

Name: Aanchal

Negi UID: 22BCS14969

Section/Group: 609(B)

➤ Print Linked List

Code:

```
class Solution {  
  
    // Function to display the elements of a linked list in the same line  
  
    void printList(Node head) {  
  
        // Iterate through the linked list and print each element  
  
        Node current = head;  
  
        while (current != null) {  
  
            System.out.print(current.data + " ");  
  
            current = current.next;  
  
        }  
  
    }  
  
}
```

Output:

The screenshot displays a coding platform interface with a dark theme. At the top, navigation links include 'Courses', 'Tutorials', 'Jobs', 'Practice', and 'Contests'. The main header shows 'Java (1.8)' and a 'Start Timer' button. The left sidebar contains an 'Output Window' and 'Compilation Results' for 'Custom Input' by 'Y.O.G.I. (AI Bot)'. A green checkmark indicates 'Problem Solved Successfully'. The results section shows 'Test Cases Passed: 1112 / 1112', 'Attempts: 2 / 3', and 'Accuracy: 66%'. The 'Time Taken' is '1.67'. The right pane shows the code editor with the provided Java code. At the bottom, there are buttons for 'Custom Input', 'Compile & Run', and 'Submit'.

Test Cases Passed	Attempts : Correct / Total	Accuracy	Time Taken
1112 / 1112	2 / 3	66%	1.67

➤ Remove duplicates from a sorted list

Code:

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

        ListNode current = head;

        while (current != null && current.next != null) {
            if (current.val == current.next.val)
            {
                current.next = current.next.next;
            }
            else
            {
                current = current.next;
            }
        }

        return head;
    }
}
```

Output:

The screenshot displays a code editor interface with a dark theme. On the left, a sidebar shows the 'Problem List' and 'All Submissions' tabs. The 'Accepted' status is highlighted, indicating 168 out of 168 testcases passed. The submission was made by 'sapphir...' on Mar 05, 2025 at 15:13. Below this, performance metrics are shown: Runtime is 0 ms (Beats 100.00%) and Memory is 44.10 MB (Beats 72.14%). A blue progress bar is visible at the bottom of the sidebar. The main editor area shows the Java code for the 'deleteDuplicates' method. The code is as follows:

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

        ListNode current = head;

        while (current != null && current.next != null) {
            if (current.val == current.next.val)
            {
                current.next = current.next.next;
            }
            else
            {
                current = current.next;
            }
        }

        return head;
    }
}
```

At the bottom of the editor, there is a 'Testcase' section with a 'Run' button and a 'Submit' button. The 'Testcase' section also shows 'Case 1' and 'Case 2' tabs, with 'Case 1' currently selected. The input for 'Case 1' is 'head = '.

➤ Reverse Linked List

Code:

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;

        while (current != null) {
            ListNode nextNode = current.next;
            current.next = prev;
            prev = current;
            current = nextNode;
        }

        return prev;
    }
}
```

Output:

The screenshot displays a code editor interface for a problem titled "Reverse Linked List". The left sidebar shows the "Accepted" status with 28/28 testcases passed, submitted by "sapphir..." on Mar 05, 2025. The "Runtime" is 0 ms (Beats 100.00%) and "Memory" is 42.67 MB (Beats 38.88%). The main editor shows the Java code for the solution. The bottom right panel shows the "Testcase" and "Test Result" section with the input "head = [1,2,3,4,5]".

Code:

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;

        while (current != null) {
            ListNode nextNode = current.next;
            current.next = prev;
            prev = current;
            current = nextNode;
        }

        return prev;
    }
}
```

Testcase:

Input: head = [1,2,3,4,5]

➤ Delete middle node of a list

Code:

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

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

        while (fast != null && fast.next != null) {
            prev = slow;
            slow = slow.next;
            fast = fast.next.next;
        }

        prev.next = slow.next;

        return head;
    }
}
```

Output:

The screenshot displays a code editor interface for a problem titled "Delete middle node of a list". The code is written in Java and is accepted, with a runtime of 3 ms and memory usage of 65.98 MB. The test result shows the expected output [2].

Code:

```
while (fast != null && fast.next != null)
{
    prev = slow;
    slow = slow.next;
    fast = fast.next.next;
}

prev.next = slow.next;
```

Testcase:

Expected: [2]

➤ Merge Two Sorted Lists

Code:

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode dummy = new ListNode();
        ListNode current = dummy;

        while (list1 != null && list2 != null) {
            if (list1.val <= list2.val) {
                current.next = list1;
                list1 = list1.next;
            }
            else {
                current.next = list2;
                list2 = list2.next;
            }
            current = current.next;
        }

        if (list1 != null) {
            current.next = list1;
        }
        else {
            current.next = list2;
        }

        return dummy.next;
    }
}
```

The screenshot displays a code editor interface with a dark theme. On the left, a sidebar shows the 'Problem List' and 'All Submissions' for the 'Merge Two Sorted Lists' problem. The submission status is 'Accepted' (208 / 208 testcases passed) by user 'sapphir...' on Mar 05, 2025. Performance metrics show 0 ms runtime and 42.62 MB memory usage, both beating 100.00% of other submissions. The main editor area shows the Java code for the solution, which is a class 'Solution' with a method 'mergeTwoLists'. The code uses a dummy node and a current pointer to merge two sorted linked lists. The right sidebar shows the 'Testcase' results, indicating the solution is 'Accepted' with a runtime of 0 ms. The bottom of the right sidebar shows the 'Input' field for the testcases.

➤ Detect a cycle in a linked list

Code:

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        if (head == null || head.next == null) {
            return false; // No cycle if list is empty or has only one node
        }

        ListNode slow = head;
        ListNode fast = head.next;

        while (slow != fast) {
            if (fast == null || fast.next == null) {
                return false; // No cycle if fast pointer reaches the end of the list
            }
            slow = slow.next;
            fast = fast.next.next;
        }

        return true; // Cycle detected if slow pointer meets fast pointer
    }
}
```

Output:

The screenshot displays a coding platform interface with the following components:

- Problem List:** A navigation bar at the top with icons for problem list, accepted solutions, editorials, solutions, and submissions.
- Accepted Solutions:** A section showing 'Accepted 29 / 29 testcases passed' and a 'Solution' button.
- Runtime:** A section showing '0 ms | Beats 100.00%' and a link to 'Analyze Complexity'.
- Memory:** A section showing '44.39 MB | Beats 75.93%'.
- Code Editor:** A section showing the Java code for the solution, with line numbers 14 to 24. The code is:


```
14 if (head == null || head.next == null) {
15     return false; // No cycle if list is empty or has only one node
16 }
17
18 ListNode slow = head;
19 ListNode fast = head.next;
20
21 while (slow != fast) {
22     if (fast == null || fast.next == null) {
23         return false; // No cycle if fast pointer reaches the end of the list
24     }
25     slow = slow.next;
26     fast = fast.next.next;
27 }
28
29 return true; // Cycle detected if slow pointer meets fast pointer
30 }
```
- Testcase:** A section showing 'Accepted Runtime: 0 ms' and a 'Test Result' button.
- Test Result:** A section showing 'Accepted Runtime: 0 ms' and a 'Test Result' button.

➤ Rotate a list

Code:

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null || k == 0) {
            return head;
        }

        // Calculate the length of the linked list
        ListNode current = head;
        int length = 1;
        while (current.next != null) {
            current = current.next;
            length = length + 1;
        }

        // Adjust k if it is greater than the length
        k = k % length;

        // If k is 0, no rotation is needed
        if (k == 0) {
            return head;
        }

        // Make the linked list circular
        current.next = head;

        // Find the new head and tail
        int stepsToNewHead = length - k;
        ListNode newTail = head;
        for (int i = 1; i < stepsToNewHead; i++) {
            newTail = newTail.next;
        }
        ListNode newHead = newTail.next;
        newTail.next = null;

        return newHead;
    }
}
```

The screenshot displays a code editor interface for a problem titled "Rotate a list". The code is written in Java and is shown in the "Code" tab. The code implements a solution to rotate a linked list to the right by k places. The code is as follows:

```
34 current.next = head;
35
36 // Find the new head and tail
37 int stepsToNewHead = length - k;
38 ListNode newTail = head;
39 for (int i = 1; i < stepsToNewHead; i++) {
40     newTail = newTail.next;
41 }
42 ListNode newHead = newTail.next;
43 newTail.next = null;
44
```

The code is saved and the "Run" button is visible. Below the code, the "Testcase" tab shows the test results. The test results indicate that the solution is "Accepted" with a runtime of 0 ms. The "Testcase" tab also shows the input for the test case.

On the left side of the editor, there is a "Problem List" tab. It shows the problem is "Accepted" with 232 / 232 testcases passed. The submission was made by "sapphir..." on Mar 05, 2025 at 16:01. Below this, there is a "Runtime" section showing 0 ms and 100.00% beats. There is also a "Memory" section showing 42.64 MB and 47.69% beats. A progress bar is visible at the bottom of the left panel, showing the current position in the problem list.

➤ Sort List

Code:

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

        ListNode left = head;
        ListNode mid = findMid(left);
        ListNode right = mid.next;
        mid.next = null;

        left = sortList(left);
        right = sortList(right);

        return merge(left, right);
    }

    private ListNode findMid(ListNode node) {
        ListNode slow = node;
        ListNode fast = node.next;

        while(fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }

        return slow;
    }

    private ListNode merge(ListNode left, ListNode right) {
        ListNode node = new ListNode();
        ListNode merged = node;
        while(left != null && right != null) {
            if(left.val < right.val) {
                node.next = left;
                left = left.next;
            } else {
                node.next = right;
                right = right.next;
            }
            node = node.next;
        }

        if(left != null) {
            node.next = left;
            node = node.next;
        }

        if(right != null) {
            node.next = right;
            node = node.next;
        }
    }
}
```

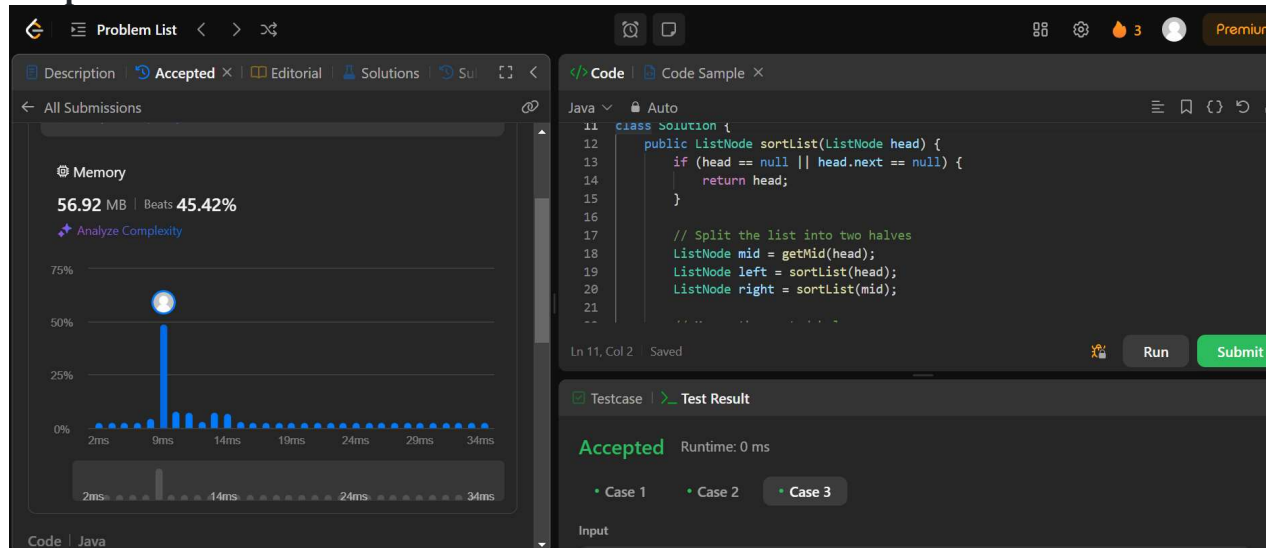


```

        return merged.next;
    }
}

```

Output:



➤ Merge k sorted lists :

Code:

```

class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        if (lists == null || lists.length == 0) {
            return null;
        }

        // Create a min-heap (priority queue) to store the nodes
        PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b) -> a.val - b.val);

        // Add the head of each list to the min-heap
        for (ListNode node : lists) {
            if (node != null) {
                minHeap.add(node);
            }
        }

        // Create a dummy node to build the merged list
        ListNode dummy = new ListNode(0);
        ListNode current = dummy;

        // Extract the smallest node from the min-heap and add it to the merged list
        while (!minHeap.isEmpty()) {
            ListNode smallest = minHeap.poll();
            current.next = smallest;
            current = current.next;

            // If the extracted node has a next node, add it to the min-heap

```

```

        if (smallest.next != null) {
            minHeap.add(smallest.next);
        }
    }

    return dummy.next;
}
}

```

Output:

The screenshot shows a LeetCode submission interface. On the left, the submission status is 'Accepted' with 134/134 testcases passed. The runtime is 4 ms, beating 71.10% of other submissions, and memory usage is 44.52 MB, beating 48.79%. The code is in Java and implements a min-heap solution for merging k sorted lists.

Code:

```

class ListNode {
    int val;
    ListNode next;
    ListNode() {}
    ListNode(int val) { this.val = val; }
    ListNode(int val, ListNode next) { this.val = val; this.next = next; }
}

class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        if (lists == null || lists.length == 0) {
            return null;
        }
        // ... (rest of the code)
    }
}

```

Testcase:

Case 1: lists = ...