ASSIGNMENT-3

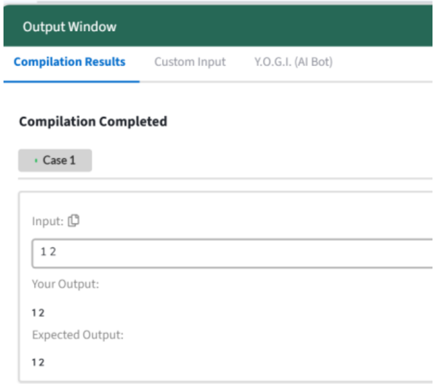
Student Name: Vaibhav Chhillar **UID:** 22BCS12585
Branch: CSE **Section/Group:** 22BCS_IOT-609/B
Semester: 6th **Subject Code:** 22CSP-351
Subject Name: Advanced Programming Lab-II

1. **Problem Statement :**
   Print Linked list
   https://www.geeksforgeeks.org/problems/print-linked-list-elements/0

   Code:

```
class Solution {
  public:
    void printList(Node *head) {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data;
            if (temp->next != nullptr) cout << " ";
            temp = temp->next;
        }
    }
};
```

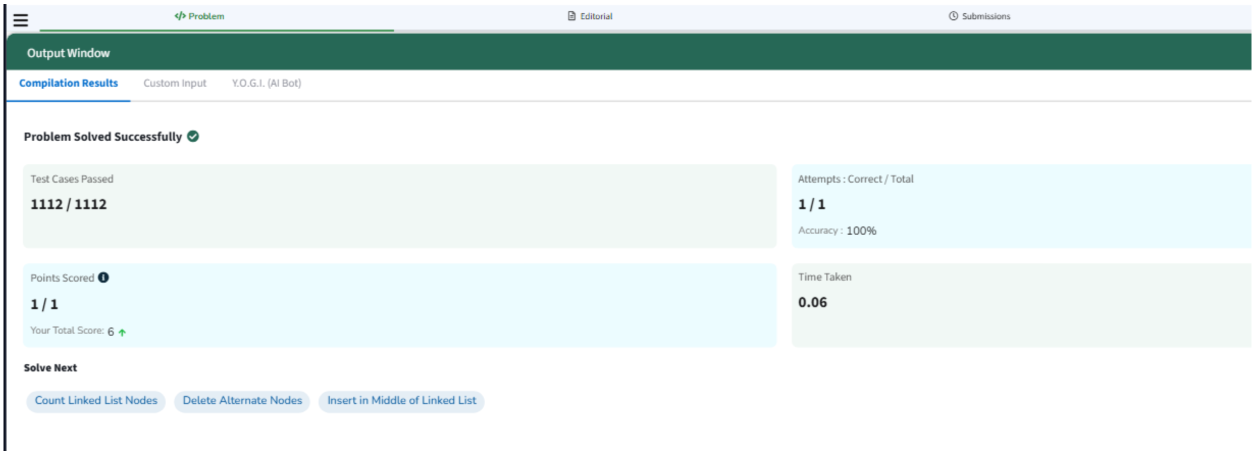
OUTPUT:



Compilation Completed

Case 1

```
Input:
1 2
Your Output:
1 2
Expected Output:
1 2
```



Problem Solved Successfully

Test Cases Passed
1112 / 1112

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
1 / 1
Your Total Score: 6

Time Taken
0.06

Solve Next

2. Problem Statement:

Remove duplicates from a sorted list

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

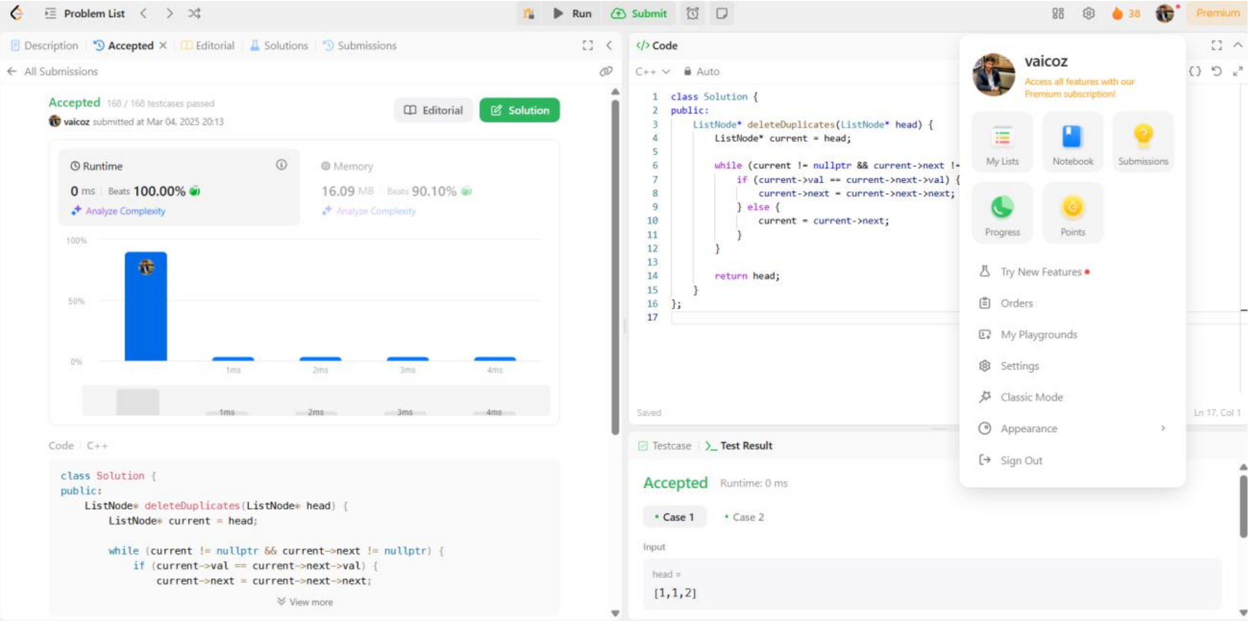
Code:

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;

        while (current != nullptr && current->next != nullptr) {
            if (current->val == current->next->val) {
                current->next = current->next->next;
            } else {
                current = current->next;
            }
        }


        return head;
    }
};
```

OUTPUT:



Accepted 168 / 168 testcases passed
vaicoz submitted at Mar 04, 2025 20:13

Runtime
0 ms | Beats 100.00%

Memory
16.09 MB | Beats 90.10%

Code | C++

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;

        while (current != nullptr && current->next != nullptr) {
            if (current->val == current->next->val) {
                current->next = current->next->next;
```

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;

        while (current != nullptr && current->next !=
            if (current->val == current->next->val) {
                current->next = current->next->next;
            } else {
                current = current->next;
            }
        }

        return head;
    }
};
```

vaicoz
Access all features with our Premium subscription!

Accepted Runtime: 0 ms

Input

head =
[1,1,2]

3. Problem Statement:

Reverse a linked list

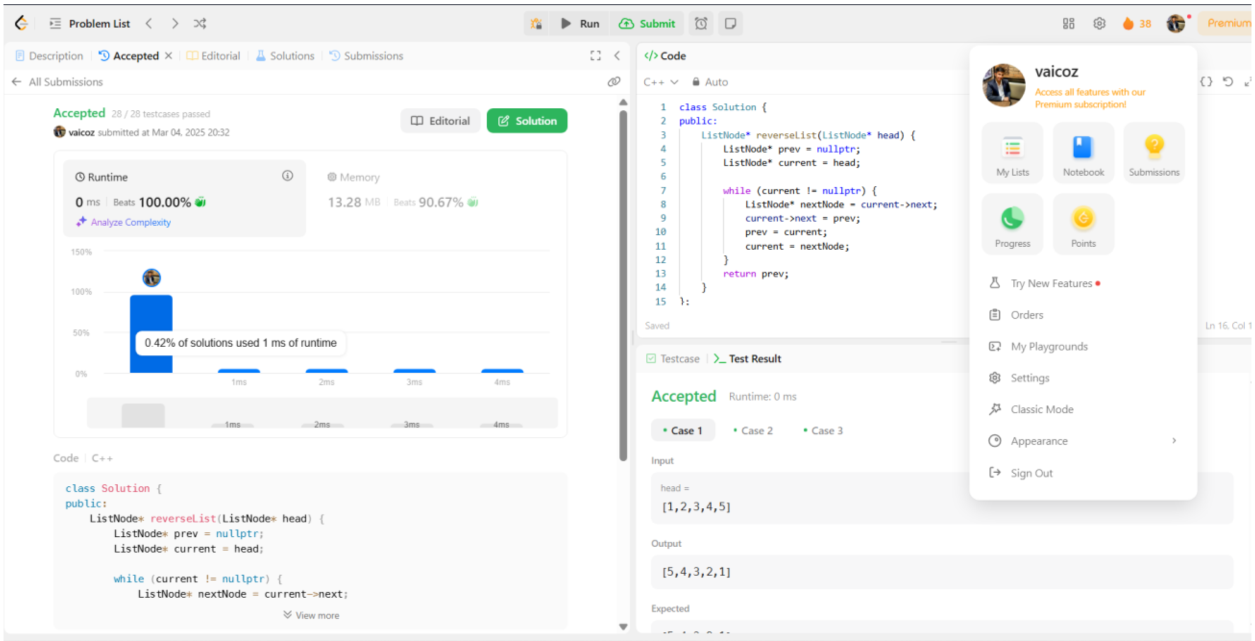
https://leetcode.com/problems/reverse-linked-list/description/

CODE:

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* current = head;

        while (current != nullptr) {
            ListNode* nextNode = current->next;
            current->next = prev;
            prev = current;
            current = nextNode;
        }
        return prev;
    }
};
```

OUTPUT:



Accepted 28 / 28 testcases passed

vaicoz submitted at Mar 04, 2025 20:32

Runtime 0 ms | Beats 100.00%

Memory 13.28 MB | Beats 90.67%

0.42% of solutions used 1 ms of runtime

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* current = head;

        while (current != nullptr) {
            ListNode* nextNode = current->next;
            current->next = prev;
            prev = current;
            current = nextNode;
        }
        return prev;
    }
};
```

Accepted Runtime: 0 ms

Input

```
head =
[1,2,3,4,5]
```

Output

```
[5,4,3,2,1]
```

Expected

4. **Problem Statement:**

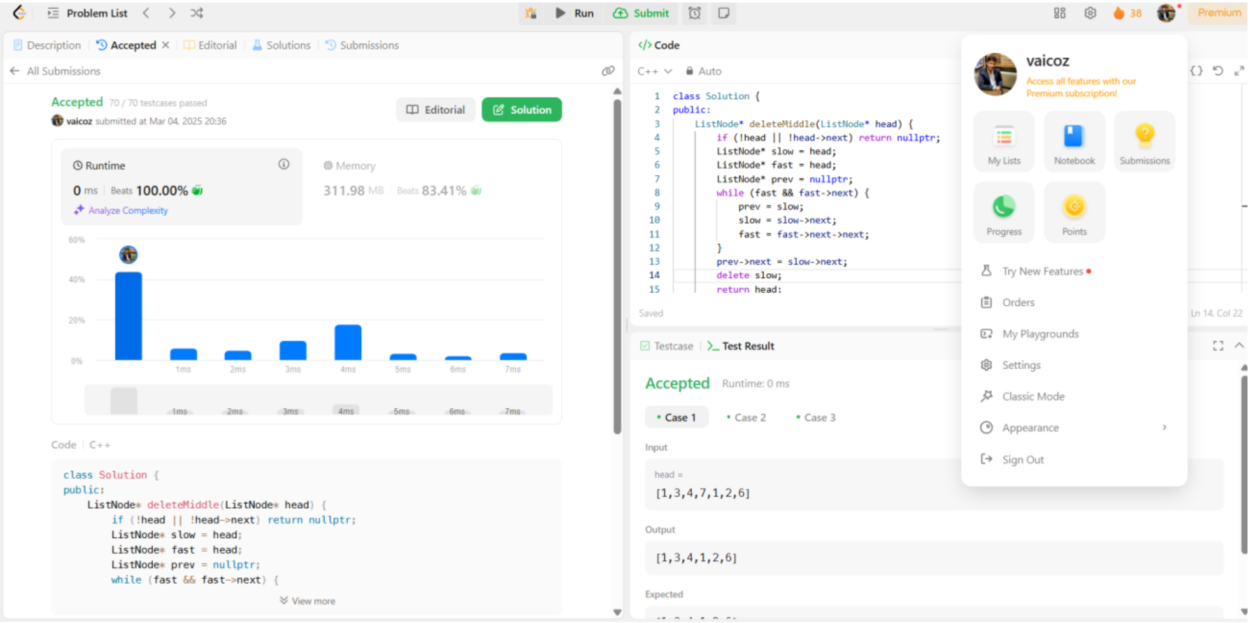
Delete middle node of a list:

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

CODE:

```
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if (!head || !head->next) return nullptr;
        ListNode* slow = head;
        ListNode* fast = head;
        ListNode* prev = nullptr;
        while (fast && fast->next) {
            prev = slow;
            slow = slow->next;
            fast = fast->next->next;
        }
        prev->next = slow->next;
        delete slow;
        return head;
    }
};
```

OUTPUT:



Accepted 70 / 70 testcases passed

vaicoz submitted at Mar 04, 2025 20:36

Runtime 0 ms | Beats 100.00%

Memory 311.98 MB | Beats 83.41%

```
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if (!head || !head->next) return nullptr;
        ListNode* slow = head;
        ListNode* fast = head;
        ListNode* prev = nullptr;
        while (fast && fast->next) {
            prev = slow;
            slow = slow->next;
            fast = fast->next->next;
        }
        prev->next = slow->next;
        delete slow;
        return head:
```

Accepted Runtime: 0 ms

Input

head =
[1,3,4,7,1,2,6]

Output

[1,3,4,1,2,6]

Expected

5. **Problem Statement:**

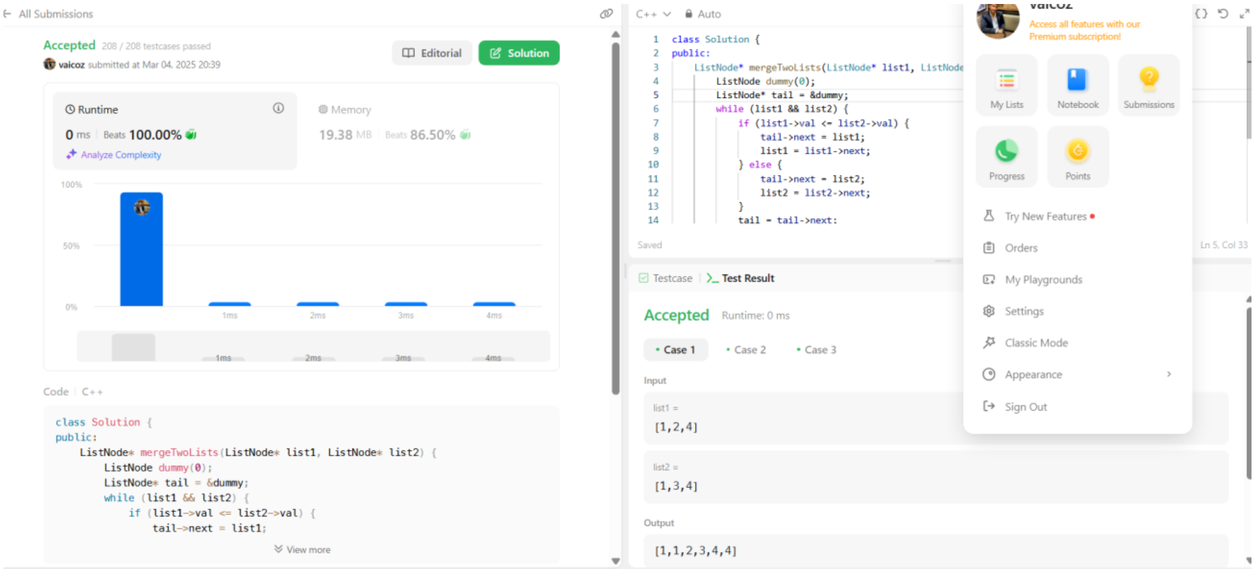
Merge two sorted linked lists:

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

CODE:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (list1 && list2) {
            if (list1->val <= list2->val) {
                tail->next = list1;
                list1 = list1->next;
            } else {
                tail->next = list2;
                list2 = list2->next;
            }
            tail = tail->next;
        }
        tail->next = list1 ? list1 : list2;
        return dummy.next;
    }
};
```

OUTPUT:



Accepted 208 / 208 testcases passed

vaicoz submitted at Mar 04, 2025 20:39

Runtime 0 ms | Beats 100.00%

Memory 19.38 MB | Beats 86.50%

Code | C++

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (list1 && list2) {
            if (list1->val <= list2->val) {
                tail->next = list1;
```

View more

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (list1 && list2) {
            if (list1->val <= list2->val) {
                tail->next = list1;
                list1 = list1->next;
            } else {
                tail->next = list2;
                list2 = list2->next;
            }
            tail = tail->next:
```

vaicoz

Access all features with our Premium subscription!

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

Output

```
[1,1,2,3,4,4]
```

6. Problem Statement:

Detect a cycle in a linked list:

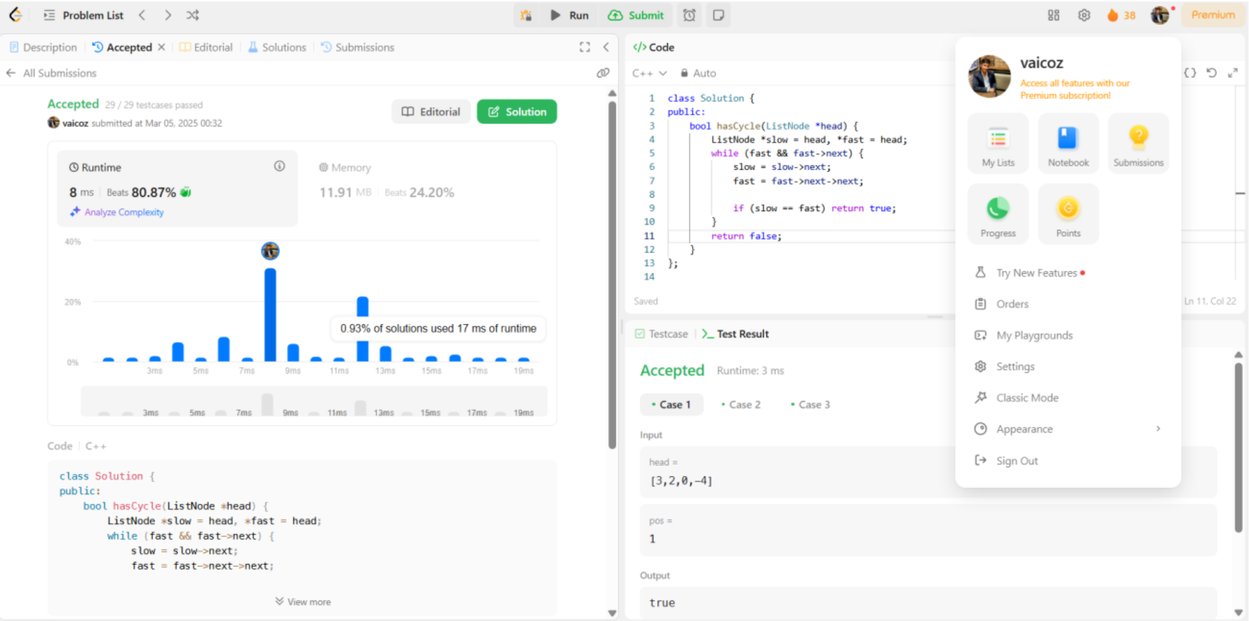
https//leetcode.com/problems/linked-list-cycle/description/

CODE:

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode *slow = head, *fast = head;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast) return true;
        }
        return false;
    }
};
```

OUTPUT:



Accepted 29 / 29 testcases passed

vaicoz submitted at Mar 05, 2025 00:32

Runtime 8 ms | Beats 80.87%

Memory 11.91 MB | Beats 24.20%

0.93% of solutions used 17 ms of runtime

Code | C++

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode *slow = head, *fast = head;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
```

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode *slow = head, *fast = head;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast) return true;
        }
        return false;
    }
};
```

Accepted Runtime: 3 ms

Input

head =
[3,2,0,-4]

pos =
1

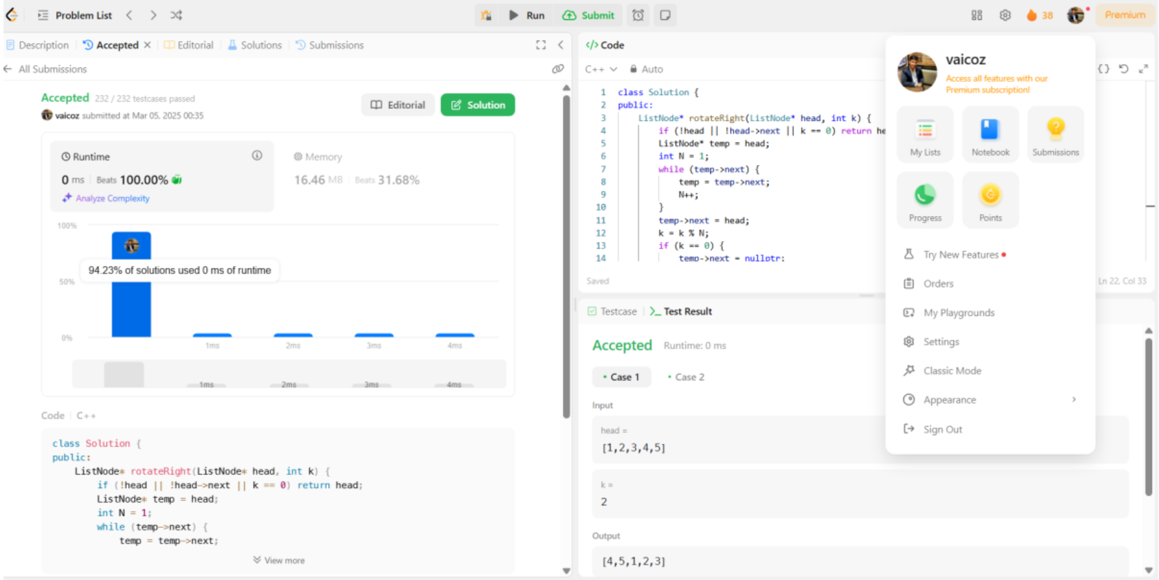
Output

true

7. **Problem Statement:**
Rotate a list:
https://leetcode.com/problems/rotate-list/description/
CODE:

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0) return head;
        ListNode* temp = head;
        int N = 1;
        while (temp->next) {
            temp = temp->next;
            N++;
        }
        temp->next = head;
        k = k % N;
        if (k == 0) {
            temp->next = nullptr;
            return head;
        }
        ListNode* newTail = head;
        for (int i = 0; i < N - k - 1; i++) {
            newTail = newTail->next;
        }
        ListNode* newHead = newTail->next;
        newTail->next = nullptr;
        return newHead;
    }
};
```

OUTPUT:



Accepted 232 / 232 testcases passed

vaicoz submitted at Mar 05, 2025 00:35

Runtime 0 ms | Beats 100.00%

Memory 16.46 MB | Beats 31.68%

94.23% of solutions used 0 ms of runtime

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0) return head;
        ListNode* temp = head;
        int N = 1;
        while (temp->next) {
            temp = temp->next;
```

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0) return he
        ListNode* temp = head;
        int N = 1;
        while (temp->next) {
            temp = temp->next;
            N++;
        }
        temp->next = head;
        k = k % N;
        if (k == 0) {
            temp->next = nullptr;
```

Accepted Runtime: 0 ms

Input

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

k =
2

Output

[4,5,1,2,3]

8. **Problem Statement:**

Sort List:

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

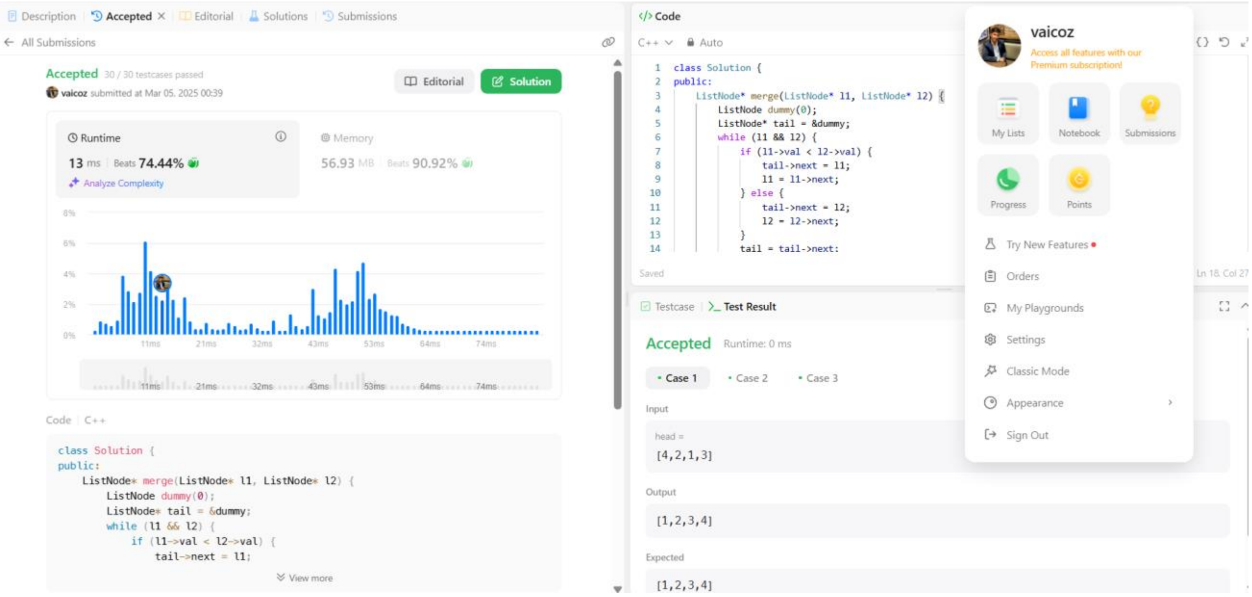
CODE:

```
class Solution {
public:
    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
                l1 = l1->next;
            } else {
                tail->next = l2;
                l2 = l2->next;
            }
            tail = tail->next;
        }

        tail->next = l1 ? l1 : l2;
        return dummy.next;
    }
    ListNode* getMid(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;
        ListNode* prev = nullptr;

        while (fast && fast->next) {
            prev = slow;
            slow = slow->next;
            fast = fast->next->next;
        }
        if (prev) prev->next = nullptr;
        return slow;
    }
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) return head;
        ListNode* mid = getMid(head);
        ListNode* left = sortList(head);
        ListNode* right = sortList(mid);
        return merge(left, right);
    }
};
```

OUTPUT:



Accepted 30 / 30 testcases passed

vaicoz submitted at Mar 05, 2025 00:39

Runtime 13 ms | Beats 74.44%

Memory 56.93 MB | Beats 90.92%

```
class Solution {
public:
    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
                l1 = l1->next;
            } else {
                tail->next = l2;
                l2 = l2->next;
            }
            tail = tail->next;
```

Accepted Runtime: 0 ms

Input

head =

[4,2,1,3]

Output

[1,2,3,4]

Expected

[1,2,3,4]

9. **Problem Statement:**

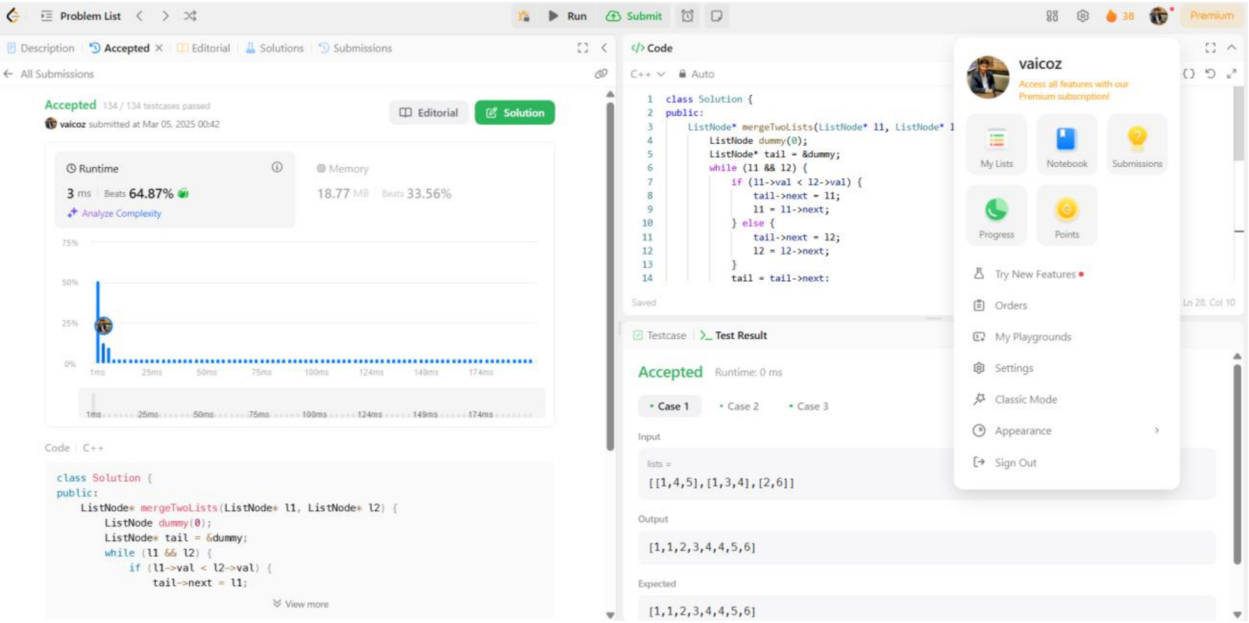
Merge K sorted List

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

CODE:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
                l1 = l1->next;
            } else {
                tail->next = l2;
                l2 = l2->next;
            }
            tail = tail->next;
        }
        tail->next = l1 ? l1 : l2;
        return dummy.next;
    }
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) return nullptr;
        int n = lists.size();
        while (n > 1) {
            int newSize = (n + 1) / 2;
            for (int i = 0; i < n / 2; i++) {
                lists[i] = mergeTwoLists(lists[i], lists[i + (n + 1) / 2]);
            }
            n = newSize;
        }
        return lists[0];
    }
};
```

OUTPUT:



Accepted 134 / 134 testcases passed

vaicoz submitted at Mar 05, 2025 00:42

Runtime: 3 ms | Beats 64.87%

Memory: 18.77 MB | Beats 33.56%

Code | C++

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
```

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l
        ListNode dummy(0);
        ListNode* tail = &dummy;
        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
                l1 = l1->next;
            } else {
                tail->next = l2;
                l2 = l2->next;
            }
            tail = tail->next:
```

Accepted Runtime: 0 ms

Input

lists =

```
[[1,4,5],[1,3,4],[2,6]]
```

Output

```
[1,1,2,3,4,4,5,6]
```

Expected

```
[1,1,2,3,4,4,5,6]
```