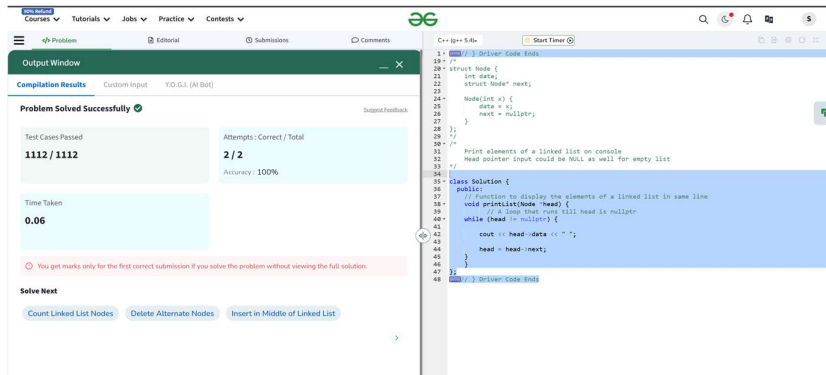


Q1. Print Linked List

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



Q2. 83. Remove Duplicates from Sorted List

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* res = head;
        while (head && head->next) {
            if (head->val == head->next->val) {
                head->next = head->next->next;
            } else {
                head = head->next;
            }
        }
        return res;
    }
};
```

Submission Details:

- Status: Accepted
- Testcases: 168 / 168 passed
- Runtime: 0 ms | Beats: 100.00%
- Memory: 16.16 MB | Beats: 67.88%

C++ Code:

```

7  * ListNode(int x) : val(x), next(nullptr) {}
8  * ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* deleteDuplicates(ListNode* head) {
14         ListNode* res = head;
15
16         while (head && head->next) {
17             if (head->val == head->next->val) {
18                 head->next = head->next->next;
19             } else {
20                 head = head->next;
21             }
22         }
23
24         return res;
25     }
26 };

```

Q3. 206. Reverse Linked List

```
class Solution {
```

```
public:
```

```
    ListNode* reverseList(ListNode* head) {
```

```
        if(head == NULL || head->next == NULL) return head;
```

```
        ListNode* prev = head;
```

```
        ListNode* curr = prev->next;
```

```
        head->next = NULL;
```

```
        while(prev != NULL && curr != NULL){
```

```
            ListNode* next = curr->next;
```

```
            curr->next = prev;prev = curr;
```

```
            curr = next;
```

```
        }
```

```
        return prev;
```

```
    }
```

```
};
```

The screenshot shows a LeetCode solution for the problem "Reverse a Linked List". The solution is in C++ and is marked as "Accepted". The runtime is 0 ms, and the memory usage is 13.32 MB, both of which are optimal (100.00% and 70.65% respectively). The code is as follows:

```

1 class Solution {
2 public:
3     ListNode* reverseList(ListNode* head) {
4         if(head == NULL || head->next == NULL) return head;
5         ListNode* prev = head;
6         ListNode* curr = prev->next;
7         head->next = NULL;
8         while(prev != NULL && curr != NULL){
9             ListNode* next = curr->next;
10            curr->next = prev; prev = curr;
11            curr = next;
12        }
13        return prev;
14    }
15 };

```

The test result shows that the solution is "Accepted" with a runtime of 0 ms. The input is not shown.

Q4. 2095. Delete the Middle Node of a Linked List

```
class Solution {
```

```
public:
```

```
    ListNode* deleteMiddle(ListNode* head) {
```

```
        if (!head || !head->next)
```

```
            return NULL;
```

```
        ListNode* ans = new ListNode(0);
```

```
        ans->next = head;
```

```
        ListNode* slow = ans;
```

```
        ListNode* fast = head;
```

```
        while (fast != NULL && fast->next != NULL) {
```

```
            slow = slow->next;
```

```
            fast = fast->next->next;
```

```
        }
```

```
        slow->next = slow->next->next;
```

```
        return ans->next;
```

```
    }
```

```
};
```

Submission Details:

- Status: Accepted
- Testcases: 70 / 70 passed
- Runtime: 4 ms | Beats 35.86%
- Memory: 312.13 MB | Beats 17.88%

Code (C++):

```

class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if (!head || !head->next)
            return NULL;
        ListNode* ans = new ListNode(0);
        ans->next = head;
        ListNode* slow = ans;
        ListNode* fast = head;
        while (fast != NULL && fast->next != NULL) {
            slow = slow->next;
            fast = fast->next->next;
        }
        slow->next = slow->next->next;
    }
};

```

Test Result:

- Accepted
- Runtime: 0 ms
- Case 1: [1,3,4,7,1,2,6]

Q5. 21. Merge Two Sorted Lists

```

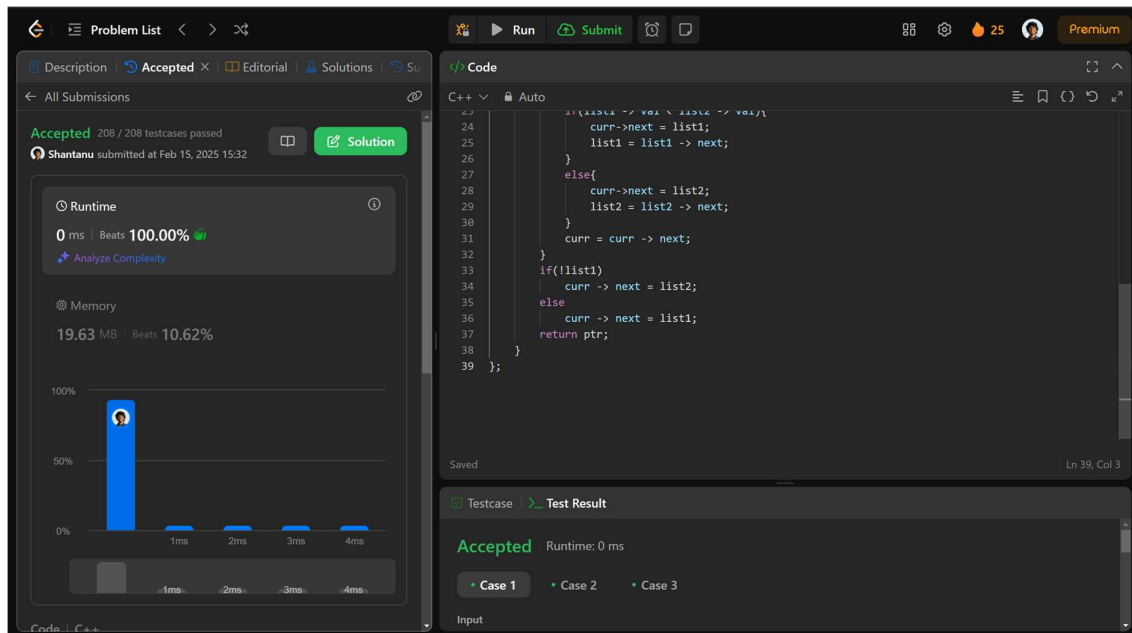
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if(list1 == NULL)
            return list2;
        if(list2 == NULL)
            return list1;
        ListNode * ptr = list1;
        if(list1 -> val > list2 -> val){
            ptr = list2;
            list2 = list2 -> next;
        }
        else{
            list1 = list1 -> next;
        }
        ListNode *curr = ptr;
        while(list1 && list2){
            if(list1 -> val < list2 -> val){
                curr->next = list1;
                list1 = list1 -> next;
            }
            else{
                curr->next = list2;
            }
        }
    }
};

```

```

        list2 = list2 -> next;
    }
    curr = curr -> next;
}
if(!list1)
    curr -> next = list2;
else
    curr -> next = list1;
return ptr;
}
};

```



Q6. [82. Remove Duplicates from Sorted List II](#)

```
class Solution {
```

```
public:
```

```
    ListNode* deleteDuplicates(ListNode* head) {
```

```
        // Edge case
```

```
        if (head == NULL || head->next == NULL) {
```

```
            return head;
```

```
        }
```

```
        ListNode* dummy = new ListNode(0, head);
```

```
        ListNode* prev = dummy;
```

```
        ListNode* temp = head;
```

```
        while (temp) {
```

```
            if (temp->next != NULL && temp->next->val == temp->val) {
```

```
                while (temp->next != NULL && temp->next->val == temp->val)
```

```

temp = temp->next;

prev->next = temp->next;
}

else {
    prev = prev->next;

}

temp = temp->next;
}
return dummy->next;
}
};

```

The screenshot shows a C++ IDE interface with the following components:

- Top Bar:** Includes navigation icons, a 'Problem List' tab, and a 'Premium' badge.
- Left Sidebar:**
 - Description:** Shows 'Accepted 166 / 166 testcases passed' and 'Shantanu submitted at Feb 15, 2025 15:34'.
 - Runtime:** 0 ms | Beats 100.00%.
 - Memory:** 15.72 MB | Beats 41.50%.
 - Bar Chart:** A chart showing performance comparison across different test cases.
- Main Editor:** Displays the C++ code for reversing a linked list. The code uses a dummy node and iterates through the list, updating the next pointers.
- Bottom Right:** Shows the 'Test Result' section with 'Accepted' status and 'Runtime: 0 ms'.

Q7. [141. Linked List Cycle](#)

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode* fast = head;
        ListNode* slow = head;

        while (fast != nullptr && fast->next != nullptr) {
            fast = fast->next->next;
            slow = slow->next;

            if (fast == slow) {
                return true;
            }
        }

        return false;
    }
};
```

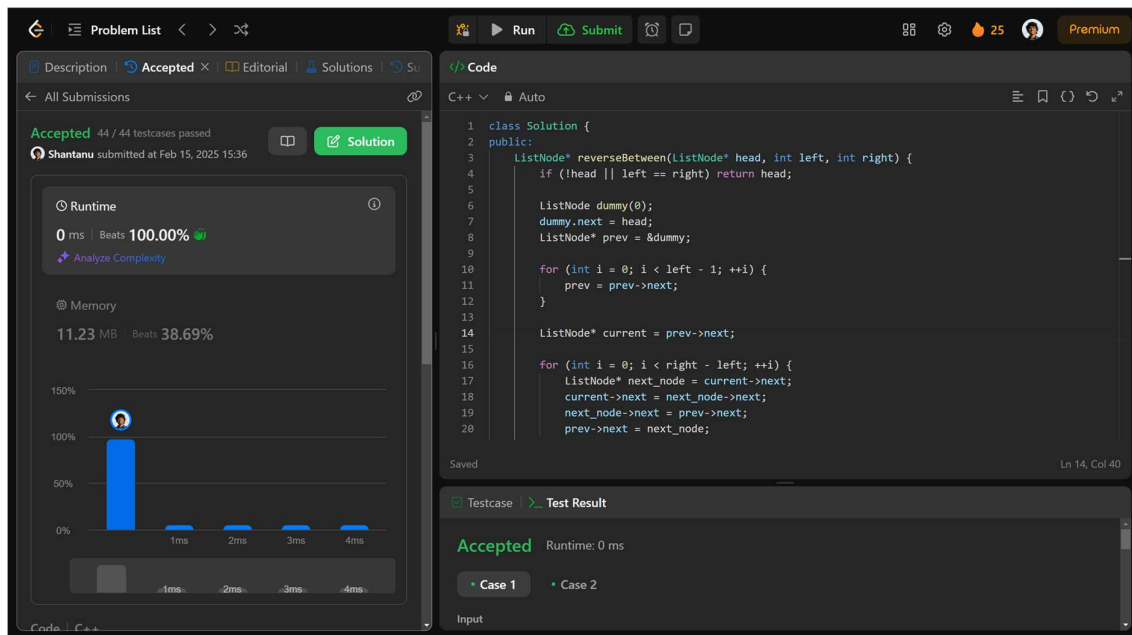
The screenshot displays a coding platform interface with the following components:

- Problem List:** Shows the problem name "141. Linked List Cycle" and its status as "Accepted".
- Submissions:** A list of submissions for this problem, with the top submission by "Shantanu" (submitted at Feb 15, 2025 15:34) marked as "Accepted".
- Runtime and Memory:** The runtime is 8 ms (Beats 81.20%) and the memory is 11.77 MB (Beats 80.07%).
- Code Editor:** Contains the C++ code for the solution, which implements the Floyd's Cycle-Finding algorithm (slow and fast pointers).
- Test Result:** Shows the result of the test cases, with "Accepted" and "Runtime: 0 ms".
- Case Selection:** A dropdown menu to select a specific test case (Case 1, Case 2, Case 3).

Q8. [92. Reverse Linked List II](#)

```
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (!head || left == right) return head;
        ListNode dummy(0);
        dummy.next = head;
        ListNode* prev = &dummy;
        for (int i = 0; i < left - 1; ++i) {
            prev = prev->next;
        }
        ListNode* current = prev->next;
        for (int i = 0; i < right - left; ++i) {
            ListNode* next_node = current->next;
            current->next = next_node->next;
            next_node->next = prev->next;
            prev->next = next_node;
        }

        return dummy.next;
    }
};
```



Q9. [61. Rotate List](#)

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if(head==nullptr || head->next==nullptr)
            return head;
        ListNode* fast=head;
        int cnt=0;
        while(fast != nullptr)
        {
            cnt++;
            fast=fast->next;
        }
        k= k%cnt;
        if(k==0)
            return head;
        fast=head;
        cnt=0;
        while(fast!=nullptr)
        {
            fast=fast->next;
            if(++cnt == k)
                break;
        }
        ListNode* slow=head;
        while(fast->next !=nullptr)
        {
            fast=fast->next;
            slow=slow->next;
        }
        fast->next=head;
        fast=slow->next;
        slow->next=nullptr;
        return fast;
    }
};
```

The screenshot displays a LeetCode submission for problem 148, "Sort List". The submission is marked as "Accepted" with 232/232 testcases passed. The runtime is 0 ms, which is faster than 100.00% of other submissions. The memory usage is 16.47 MB, which is better than 31.79% of other submissions. The code is written in C++ and implements a sorting algorithm for a linked list. The test result shows that the solution passed all test cases.

Q10. 148. Sort List

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        vector<int> arr;
        ListNode* temp = head;
        while(temp != nullptr){
            arr.push_back(temp->val);
            temp = temp->next;
        }
        sort(arr.begin(), arr.end());
        temp = head;
        for(int i = 0; temp != nullptr; i++){
            temp->val = arr[i];
            temp = temp->next;
        }
        return head;
    }
};
```

Problem List < >

Description | Accepted X | Editorial | Solutions | Su

< All Submissions

Accepted 30 / 30 testcases passed

Shantanu submitted at Feb 15, 2025 15:38

Runtime

13 ms | Beats 74.54%

Analyze Complexity

Memory

58.10 MB | Beats 75.12%

8% 6% 4% 2% 0%

22ms 43ms 65ms

27ms 54ms 81ms

Code

```

11 class Solution {
12 public:
13     ListNode* sortList(ListNode* head) {
14         vector<int> arr;
15         ListNode* temp = head;
16         while(temp != nullptr){
17             arr.push_back(temp->val);
18             temp = temp->next;
19         }
20         sort(arr.begin(), arr.end());
21         temp = head;
22         for(int i = 0; temp != nullptr; i++){
23             temp->val = arr[i];
24             temp = temp->next;
25         }
26         return head;
27     }
28 };
29

```

Saved Ln 29, Col 3

Testcase > Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

Q11. [142. Linked List Cycle II](#)

```

class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        ListNode *slow = head, *fast = head;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
            if (slow == fast) break;
        }
        if (!(fast && fast->next)) return NULL;
        while (head != slow) {
            head = head->next;
            slow = slow->next;
        }
        return head;
    }
};

```

Problem List

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 18 / 18 testcases passed

Shantanu submitted at Feb 15, 2025 15:38

Solution

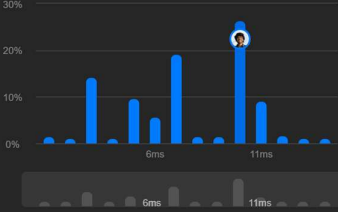
Runtime

10 ms | Beats 45.96%

Analyze Complexity

Memory

11.36 MB | Beats 53.95%



Code | C++

Run | Submit

Premium

Code

C++ | Auto

```
1 class Solution {
2 public:
3     ListNode *detectCycle(ListNode *head) {
4         ListNode *slow = head, *fast = head;
5         while (fast && fast->next) {
6             slow = slow->next;
7             fast = fast->next->next;
8             if (slow == fast) break;
9         }
10        if (!(fast && fast->next)) return NULL;
11        while (head != slow) {
12            head = head->next;
13            slow = slow->next;
14        }
15        return head;
16    }
17 };
```

Saved Ln 17, Col 3

Testcase | Test Result

Accepted Runtime: 2 ms

Case 1 Case 2 Case 3

Input