# Assignment no.3

Ques 1.[Print linked list](https://www.geeksforgeeks.org/problems/print-linked-list-elements/0)

class Solution {

// Function to display the elements of a linked list in same line

void printList(Node head) {

// add code here.

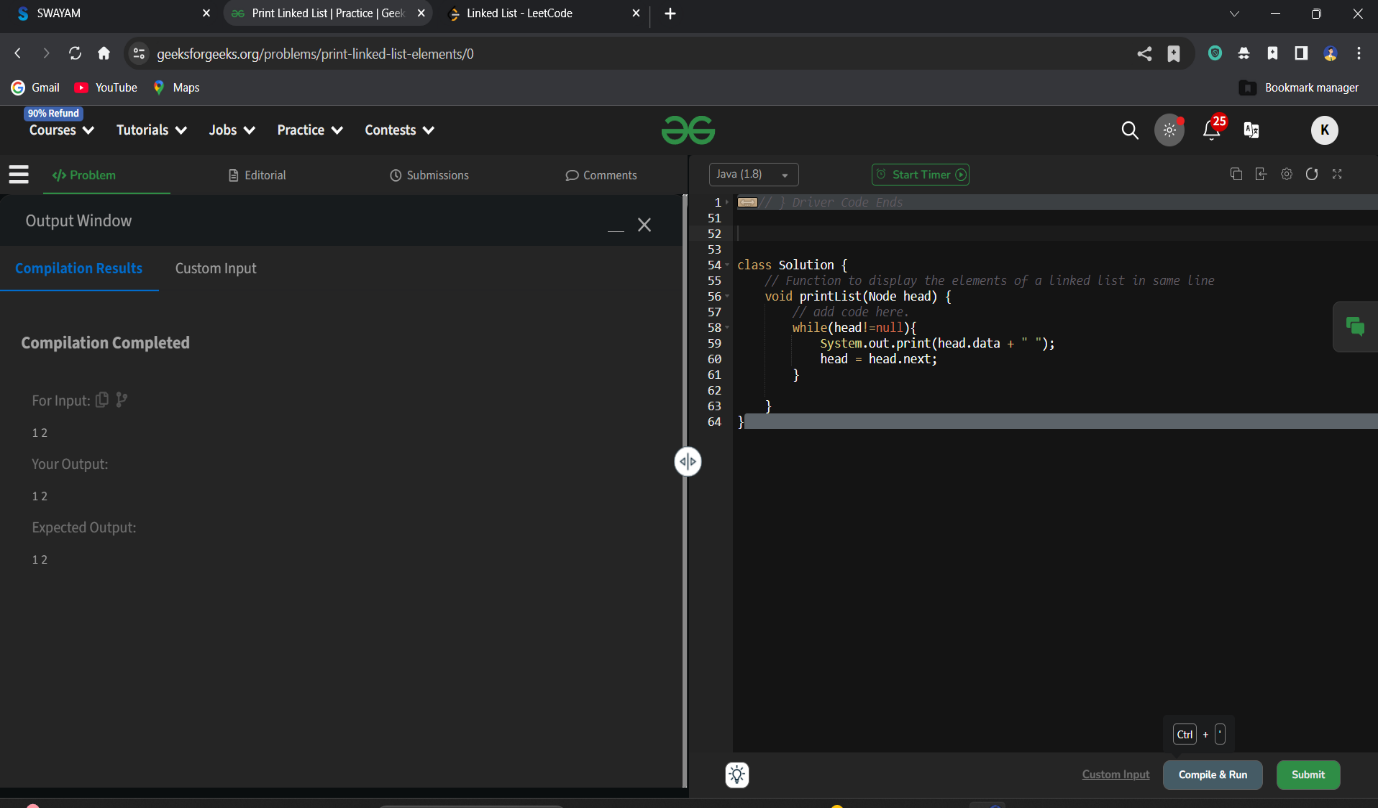
while(head!=null){

System.out.print(head.data + " ");

head = head.next;

}

}

}

Ques 2.[Remove duplicates from a sorted list](https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/)

class Solution {

    public ListNode deleteDuplicates(ListNode head) {

        ListNode res = head;

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

            if (head.val == head.next.val) {

                head.next = head.next.next;

            } else {

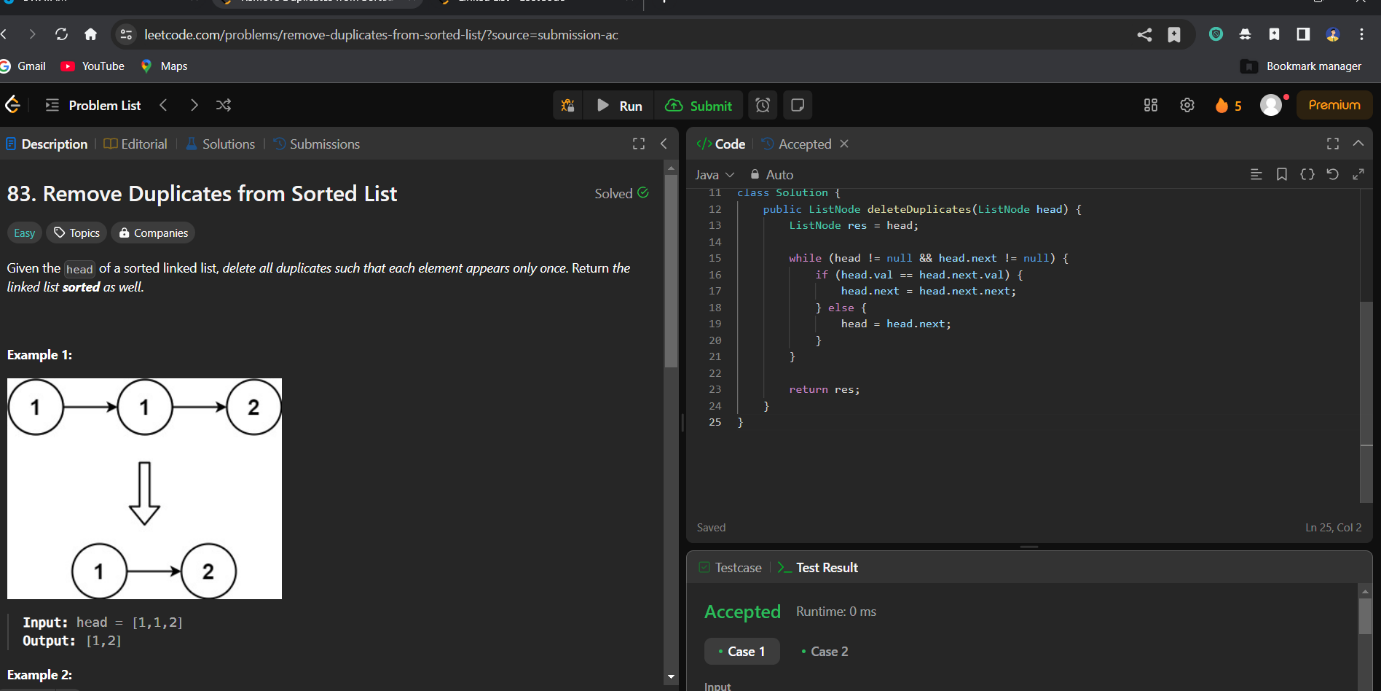
                head = head.next;

            }

        }

return res;

    }

}

Ques 3. Reverse a linked list] (<https://leetcode.com/problems/reverse-linked-list/>

class Solution {

    public ListNode reverseList(ListNode head) {

        ListNode node = null;

        while (head != null) {

            ListNode temp = head.next;

            head.next = node;

            node = head;

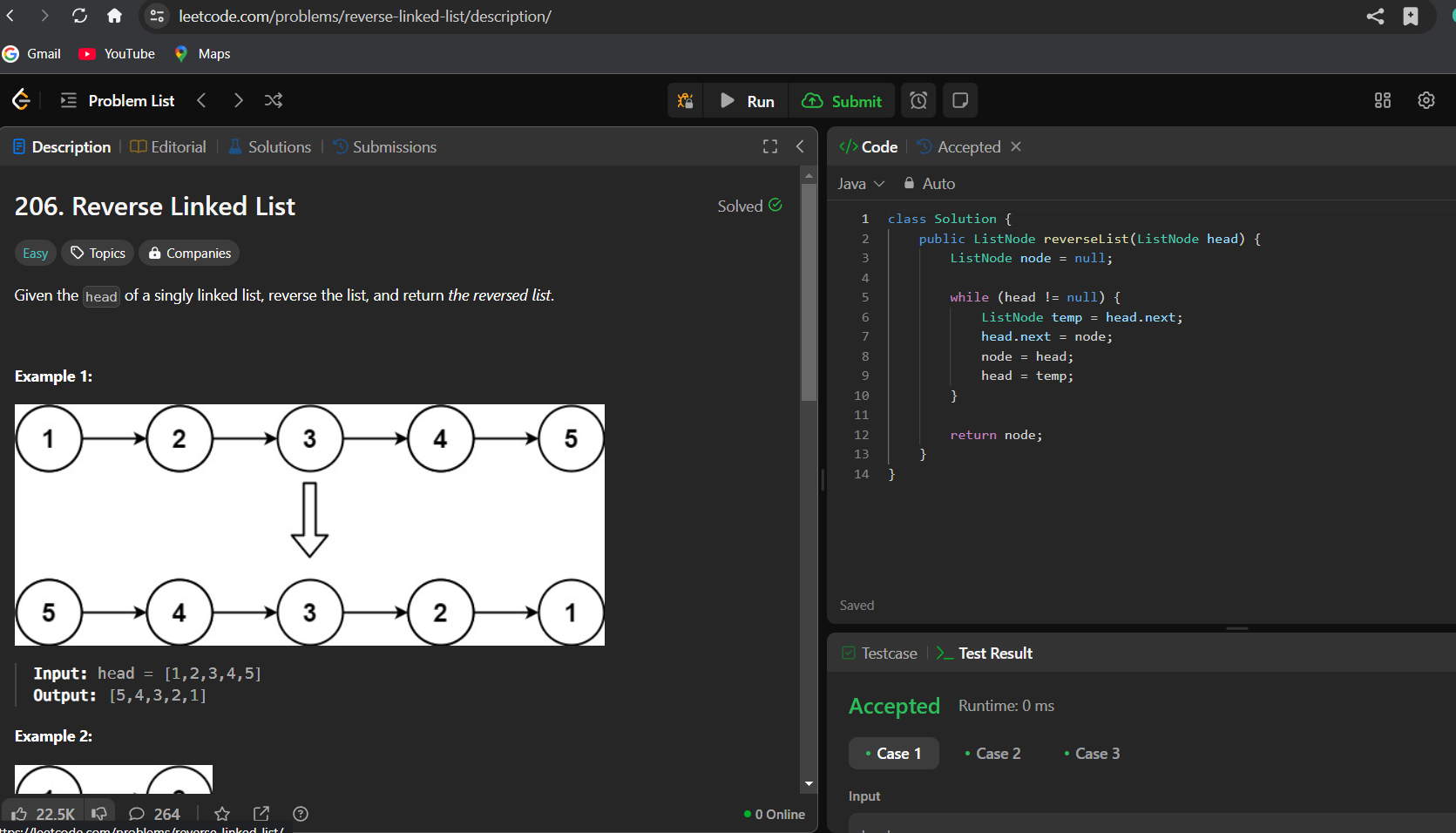
            head = temp;

        }

        return node;

    }

}



Ques 4. [Delete middle node of a list](https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/)

class Solution {

    public ListNode deleteMiddle(ListNode head) {

        if(head == null)return null;

        ListNode prev = new ListNode(0);

        prev.next = head;

        ListNode slow = prev;

        ListNode fast = head;

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

            slow = slow.next;

            fast = fast.next.next;

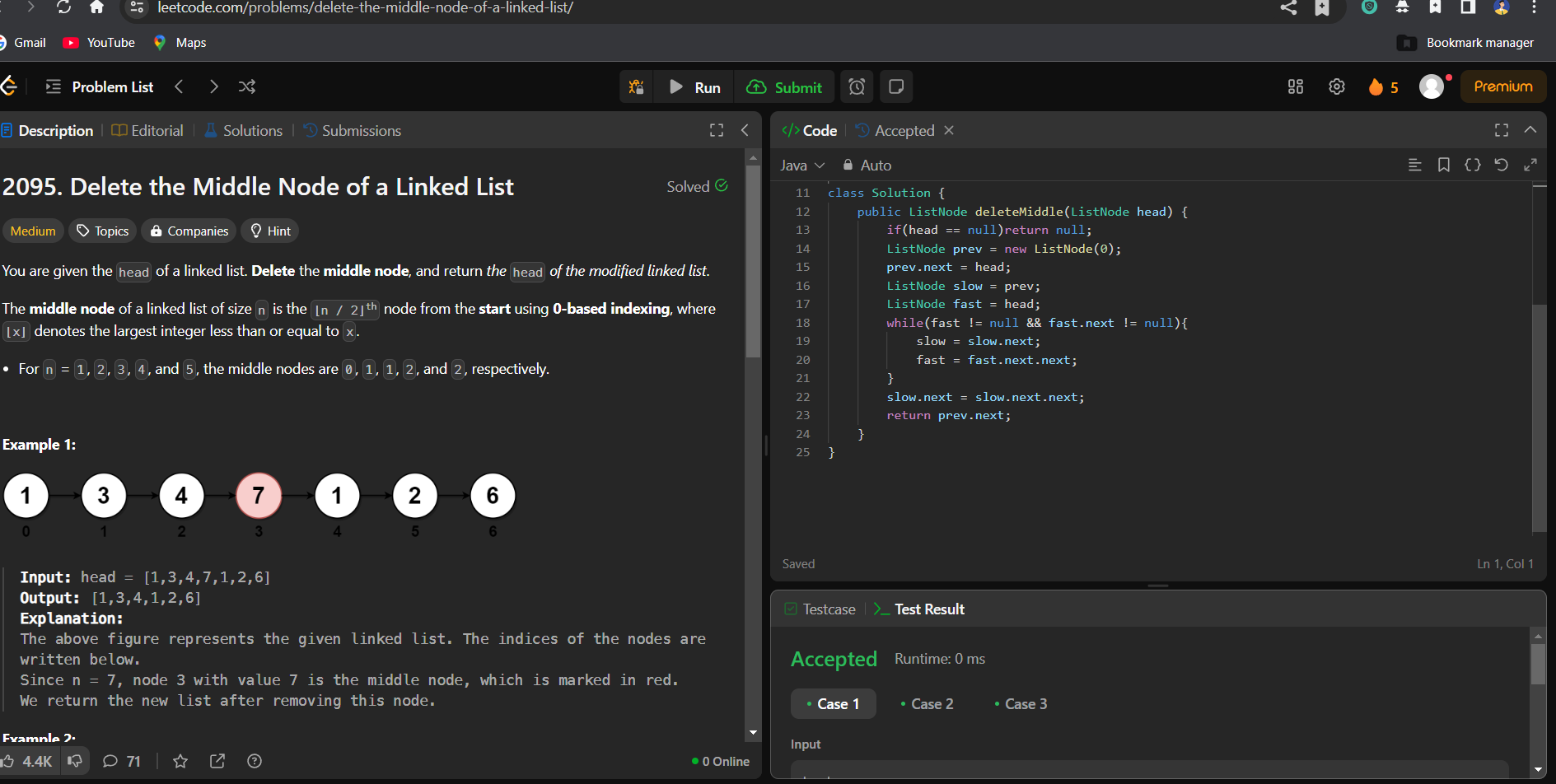
        }

        slow.next = slow.next.next;

        return prev.next;

    }

}



Ques 5.[Merge two sorted linked lists](https://leetcode.com/problems/merge-two-sorted-lists/description/)

class Solution {

    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

        if(list1!=null && list2!=null){

        if(list1.val<list2.val){

            list1.next=mergeTwoLists(list1.next,list2);

            return list1;

            }

            else{

                list2.next=mergeTwoLists(list1,list2.next);

                return list2;

        }

        }

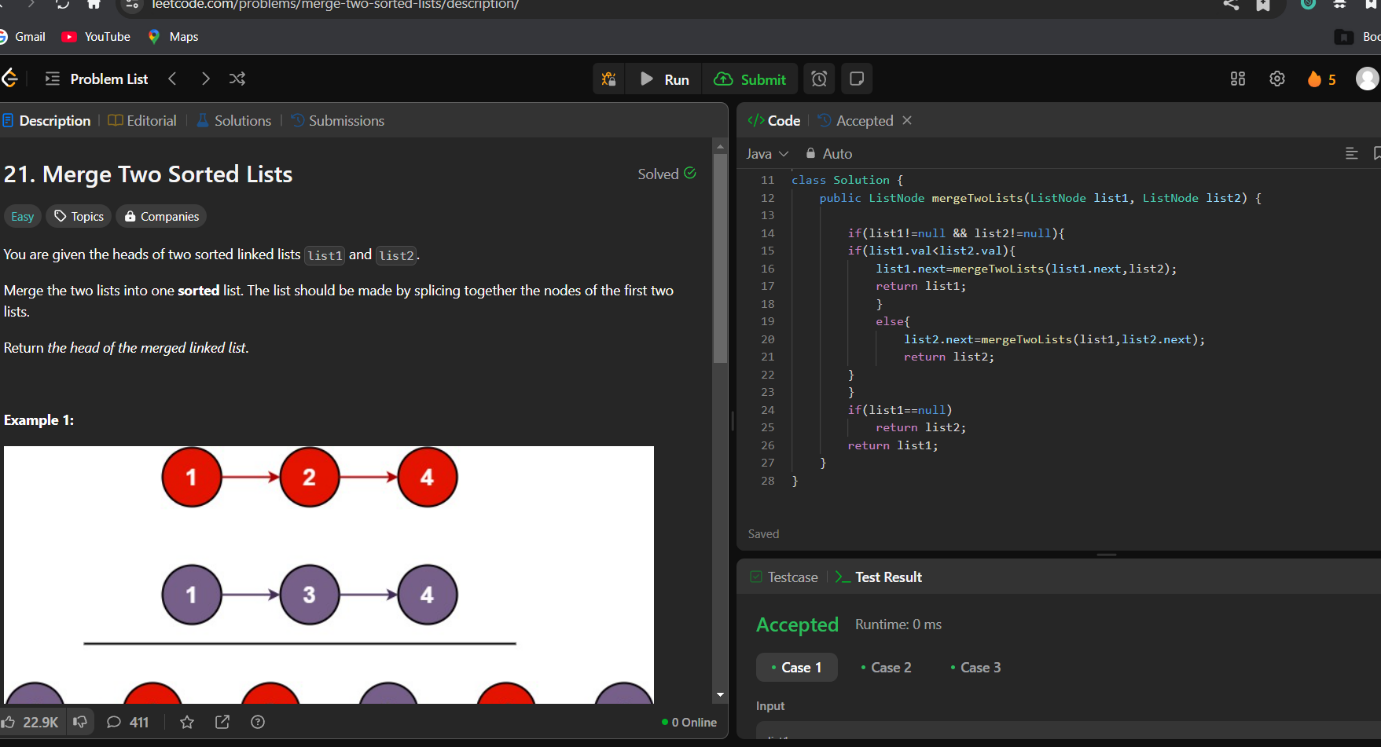
        if(list1==null)

            return list2;

        return list1;

    }

}



Ques 6.[Remove duplicates from sorted lists 2](https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/description/)

class Solution {

    public ListNode deleteDuplicates(ListNode head) {

        ListNode ans = new ListNode(1000, head); // Dummy node to handle edge cases

        ListNode cur = ans;

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

            if (cur.next.val == cur.next.next.val) { // Check if duplicates exist

                int val = cur.next.val;

                while (cur.next != null && cur.next.val == val) { // Skip all duplicates

                    cur.next = cur.next.next;

                }

            } else {

                cur = cur.next; // Move to the next node

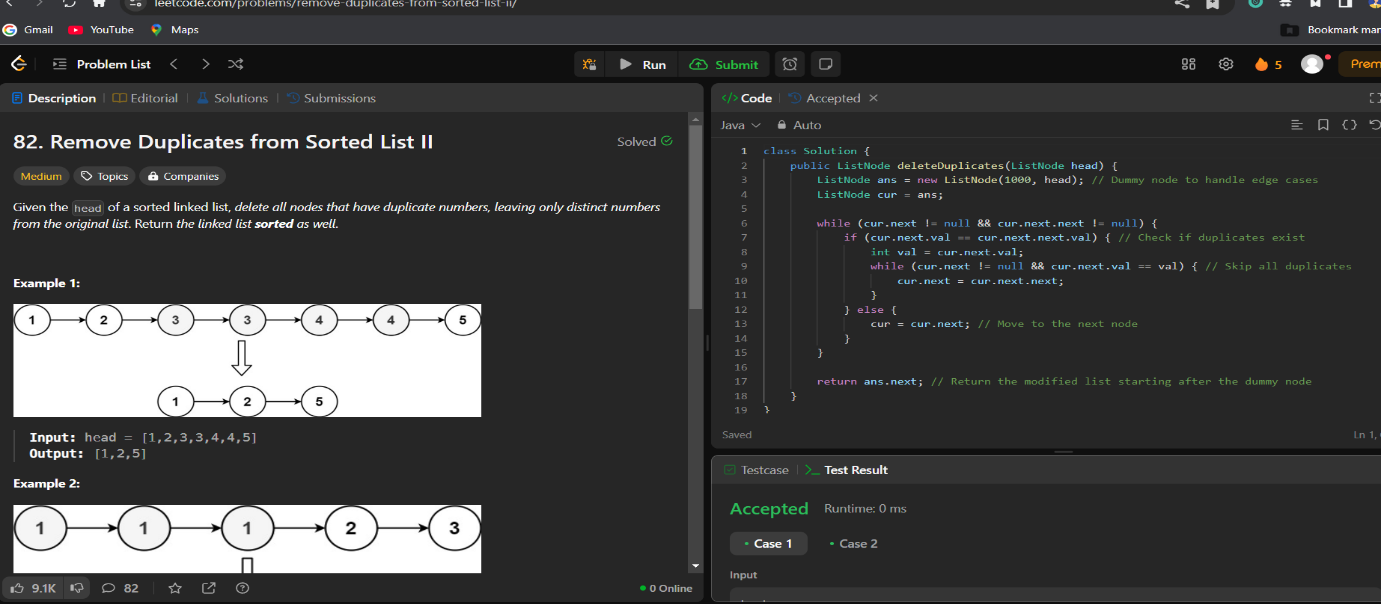
            }

        }

        return ans.next; // Return the modified list starting after the dummy node

    }

}



Ques 7.[Detect a cycle in a linked list](https://leetcode.com/problems/linked-list-cycle/description/)

public class Solution {

    public boolean hasCycle(ListNode head) {

        ListNode fast = head;

        ListNode slow = head;

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

            fast = fast.next.next;

            slow = slow.next;

            if (fast == slow) {

                return true;

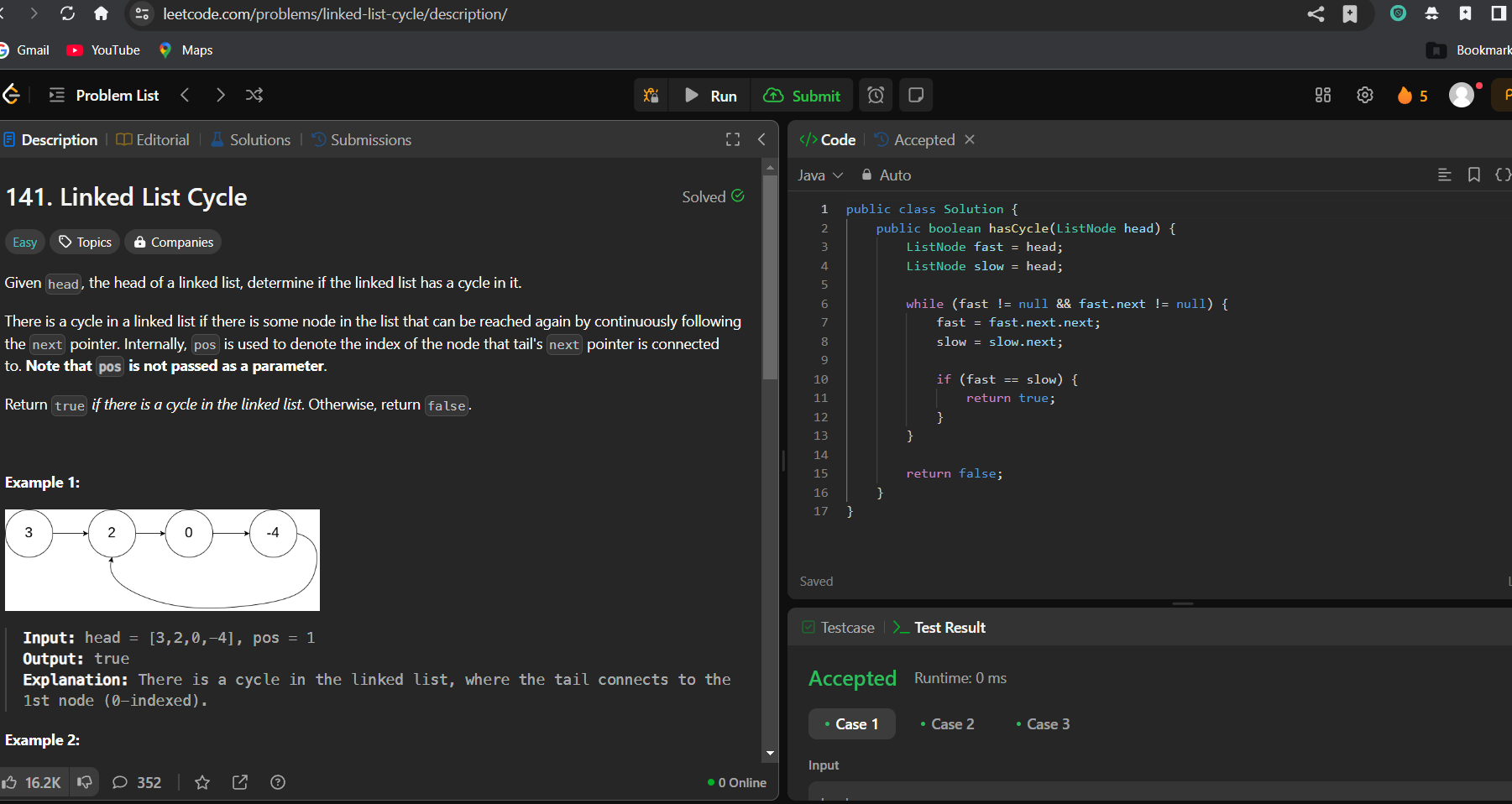
            }

        }

        return false;

    }

}



Ques 8.[Reverse linked list 2](https://leetcode.com/problems/reverse-linked-list-ii/description/)

class Solution {

    public ListNode reverseBetween(ListNode head, int left, int right) {

        if (head == null || left == right) {

            return head;

        }

        ListNode dummy = new ListNode(0);

        dummy.next = head;

        ListNode prev = dummy;

        for (int i = 0; i < left - 1; i++) {

            prev = prev.next;

        }

        ListNode cur = prev.next;

        for (int i = 0; i < right - left; i++) {

            ListNode temp = cur.next;

            cur.next = temp.next;

            temp.next = prev.next;

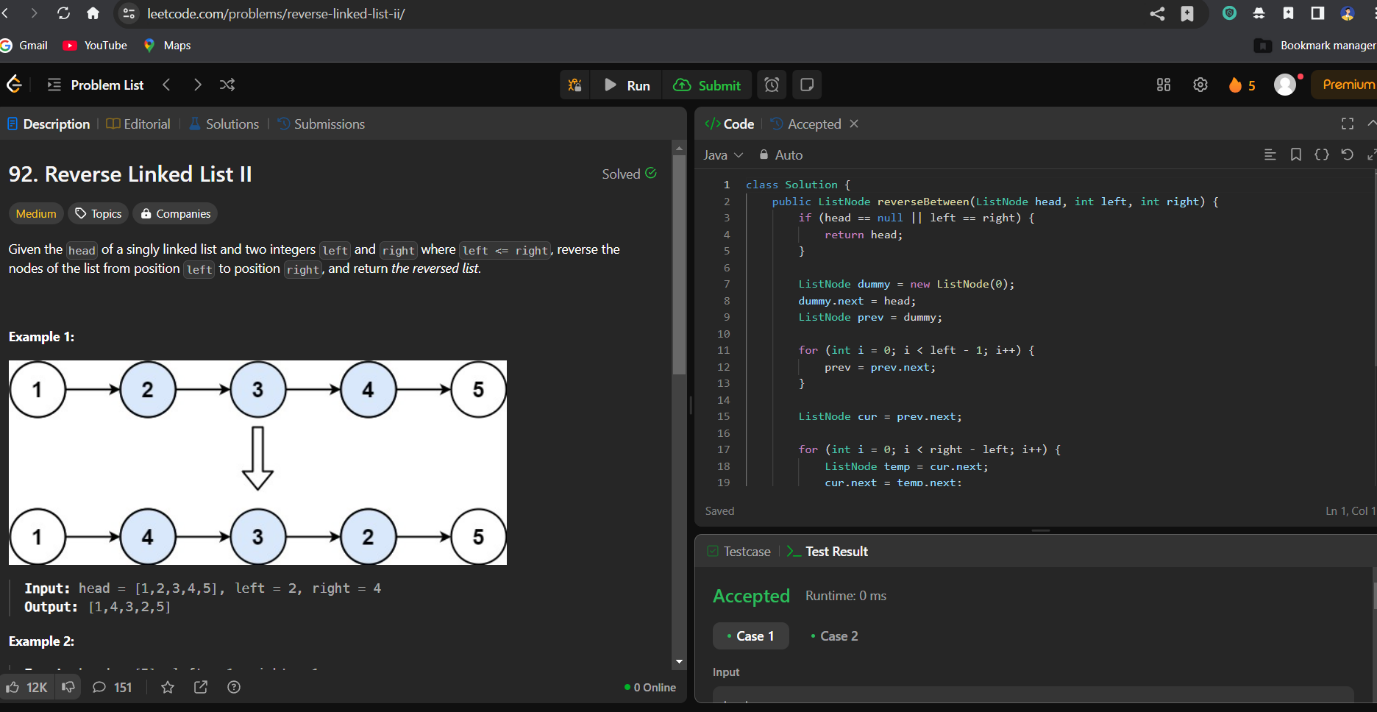
            prev.next = temp;

        }

return dummy.next;

    }

}



Ques 9.[rotate a list](https://leetcode.com/problems/rotate-list/description/)

class Solution {

    public ListNode rotateRight(ListNode head, int k) {

        if(head==null)

        return head;

        int size=size(head);

        if(k==0 || k%size==0)

        return head;

        if(k>size){

            k=k%size;

        }

        k=size-k;

        ListNode temp=head;

        while(temp.next!=null){

            temp=temp.next;

        }

        ListNode tail=temp;

        temp=head;

        while(k!=1)

        {

            temp=temp.next;

            k--;

        }

        tail.next=head;

        head=temp.next;

        temp.next=null;

        return head;

    }

    public int size(ListNode head){

        int size=0;

        while(head!=null){

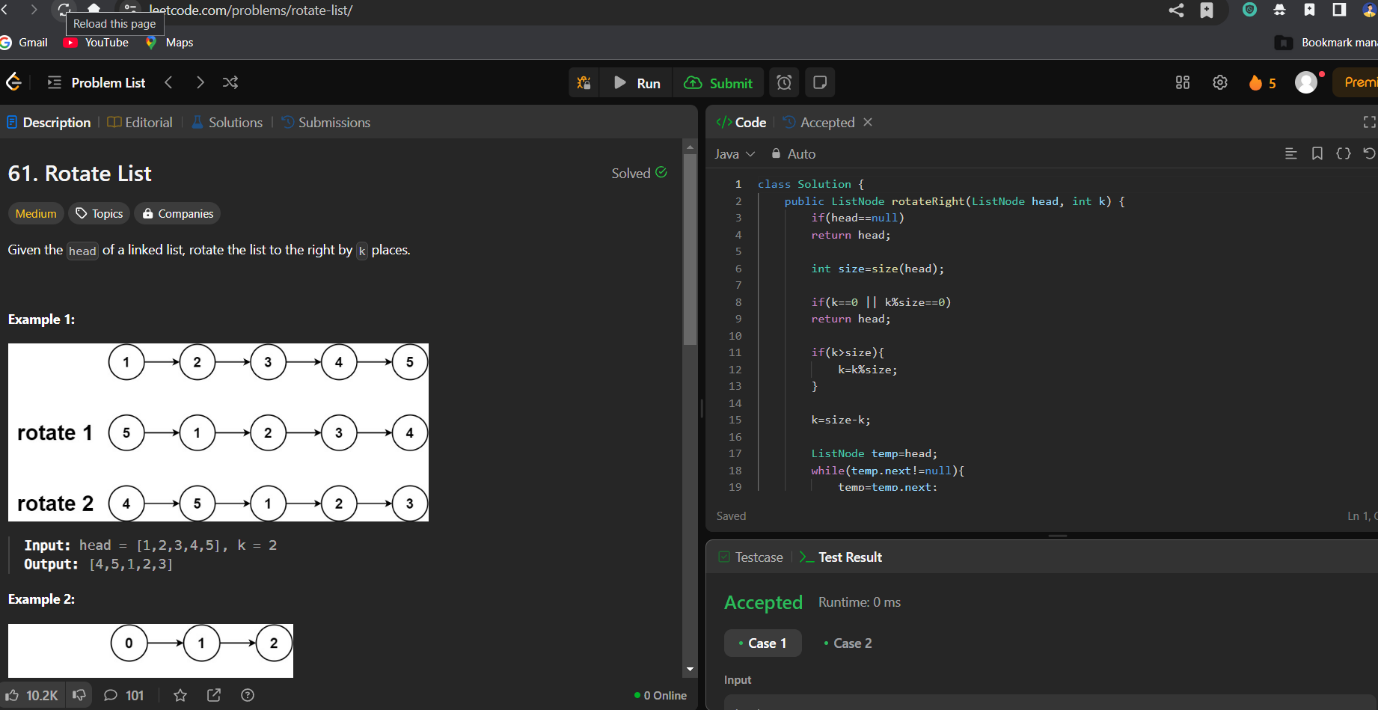
            head=head.next;

            size++;

        }return size;

    }

}



Ques 10.[Sort List](https://leetcode.com/problems/sort-list/description/)

public class Solution {

  public ListNode sortList(ListNode head) {

    if (head == null || head.next == null)

      return head;

    // step 1. cut the list to two halves

    ListNode prev = null, slow = head, fast = head;

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

      prev = slow;

      slow = slow.next;

      fast = fast.next.next;

    }

    prev.next = null;

    // step 2. sort each half

    ListNode l1 = sortList(head);

    ListNode l2 = sortList(slow);

    // step 3. merge l1 and l2

    return merge(l1, l2);

  }

  ListNode merge(ListNode l1, ListNode l2) {

    ListNode l = new ListNode(0), p = l;

    while (l1 != null && l2 != null) {

      if (l1.val < l2.val) {

        p.next = l1;

        l1 = l1.next;

      } else {

        p.next = l2;

        l2 = l2.next;

      }

      p = p.next;

    }

    if (l1 != null)

      p.next = l1;

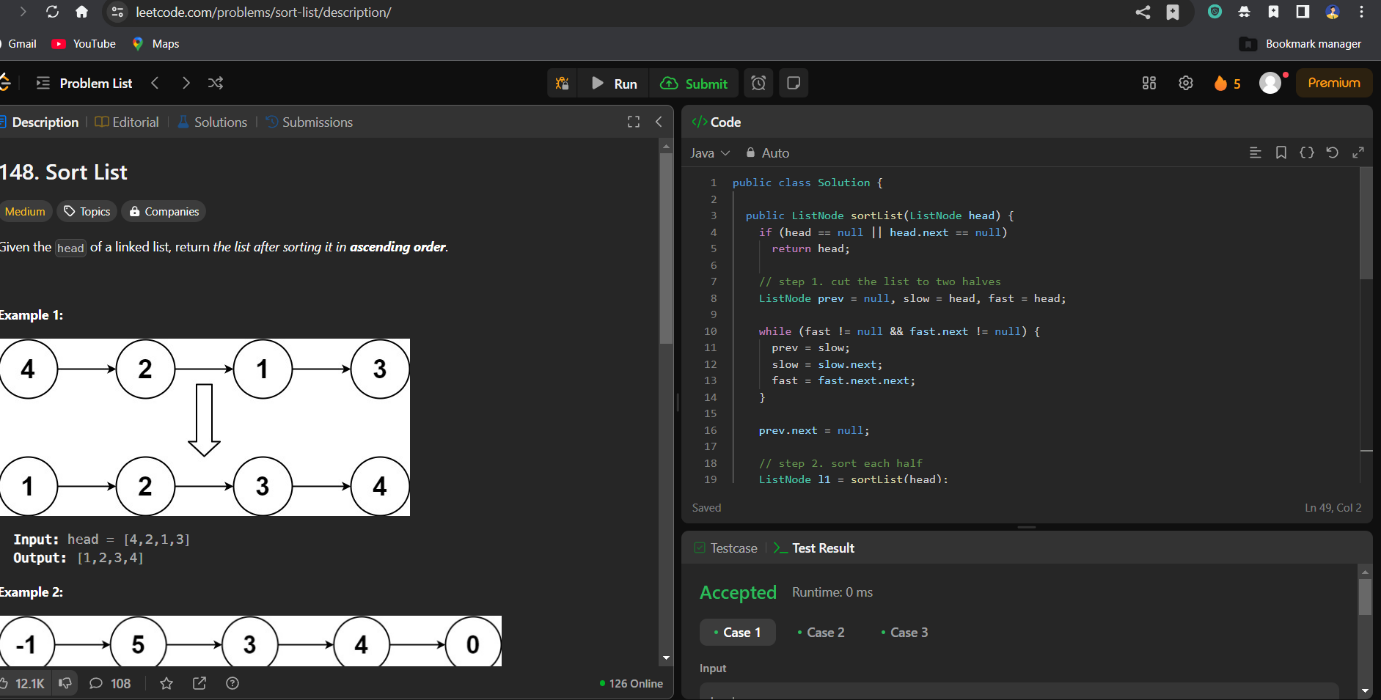
    if (l2 != null)

      p.next = l2;

    return l.next;

  }

}



Ques 11.[Detect a cycle in a linked list 2](https://leetcode.com/problems/linked-list-cycle-ii/description/)

public class Solution {

    public ListNode detectCycle(ListNode head) {

        ListNode slow = head, fast = head;

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

            slow = slow.next;

            fast = fast.next.next;

            if (slow == fast) break;

        }

        if (fast == null || fast.next == null) return null;

        while (head != slow) {

            head = head.next;

            slow = slow.next;

        }

        return head;

    }

}

