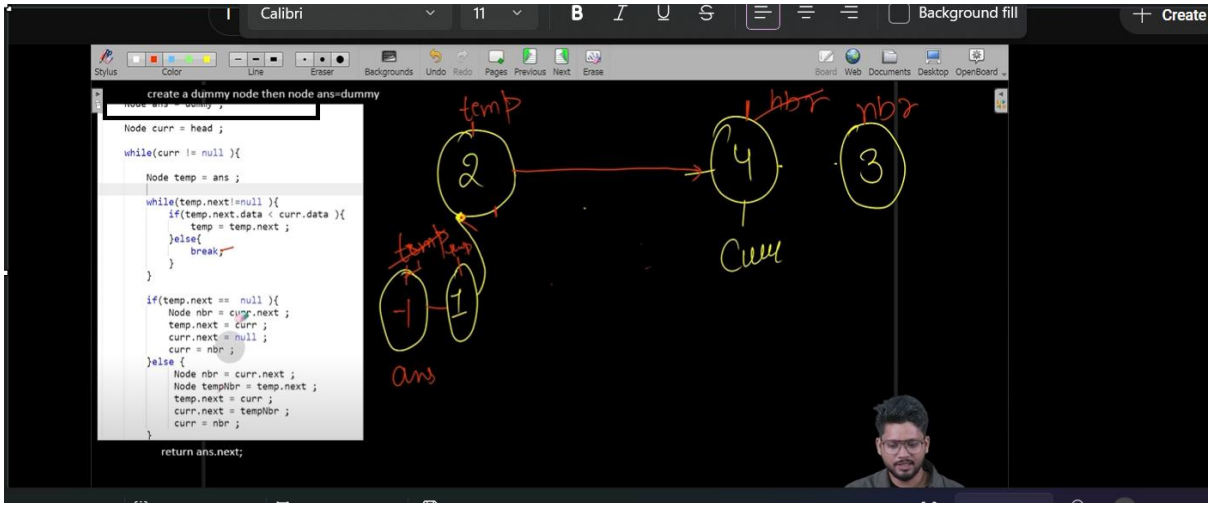




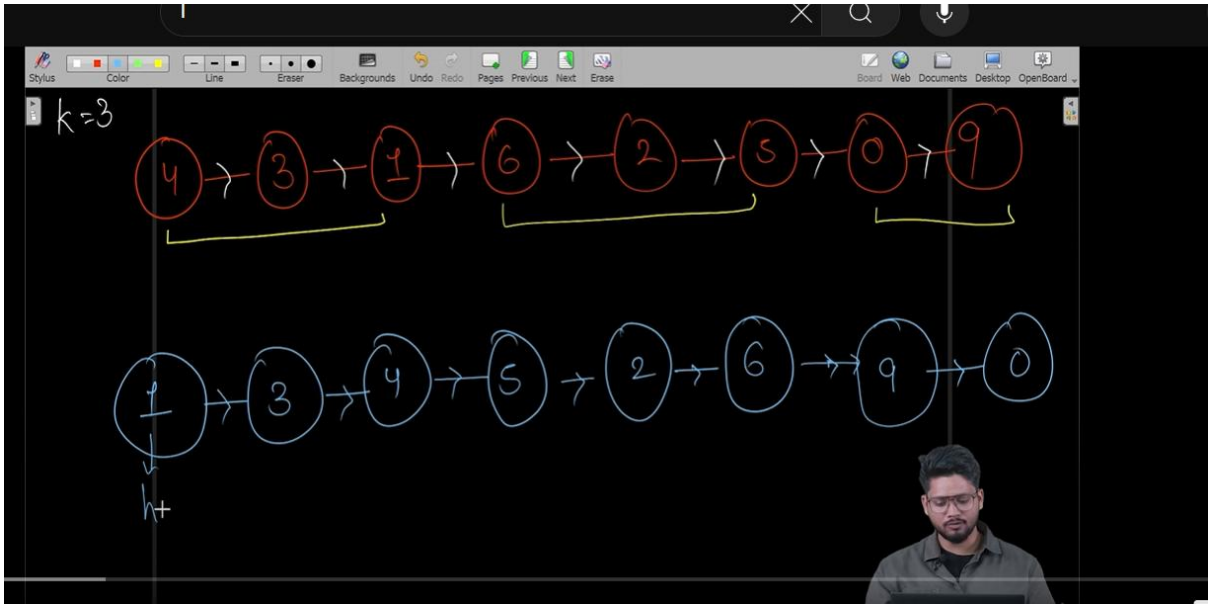
```
Node mergeBottomSorted(Node head1 , Node head2){

    if(head1 == null || head2 == null){
        return head1==null?head2:head1;
    }

    Node ptr1 = head1 ;
    Node ptr2 = head2 ;
    Node dummy = new Node(-1);
    Node ans = dummy ;

    while(ptr1!=null || ptr2!=null){

      int val1 = ptr1!=null ? ptr1.data : Integer.MAX_VALUE ;
      int val2 = ptr2!=null ? ptr2.data : Integer.MAX_VALUE ;

      if(val1<val2 ){
          dummy.bottom = ptr1 ;
          ptr1 = ptr1.bottom ;
      }else{
            dummy.bottom = ptr2 ;
          ptr2 = ptr2.bottom ;
      }

       dummy = dummy.bottom ;
    }

    return ans.bottom ;
```

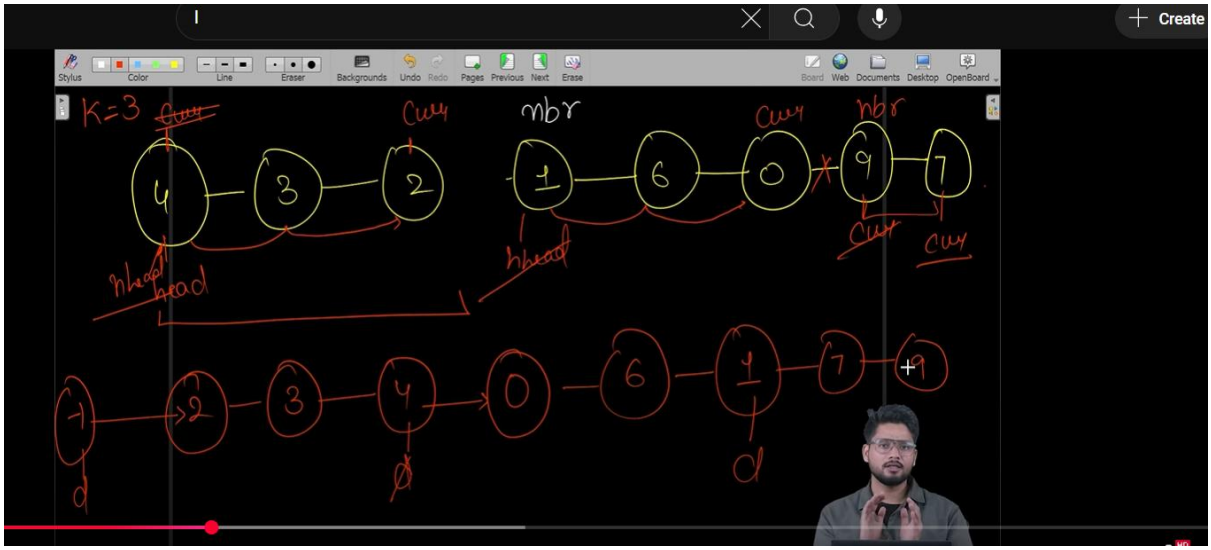


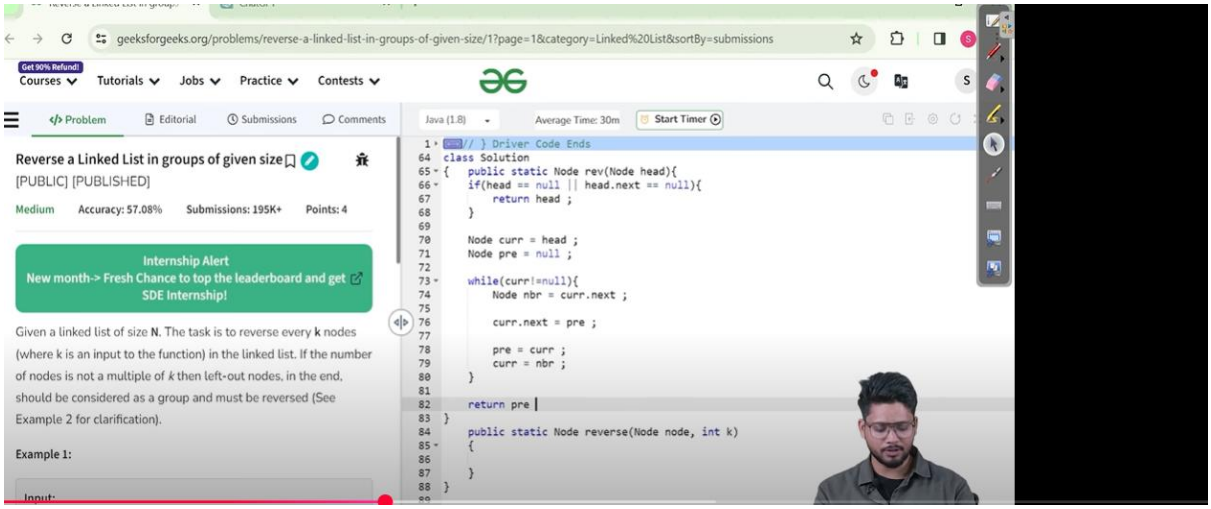
20.Insertion sort for Single Linked List

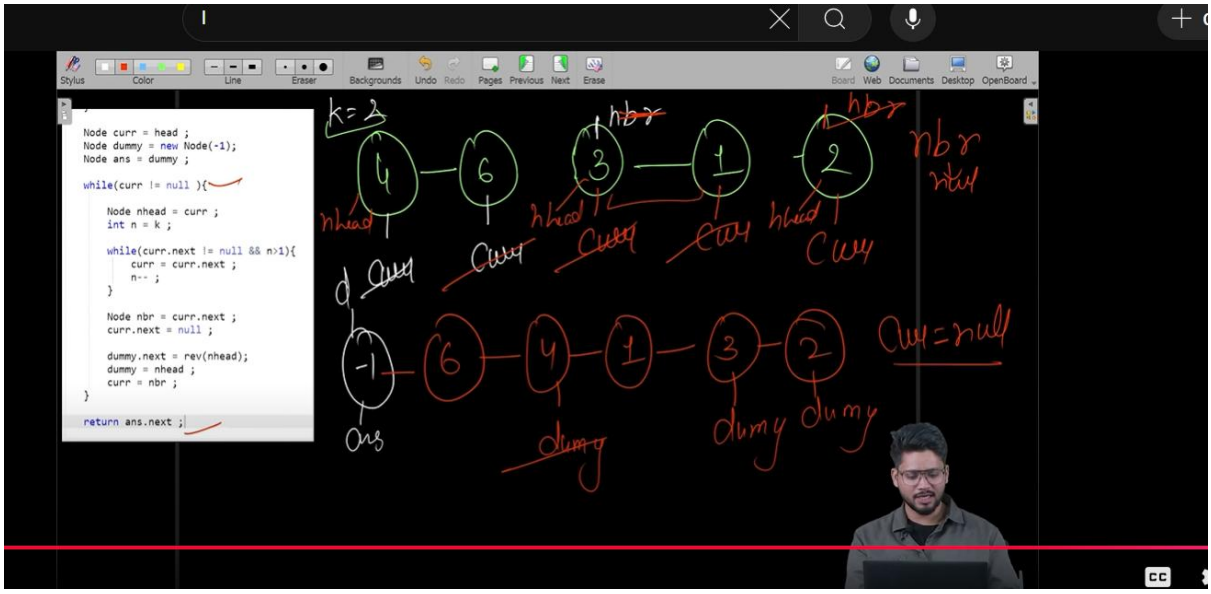


21.Reverse a Linked List in group size








Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Desktop
OpenBoard
Node curr = head ;
Node dummy = new Node(-1);
Node ans = dummy ;
while(curr != null){
Node nhead = curr ;
int n = k ;
while(curr.next != null && n>1){
curr = curr.next ;
n-- ;
}
Node nbr = curr.next ;
curr.next = null ;
dummy.next = rev(nhead);
dummy = nhead ;
curr = nbr ;
}
return ans.next ;
k=2
curr=null
dummy
dummy dumy
CC