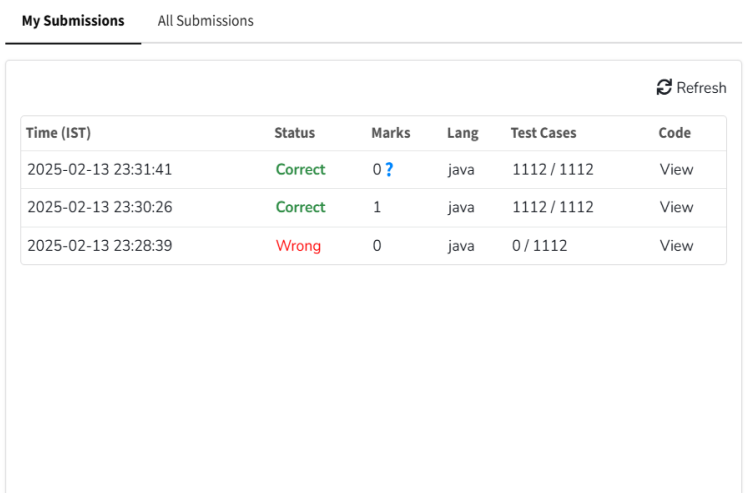
Assignment

Name – JIGYA JAIN
UID – 22BCS50101
Section – 609-A

1. <u>Print Linked List</u>Code –

```
class Solution {
    void printList(Node head) {
        while(head != null)
        {
            System.out.print(head.data + " ");
            head = head.next;
        }
    }
}
```

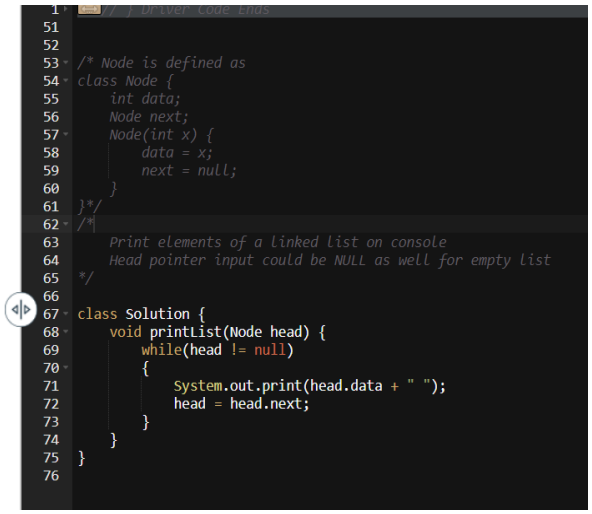
Submission –



My Submissions | All Submissions

Refresh

Time (IST)	Status	Marks	Lang	Test Cases	Code
2025-02-13 23:31:41	Correct	0 ?	java	1112 / 1112	View
2025-02-13 23:30:26	Correct	1	java	1112 / 1112	View
2025-02-13 23:28:39	Wrong	0	java	0 / 1112	View



```
/* Node is defined as
class Node {
    int data;
    Node next;
    Node(int x) {
        data = x;
        next = null;
    }
}*/
/*
    Print elements of a linked list on console
    Head pointer input could be NULL as well for empty list
*/

class Solution {
    void printList(Node head) {
        while(head != null)
        {
            System.out.print(head.data + " ");
            head = head.next;
        }
    }
}
```

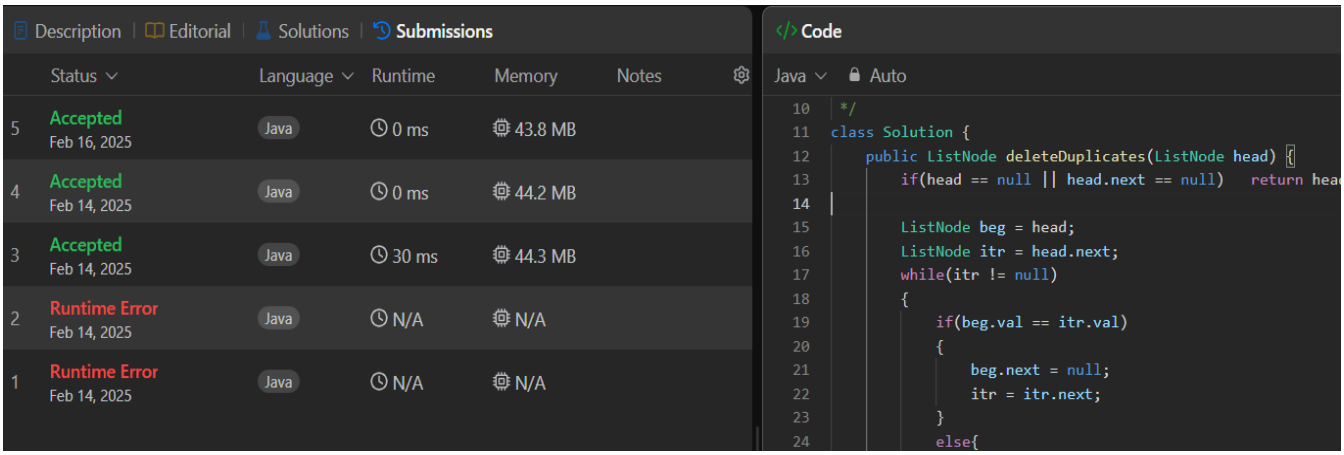
2. <u>Remove duplicates from a sorted list</u> –

Code –

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if(head == null || head.next == null)   return head;
        ListNode beg = head;
        ListNode itr = head.next;
        while(itr != null)
        {
            if(beg.val == itr.val)
            {
```

```
            beg.next = null;
            itr = itr.next;
        }
        else{
            beg.next = itr;
            beg = itr;
            itr = itr.next;
        }
    }
    return head;
  }
}
```

Submission –



3. <u>Reverse a linked list</u> –

Code –

```
class Solution {
    public ListNode reverseList(ListNode head) {
        if(head == null || head.next == null)  return head;
        ListNode ans = reverseList(head.next);
        head.next.next = head;
        head.next = null;
        return ans;
    }
}
```

Submission –

	Status	Language	Runtime	Memory	Notes
3	Accepted Feb 14, 2025	Java	0 ms	42.4 MB	
2	Accepted Feb 14, 2025	Java	0 ms	42.6 MB	
1	Compile Error Feb 14, 2025	Java	N/A	N/A	+ Notes

Java · Auto

```
* }
*/

class Solution {
    public ListNode reverseList(ListNode head) {
        if(head == null || head.next == null) return head;
        ListNode ans = reverseList(head.next);
        head.next.next = head;
        head.next = null;

        return ans;
    }
}
```

4. <u>Delete middle node of a list</u> –

Code –

```
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        if(head.next == null) return null;
        ListNode mid = head;
        ListNode temp = head;
        ListNode ans = null;
        while(temp != null && temp.next != null)
        {
            ans = mid;
            mid = mid.next;
            temp = temp.next.next;
        }
        ans.next = mid.next;
        return head;
    }
}
```

Submission –

	Status	Language	Runtime	Memory	Notes
6	Accepted Feb 20, 2025	Java	3 ms	63.7 MB	
5	Runtime Error Feb 20, 2025	Java	N/A	N/A	
4	Accepted Feb 20, 2025	Java	3 ms	65.5 MB	
3	Accepted Feb 20, 2025	Java	3 ms	64.3 MB	
2	Accepted Feb 20, 2025	Java	3 ms	64.4 MB	
1	Accepted Feb 18, 2025	Java	3 ms	63 MB	

Java · Auto

```
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        if(head.next == null) return null;

        ListNode mid = head;
        ListNode temp = head;
        ListNode ans = null;

        while(temp != null && temp.next != null)
        {
            ans = mid;
            mid = mid.next;
            temp = temp.next.next;
        }
        ans.next = mid.next;
        return head;
    }
```

5. <u>Merge two sorted linked lists</u> –

Code –

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode l1 = list1;
        ListNode l2 = list2;
        ListNode res = new ListNode(0);
        ListNode temp = res;

        while(l1 != null && l2 != null )
        {
            if(l1.val < l2.val)
            {
                temp.next = l1;
                l1 = l1.next;
            }
            else
            {
                temp.next = l2;
                l2 = l2.next;
            }
            temp = temp.next;
        }
        temp.next = (l1 != null ) ? l1 : l2;
        return res.next;
    }
}
```

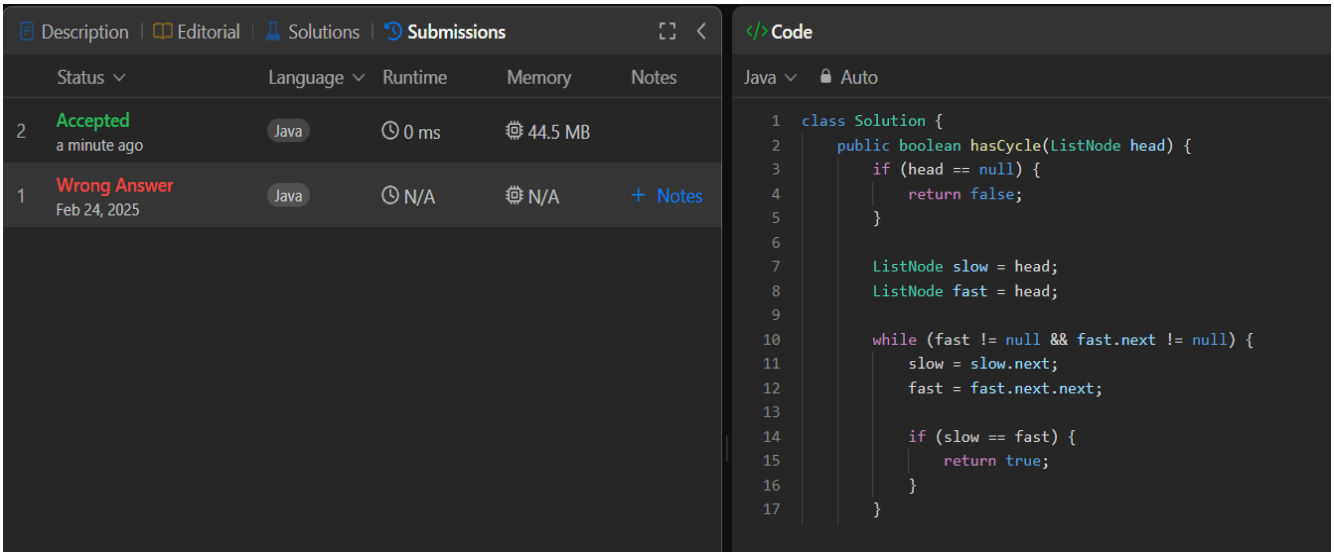
Submission –

6. Detect a cycle in a linked list –

Code –

```
class Solution {
    public boolean hasCycle(ListNode head) {
        if (head == null) {
            return false;
        }
        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;
    }
}
```

Submission –



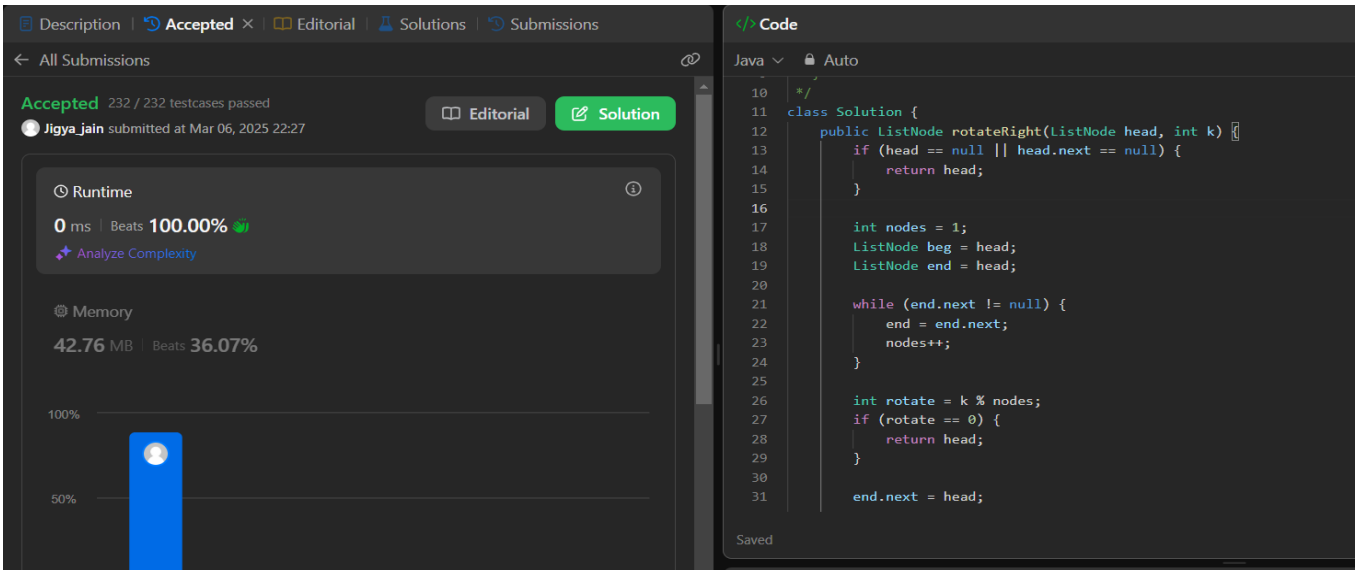
7. Rotate a list –

Code –

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null) {
            return head;
```

```
        }

        int nodes = 1;
        ListNode beg = head;
        ListNode end = head;

        while (end.next != null) {
            end = end.next;
            nodes++;
        }

        int rotate = k % nodes;
        if (rotate == 0) {
            return head;
        }
        end.next = head;

        ListNode newTail = head;
        for (int i = 0; i < nodes - rotate - 1; i++) {
            newTail = newTail.next;
        }
        ListNode newHead = newTail.next;
        newTail.next = null;
        return newHead;
    }
}
```

Submission –

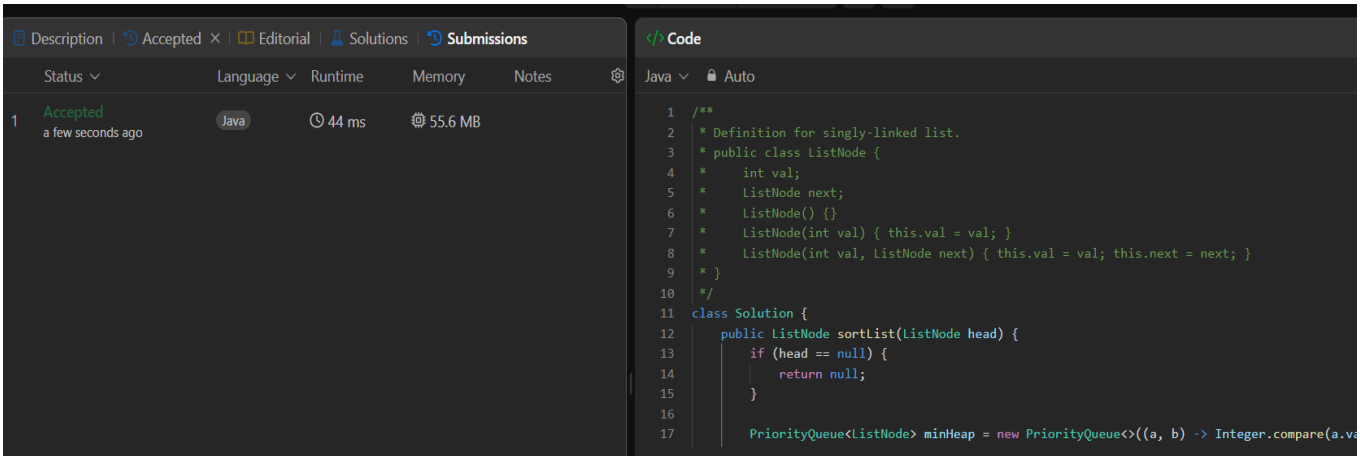


8. <u>Sort List</u> –

Code –

```
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null) {
            return null;
        }

        PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b) ->
Integer.compare(a.val, b.val));

        while (head != null) {
            minHeap.add(head);
            head = head.next;
        }

        ListNode dummy = new ListNode(0);
        ListNode temp = dummy;

        while (!minHeap.isEmpty()) {
            temp.next = minHeap.poll();
            temp = temp.next;
        }

        temp.next = null; // Ensure the last node points to null
        return dummy.next;
    }
}
```

Submission –



9. <u>Merge k sorted lists</u> –

Code –

```
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        PriorityQueue<int[]> minHeap = new PriorityQueue<>((a, b) ->
Integer.compare(a[0], b[0]));
        ListNode head = null, temp = null;

        for (int i = 0; i < lists.length; i++) {
            if (lists[i] != null) {
                minHeap.add(new int[]{lists[i].val, i});
            }
        }

        while (!minHeap.isEmpty()) {
            int[] smallest = minHeap.poll();
            int i = smallest[1];

            if (head == null) {
                head = lists[i];
                lists[i] = lists[i].next;
                temp = head;
            } else {
                temp.next = lists[i];
                temp = temp.next;
                lists[i] = lists[i].next;
            }

            if (lists[i] != null) {
                minHeap.add(new int[]{lists[i].val, i});
            }
        }

        return head;
    }
}
```

Submission –

Description | Accepted | Editorial | Solutions | **Submissions**

	Status	Language	Runtime	Memory	Notes
1	Accepted a few seconds ago	Java	5 ms	44.6 MB	

Code

Java | Auto

```java
*     ListNode(int val, ListNode next) { this.val = val; this.nex
* }
*/
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        PriorityQueue<int[]> minHeap = new PriorityQueue<>((a, b)
        ListNode head = null, temp = null;

        for (int i = 0; i < lists.length; i++) {
            if (lists[i] != null) {
                minHeap.add(new int[]{lists[i].val, i});
            }
        }

        while (!minHeap.isEmpty()) {
            int[] smallest = minHeap.poll();
            int i = smallest[1];
```