**Rattan Kaur Bajwa**

**22BCS16185**

**FL-604-A**

Remove duplicates from a sorted arrayCW

<https://leetcode.com/problems/remove-duplicates-from-sorted-array/>

class Solution {

    public int removeDuplicates(int[] nums) {

     int index=1;

     for(int i=1;i<nums.length;i++){

        if(nums[i]!=nums[i-1]){

          nums[index]=nums[i];

          index++;

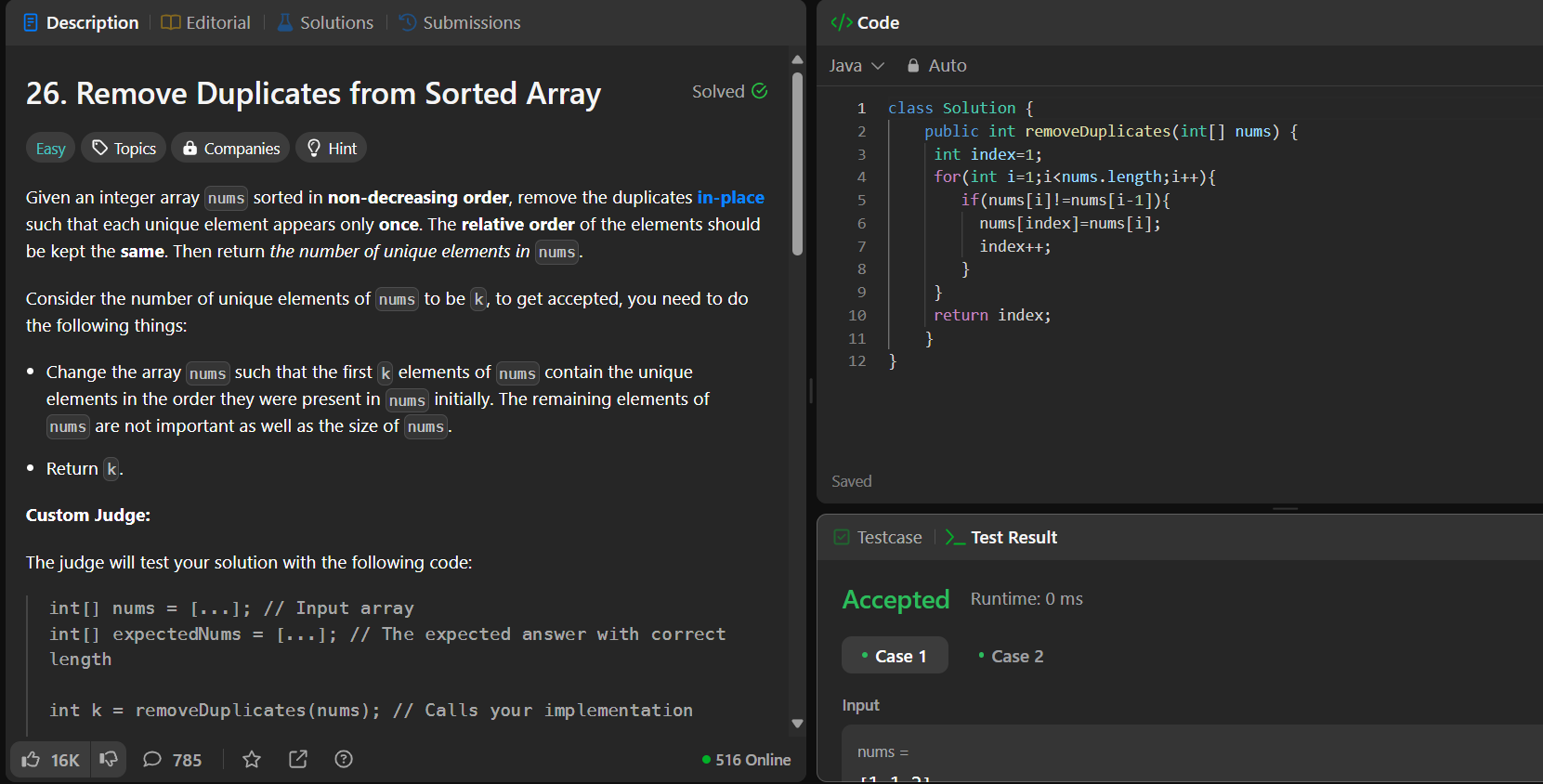
        }

     }

     return index;

    }

}



1. Implementing insertion sortCW

<https://www.geeksforgeeks.org/problems/insertion-sort/1>

class Solution {

// Please change the array in-place

public void insertionSort(int arr[]) {

for(int i=1;i<arr.length;i++){

for(int j=0;j<i;j++){

if(arr[j]>arr[i]){

int temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

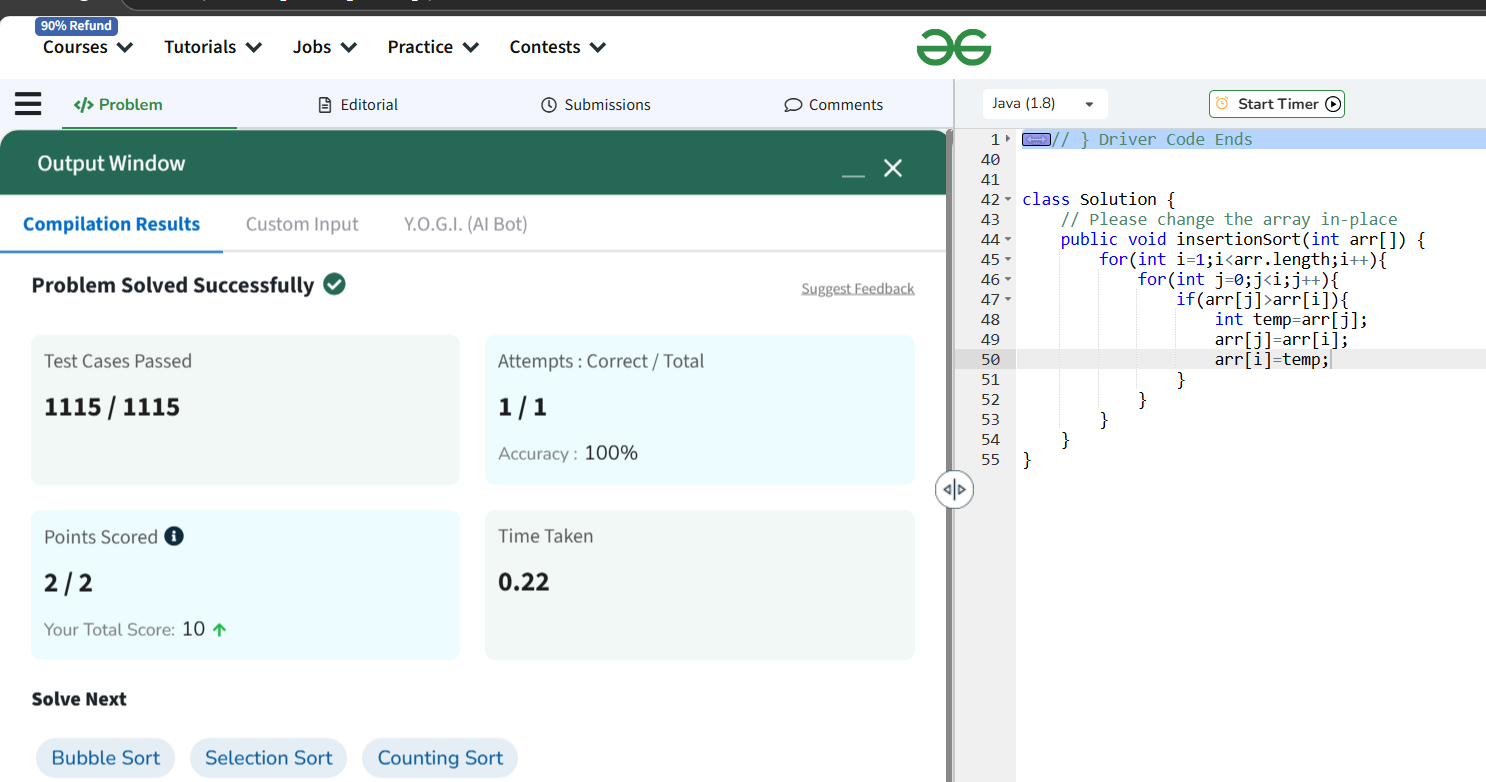
}

}

}

}

}



Contains duplicateHW

<https://leetcode.com/problems/contains-duplicate/description/>

class Solution {

    public boolean containsDuplicate(int[] nums) {

        for(int i=0;i<nums.length;i++){

            for(int j=i+1;j<nums.length;j++){

                if(nums[i]==nums[j]){

                    return true;

                }

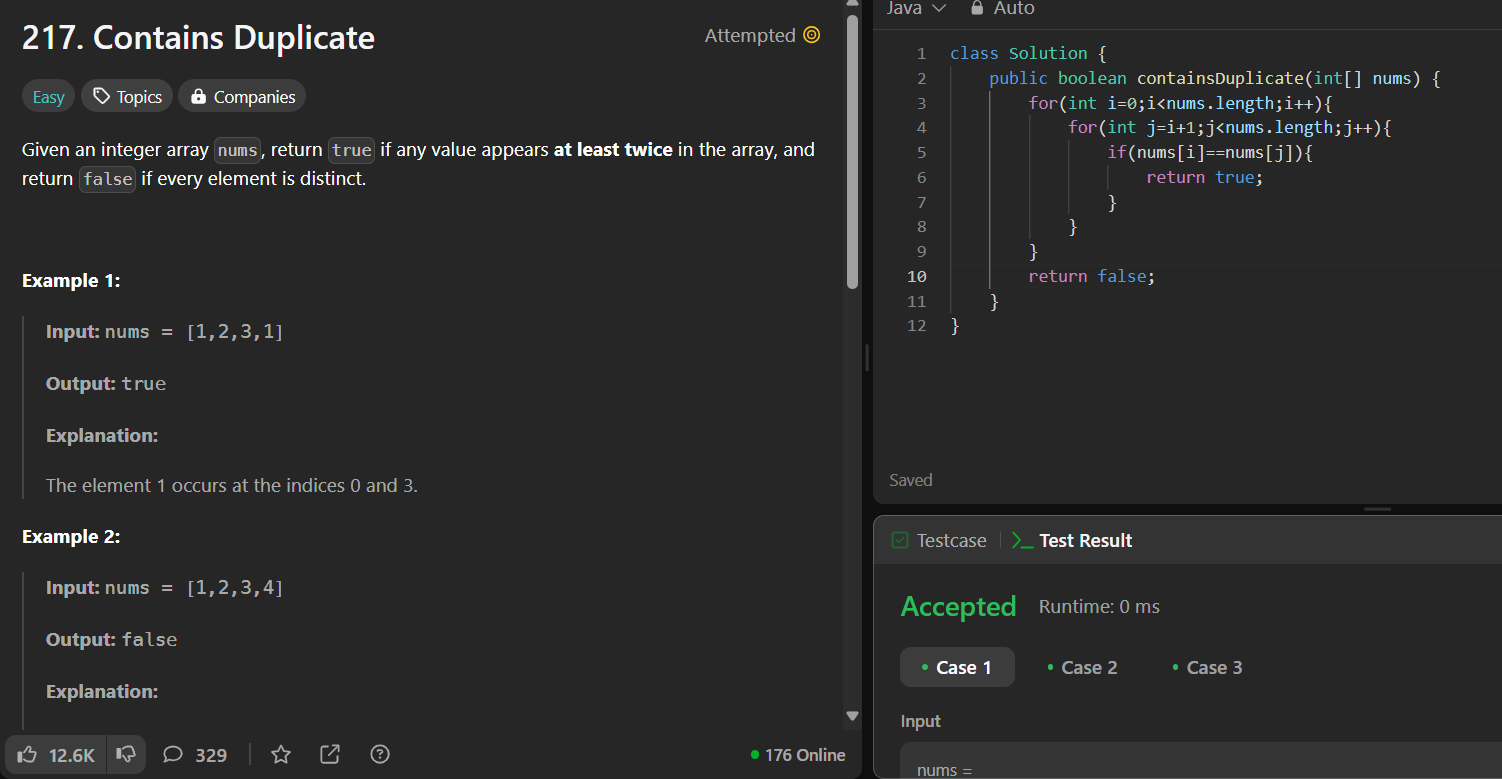
            }

        }

        return false;

    }

}



Two SumCW

<https://leetcode.com/problems/two-sum/>

Jump GameHW

<https://leetcode.com/problems/jump-game/>

Majority ElementHW

<https://leetcode.com/problems/majority-element>

class Solution {

    public int majorityElement(int[] nums) {

        int count=0;

        int cell=nums[0];

        for(int i=0;i<nums.length;i++){

            if(count==0){

                cell=nums[i];

                count++;

}else if(cell==nums[i]){

                count++;

            }

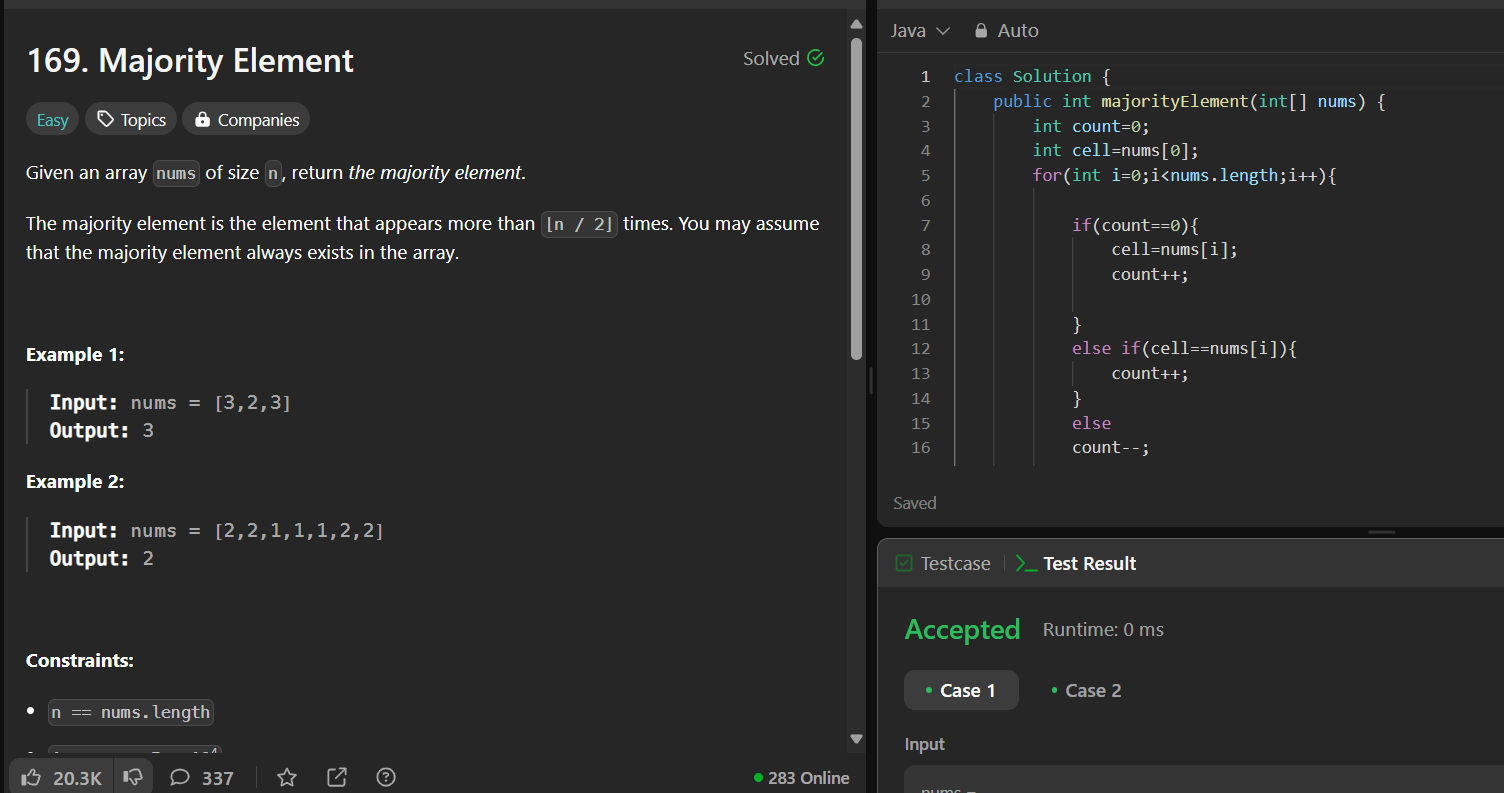
            Else

count--;

       }

        return cell;

        }}



Valid PalindromCW

<https://leetcode.com/problems/valid-palindrome/>

class Solution {

    public boolean isPalindrome(String s) {

        StringBuilder k = new StringBuilder();

        s = s.toLowerCase();

        for (int i = 0; i < s.length(); i++) {

            char ch = s.charAt(i);

            if  ((ch >= 'a' && ch <= 'z') || (ch >= '0' && ch <= '9')) {

                k.append(ch);

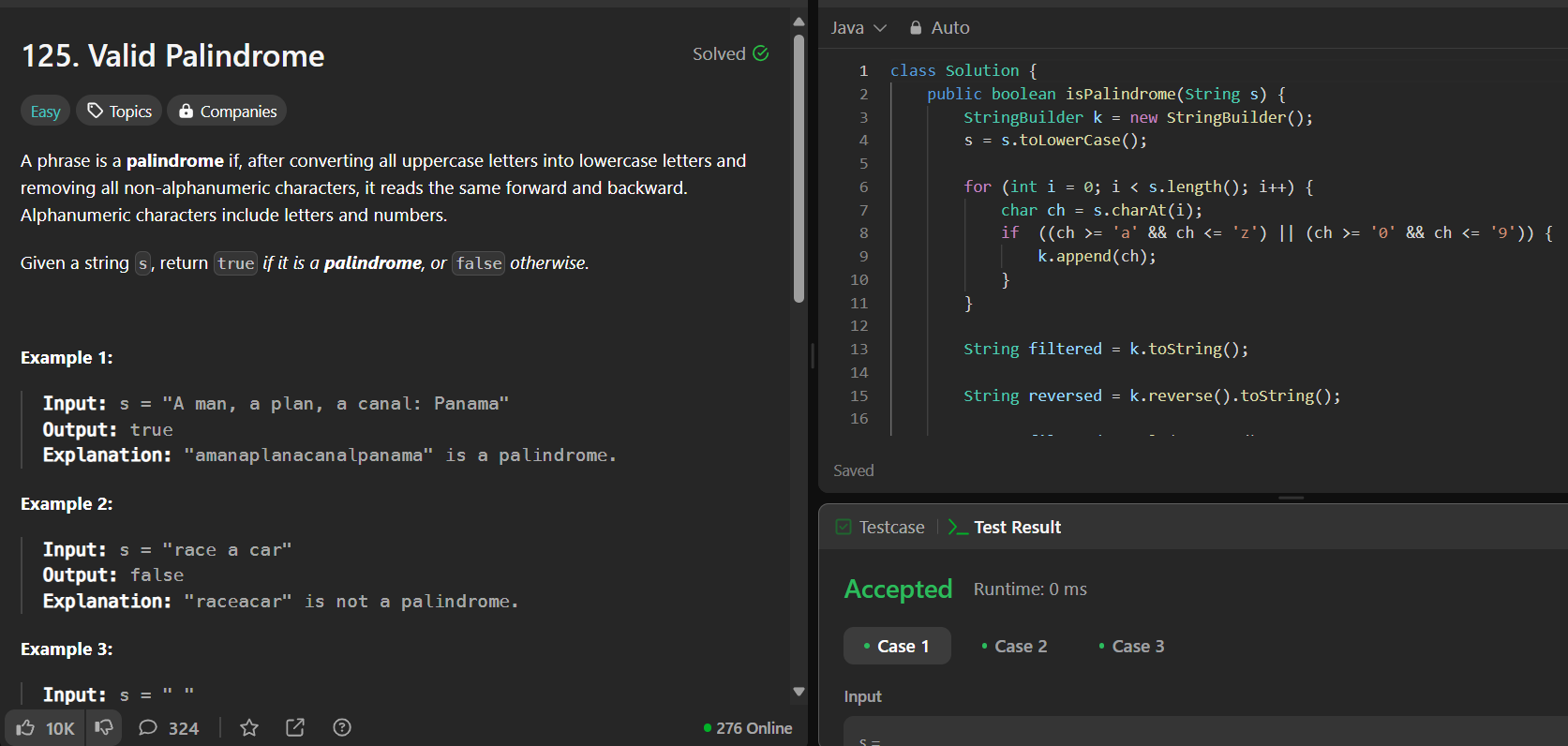
            } }

        String filtered = k.toString();

        String reversed = k.reverse().toString();

        return filtered.equals(reversed);

    }}



Jump Game 2HW

<https://leetcode.com/problems/jump-game-ii>

3SumHW

<https://leetcode.com/problems/3sum/>

Set Matrix zeroesCW

<https://leetcode.com/problems/set-matrix-zeroes/>

Longest substring withput repeating charactersHW

<https://leetcode.com/problems/longest-substring-without-repeating-characters/description/>

Finding duplicate numberHW

<https://leetcode.com/problems/find-the-duplicate-number/description/>

Print linked listCW

<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){

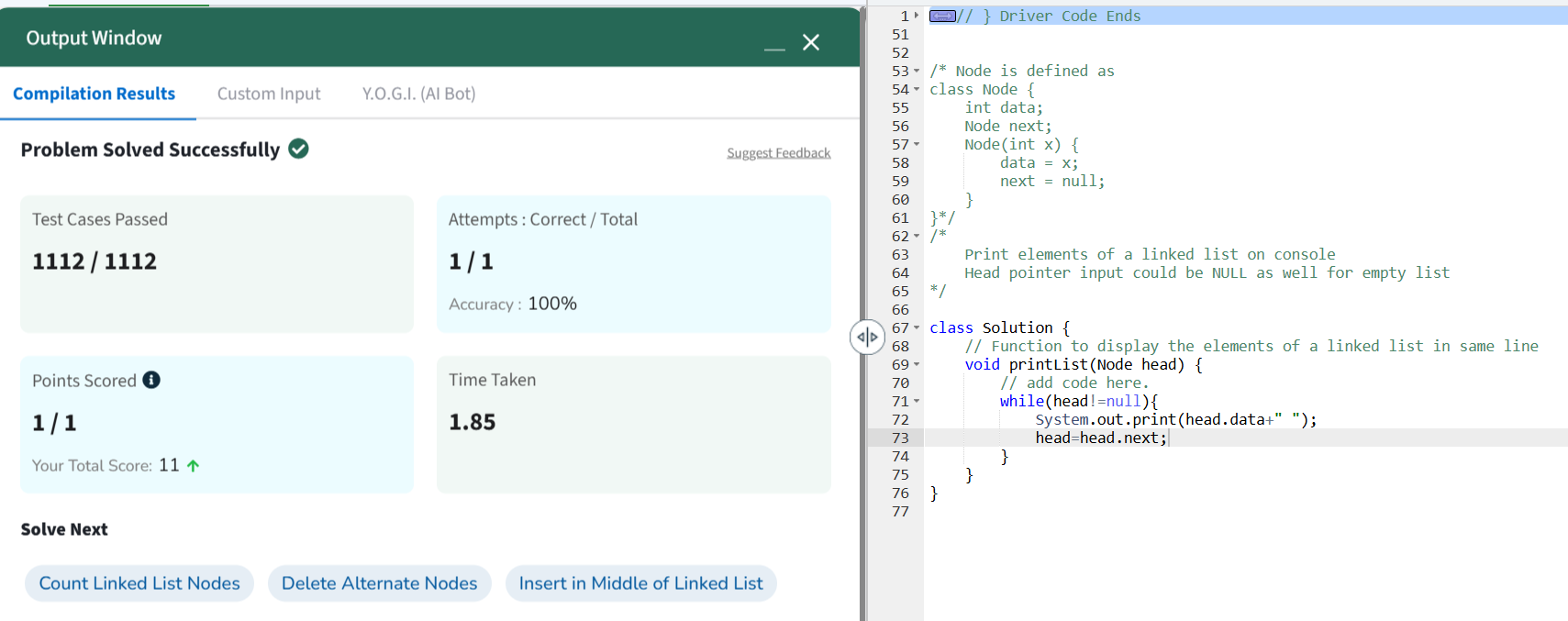
System.out.print(head.data+" ");

head=head.next;

}

}

}



Remove duplicates from a sorted listCW

<https://leetcode.com/problems/remove-duplicates-from-sorted-list>

Reverse a linked listCW

<https://leetcode.com/problems/reverse-linked-list/>

class Solution {

    public ListNode reverseList(ListNode head) {

        ListNode prev=null;

        ListNode curr=head;

        while(curr!=null){

        ListNode nextTemp = curr.next;

            curr.next = prev;

            prev = curr;

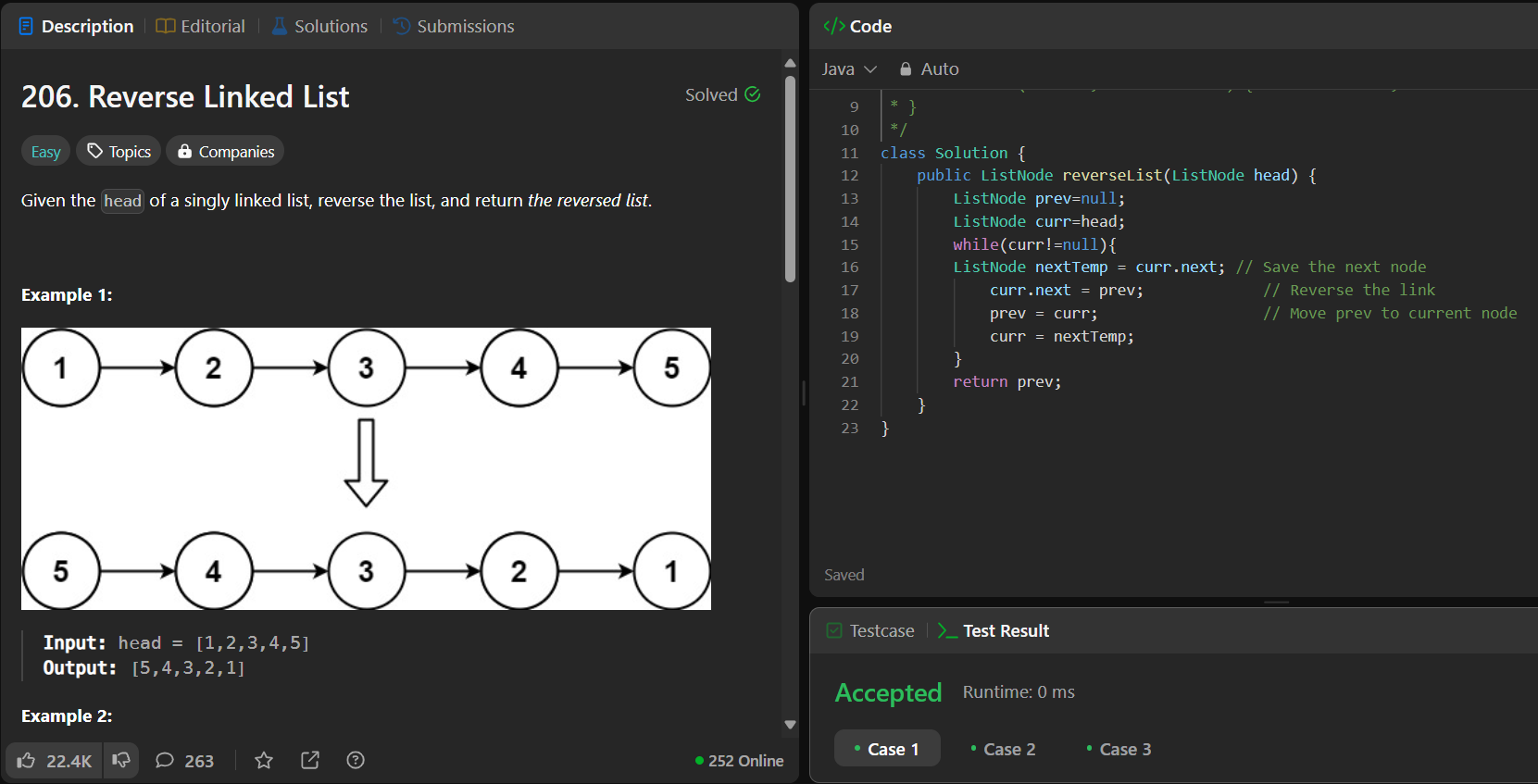
            curr = nextTemp;

        }

        return prev;

    }

}



Delete middle node of a listCW

<https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list>

class Solution {

    public ListNode deleteMiddle(ListNode head) {

        if(head.next==null){

            return head.next;

        }

        ListNode slow=new ListNode(-1);

        slow.next=head;

        ListNode fast=head;

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

            slow=slow.next;

            fast=fast.next.next;

        }

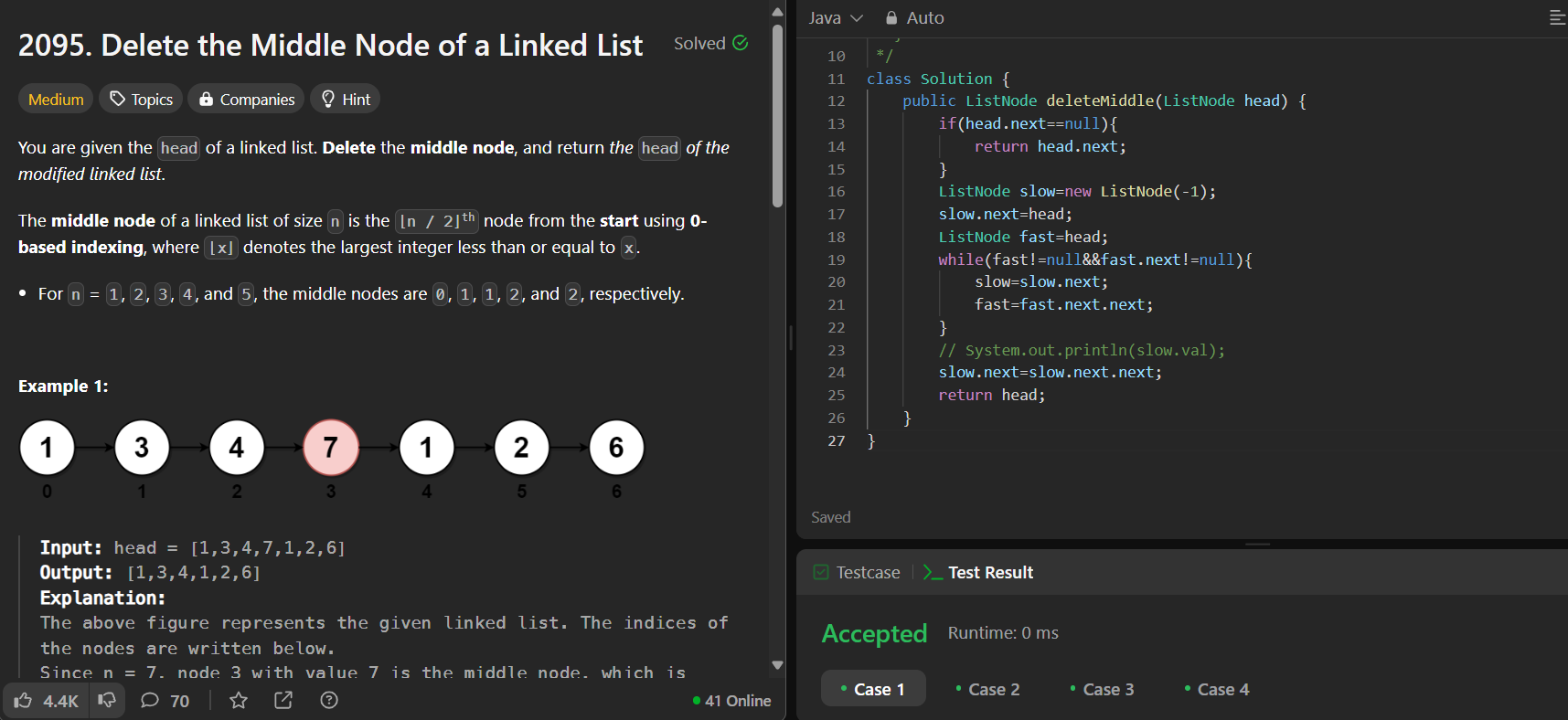
        // System.out.println(slow.val);

        slow.next=slow.next.next;

        return head;

    }

}



Merge two sorted linked listsCW

<https://leetcode.com/problems/merge-two-sorted-lists>

Remove duplicates from sorted lists 2CW

<https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii>

Detect a cycle in a linked listCW

<https://leetcode.com/problems/linked-list-cycle>

public class Solution {

    public boolean hasCycle(ListNode head) {

        ListNode slow=head;

        ListNode fast= head;

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

            slow=slow.next;

            fast=fast.next.next;

            if(slow==fast){

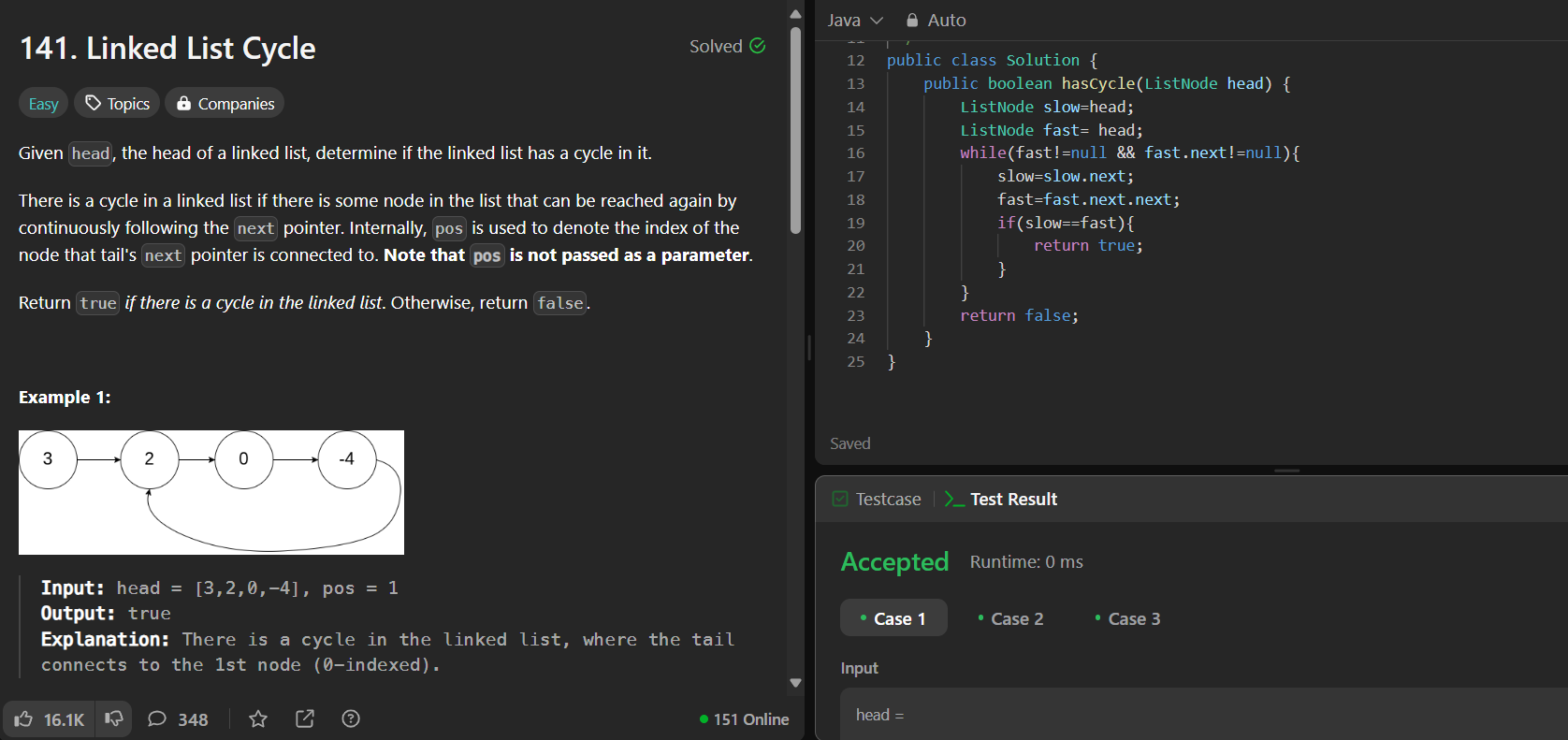
                return true;

            } }

        return false;

    }

}



Reverse linked list 2CW

<https://leetcode.com/problems/reverse-linked-list-ii>

class Solution {

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

        if (head == null) {

            return head;

        }

        ListNode dummy = new ListNode(-1);

        dummy.next = head;

        ListNode leftprev = dummy;

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

            leftprev = leftprev.next;

        }

        ListNode curr = leftprev.next;

        ListNode prev = null;

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

            ListNode temp = curr.next;

            curr.next = prev;

            prev = curr;

            curr = temp;

        }

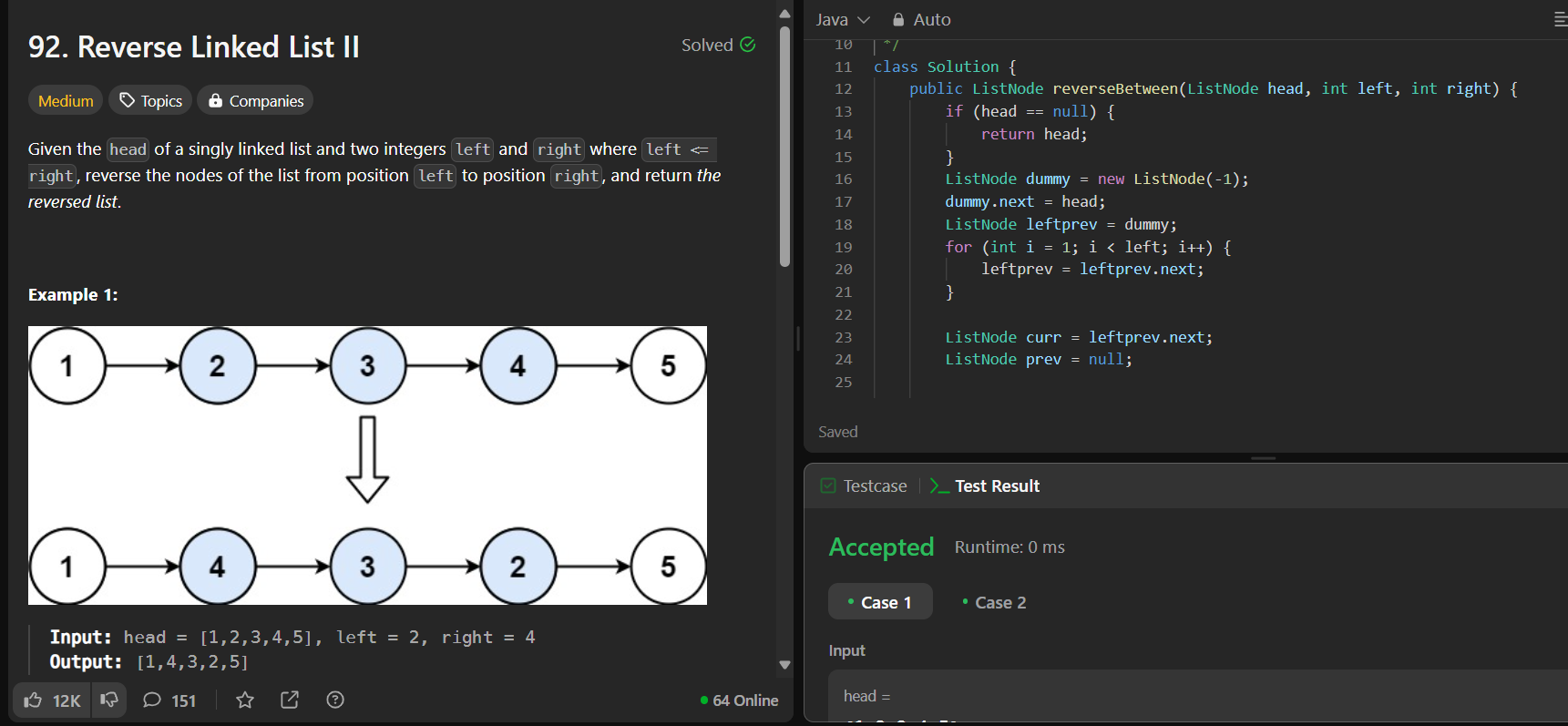
        leftprev.next.next = curr;

        leftprev.next = prev;

        return dummy.next;

    }

}



rotate a listCW

<https://leetcode.com/problems/rotate-list>

class Solution {

    public ListNode rotateRight(ListNode head, int k) {

        if (head == null) {

            return null;

        }

        ListNode dummy = new ListNode(-1);

        dummy.next = head;

        ListNode curr = head;

        ListNode prev = dummy;

        int size = 0;

        while (curr != null) {

            prev = curr;

            curr = curr.next;

            size++;

        }

        prev.next = head;

        k = k % size;

        curr = head;

        while (size - k > 1) {

            curr = curr.next;

            size--;

        }

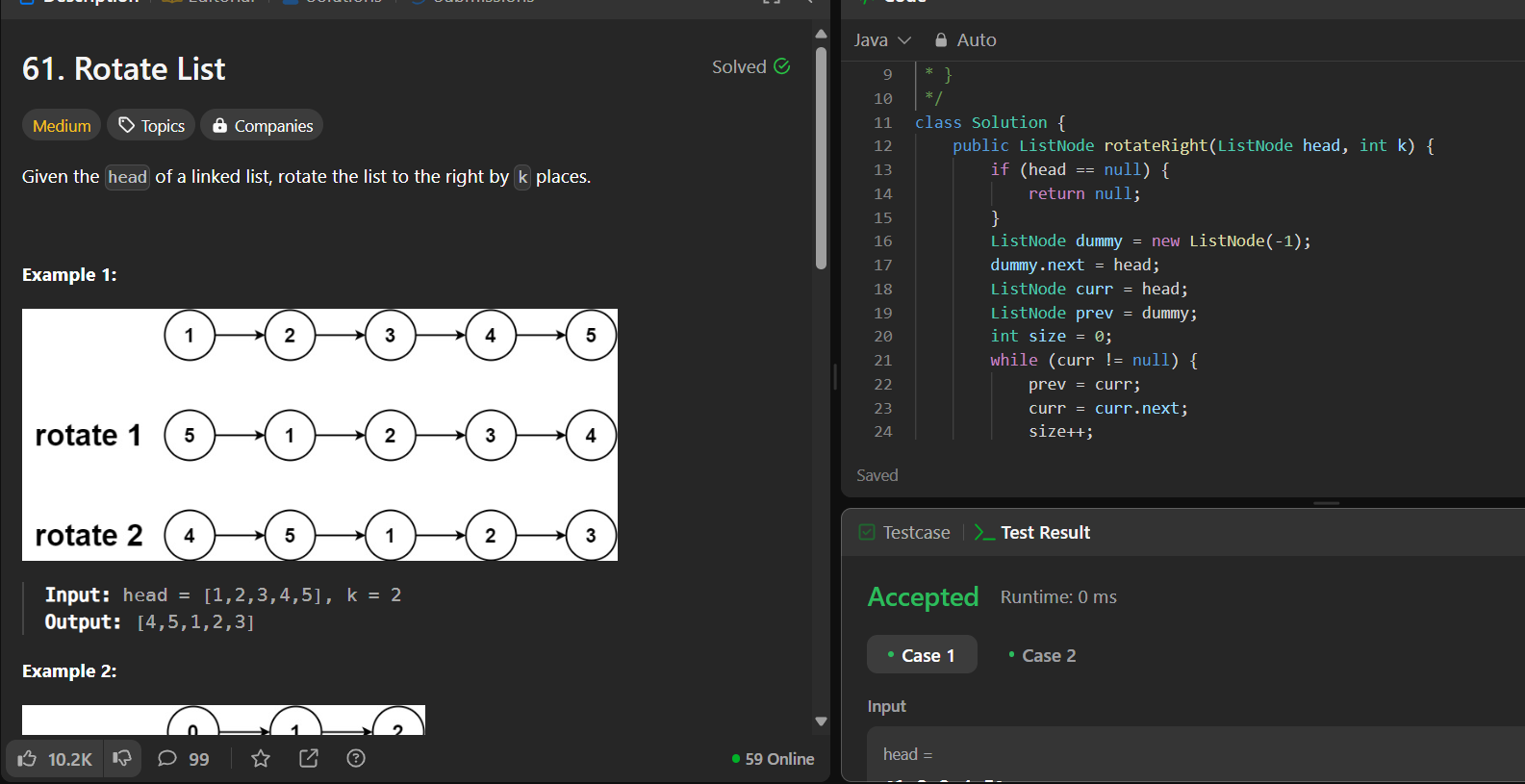
        dummy.next = curr.next;

        curr.next = null;

        return dummy.next;

    }

}



Merge k sorted listsCW

<https://leetcode.com/problems/merge-k-sorted-lists/>

Sort ListHW

<https://leetcode.com/problems/sort-list/description/>

Merge k sorted listsCW

<https://leetcode.com/problems/merge-k-sorted-lists/>