

Name : Ishita
UID : 22BCS14845
Subject : Advance Programming Lab

Date of Submission : 05/03/2025
Subject Code : 22CSP - 351
Submitted to : Er. Pratima Sonali

ASSIGNMENT - 2

Question 1

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

The screenshot shows a C++ IDE interface. On the left, the 'Submissions' tab is active, displaying a table with one submission record: 'Correct' status, 1 mark, C++ language, and 1112/1112 test cases passed. On the right, the code editor shows the implementation of the 'printList' function and the driver code in 'main'. The 'printList' function iterates through the linked list and prints the data of each node. The driver code reads the input, creates the linked list, and calls the 'printList' function.

```
32 // head pointer input could be NULL as well for empty list
33
34
35 class Solution {
36 public:
37     void printList(Node *head) {
38         Node*temp=head;
39         while(temp!=NULL){
40             cout<<temp->data<<" ";
41             temp=temp->next;
42         }
43     }
44 };
45
46 // Driver Code Starts.
47
48 int main() {
49     int t;
50     cin >> t;
51     cin.ignore(); // Ignore the newline character after t
52
53     while (t--) {
54         string input;
55         getline(cin, input); // Read the entire line for the array elements
56
57         stringstream ss(input);
58         Node *head = nullptr, *tail = nullptr;
59         int x;
60
```

Question 2

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (!head) {
            return nullptr;
        }

        ListNode* temp = head;
        ListNode* temp2 = head->next;
        int last = head->val;

        while (temp2 != nullptr) {
            if (temp2->val == last) {
                if (temp2->next == nullptr) {
                    temp->next = nullptr;
                    break;
                }
                temp2 = temp2->next; /
                temp->next = temp2;
            } else {
```

```

        temp = temp2;
        last = temp->val;
        temp2 = temp2->next;
    }
}
return head;
}
};

```

```

class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (!head) {
            return nullptr;
        }

        ListNode* temp = head;
        ListNode* temp2 = head->next;
        int last = head->val;

        while (temp2 != nullptr) {
            if (temp2->val == last) {
                if (temp2->next == nullptr) {
                    temp->next = nullptr;
                }
            } else {
                last = temp2->val;
                temp = temp2;
            }
            temp2 = temp2->next;
        }

        return head;
    }
};

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

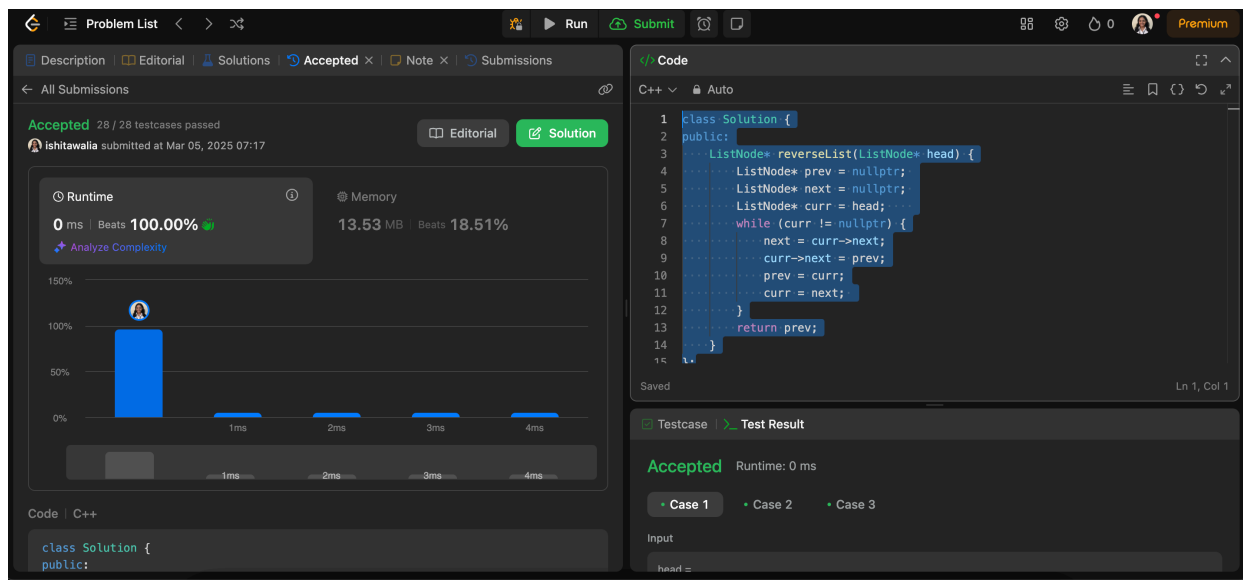
head =

Question 3

```

class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* next = nullptr;
        ListNode* curr = head;
        while (curr != nullptr) {
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
};

```

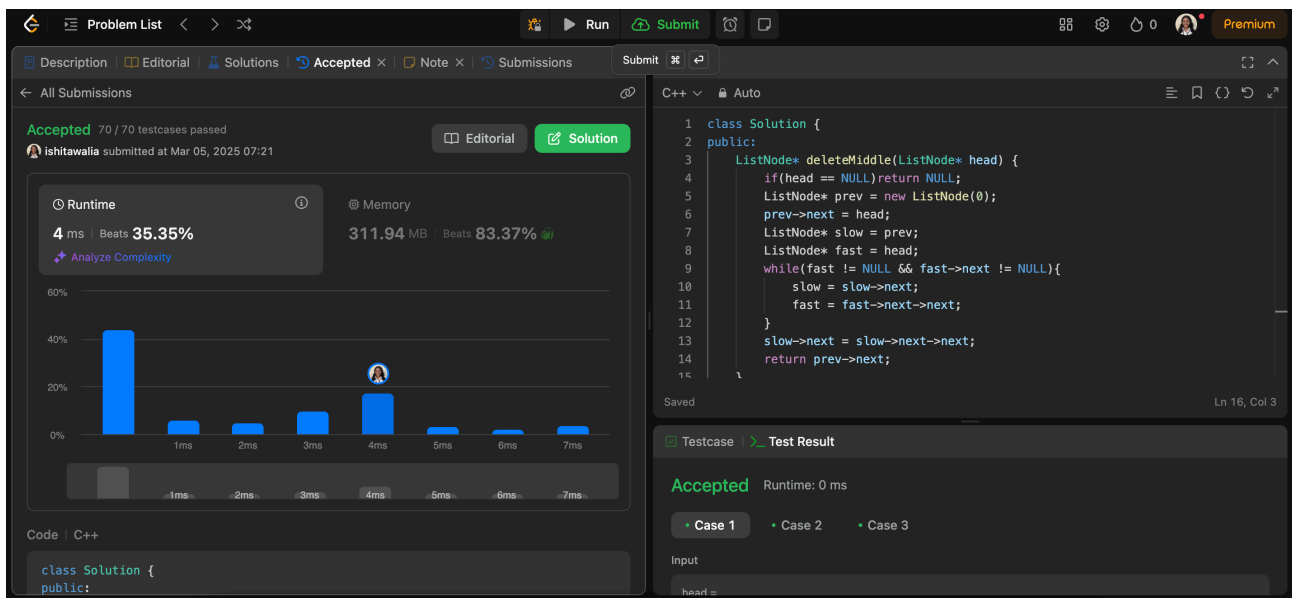


Question 4

```

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

```



Question 5

```

class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        if(l1 == NULL){

```

```

        return l2;
    }
    if(l2 == NULL){
        return l1;
    }
    if(l1 -> val <= l2 -> val){
        l1 -> next = mergeTwoLists(l1 -> next, l2);
        return l1;
    }
    else{
        l2 -> next = mergeTwoLists(l1, l2 -> next);
        return l2;
    }
}
};

```

The screenshot displays a C++ code submission on a platform. The top bar shows the problem list, run, submit, and other icons. The main area is divided into three sections: a description/submissions tab, a runtime/memory graph, and a code editor. The runtime graph shows a single bar at 0 ms. The code editor contains the following C++ code:

```

1 class Solution {
2 public:
3     ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
4         if(l1 == NULL){
5             return l2;
6         }
7         if(l2 == NULL){
8             return l1;
9         }
10        if(l1 -> val <= l2 -> val){
11            l1 -> next = mergeTwoLists(l1 -> next, l2);
12            return l1;
13        }
14        else{
15            l2 -> next = mergeTwoLists(l1, l2 -> next);
16        }
17    }
18 }

```

The bottom right section shows the test results, indicating that the solution is 'Accepted' with a runtime of 0 ms. The input field shows 'list1 = '.

Question 6

```

class Solution {
public:
    bool 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;
    }
};

```

Problem List < > > Run Submit

Description Editorial Solutions Accepted x Note x Submissions

All Submissions

Accepted 29 / 29 testcases passed
ishitawalia submitted at Mar 05, 2025 07:25

Editorial Solution

Runtime 12 ms | Beats 40.49%
Memory 11.95 MB | Beats 24.25%

Analyze Complexity

40%
20%
0%
3ms 5ms 7ms 9ms 11ms 13ms 15ms 17ms 19ms

Code | C++

```
class Solution {
public:

```

Code

```
2 public:
3     bool hasCycle(ListNode* head) {
4         ListNode* slow = head;
5         ListNode* fast = head;
6         while (fast != NULL && fast->next != NULL) {
7             slow = slow->next;
8             fast = fast->next->next;
9             if (slow == fast) {
10                 return true;
11             }
12         }
13         return false;
14     }
15 };
```

Saved Ln 15, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =

Question 7

```
class Solution {
public:

```

```
    ListNode* rotateRight(ListNode* head, int k) {
        // base condition
        if(head==NULL || head->next==NULL || k==0) return head;

```

```
        ListNode* curr=head;
        int count=1;
        while(curr->next!=NULL){
            curr=curr->next;
            count++;
        }
        curr->next=head;
        k=count-(k%count);
        while(k-->0){
            curr=curr->next;
        }
        head=curr->next;
        curr->next=NULL; // curr points to tail node sorta

```

```
        return head;

```

```
    }
};

```

Problem List < > > Run Submit

Description Editorial Solutions Accepted x Note x Submissions

All Submissions

Accepted 232 / 232 testcases passed
ishitawalia submitted at Mar 05, 2025 07:27

Editorial Solution

Runtime 0 ms | Beats 100.00%
Memory 16.50 MB | Beats 4.05%

Analyze Complexity

100%
50%
0%
1ms 2ms 3ms 4ms

Code | C++

```
class Solution {
public:

```

Code

```
11         count++;
12     }
13     curr->next=head;
14     k=count-(k%count);
15     while(k-->0){
16         curr=curr->next;
17     }
18     head=curr->next;
19     curr->next=NULL; // curr points to tail node sorta
20
21     return head;
22 }
23
24 };
```

Saved Ln 24, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

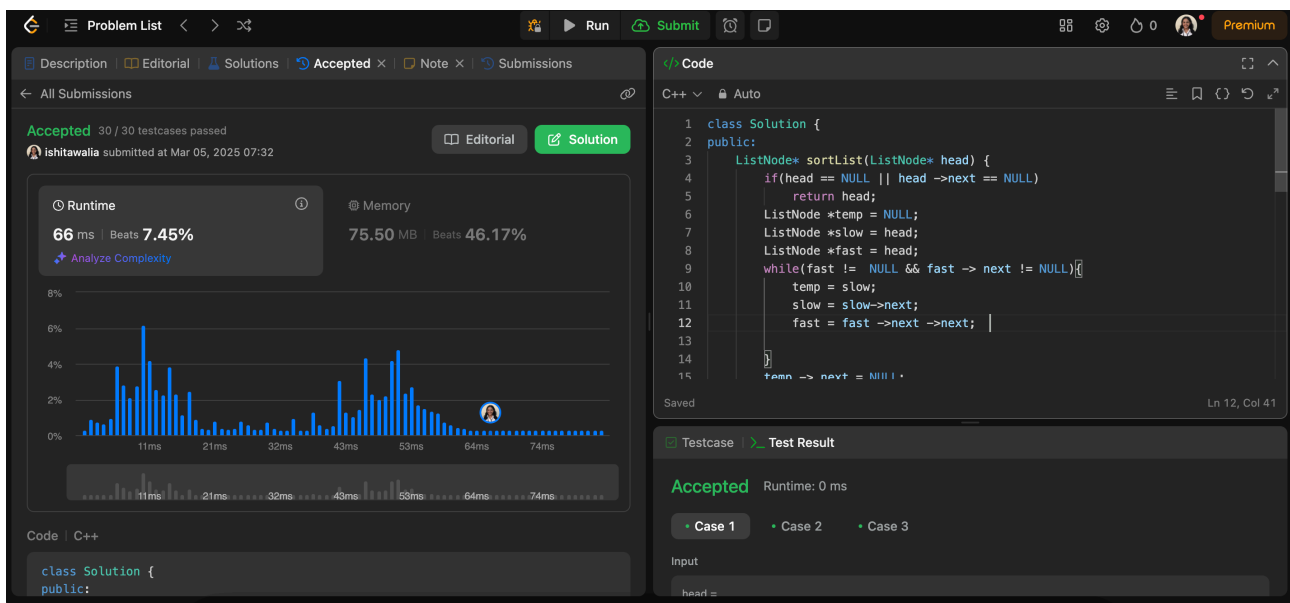
Case 1 Case 2

Input

head =

Question 8

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if(head == NULL || head ->next == NULL)
            return head;
        ListNode *temp = NULL;
        ListNode *slow = head;
        ListNode *fast = head;
        while(fast != NULL && fast -> next != NULL){
            temp = slow;
            slow = slow->next;
            fast = fast ->next ->next;
        }
        temp -> next = NULL;
        ListNode* l1 = sortList(head);
        ListNode* l2 = sortList(slow);
        return mergelist(l1, l2);
    }
    ListNode* mergelist(ListNode *l1, ListNode *l2){
        ListNode *ptr = new ListNode(0);
        ListNode *curr = ptr;
        while(l1 != NULL && l2 != NULL){
            if(l1->val <= l2->val){
                curr -> next = l1;
                l1 = l1 -> next;
            }
            else{
                curr -> next = l2;
                l2 = l2 -> next;
            }
        }
        curr = curr ->next;
    }
    if(l1 != NULL){
        curr -> next = l1;
        l1 = l1->next;
    }
    if(l2 != NULL){
        curr -> next = l2;
        l2 = l2 ->next;
    }
    return ptr->next;
}
};
```



Question 9

```
#include <vector>
```

```
using namespace std;
```

```
class Solution {
```

```
public:
```

```
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
```

```
        if (!l1) return l2;
```

```
        if (!l2) return l1;
```

```
        if (l1->val < l2->val) {
```

```
            l1->next = mergeTwoLists(l1->next, l2);
```

```
            return l1;
```

```
        } else {
```

```
            l2->next = mergeTwoLists(l1, l2->next);
```

```
            return l2;
```

```
        }
```

```
    }
```

```
    ListNode* mergeKLists(vector<ListNode*>& lists) {
```

```
        if (lists.empty()) return nullptr;
```

```
        return divideAndConquer(lists, 0, lists.size() - 1);
```

```
    }
```

```
    ListNode* divideAndConquer(vector<ListNode*>& lists, int left, int right) {
```

```
        if (left == right) return lists[left];
```

```
        int mid = left + (right - left) / 2;
```

```
        ListNode* l1 = divideAndConquer(lists, left, mid);
```

```
        ListNode* l2 = divideAndConquer(lists, mid + 1, right);
```

```
        return mergeTwoLists(l1, l2);
```

```
    }
```

```
};
```

Problem List

Accepted

All Submissions

Accepted 134 / 134 testcases passed
ishitawalia submitted at Mar 05, 2025 07:33

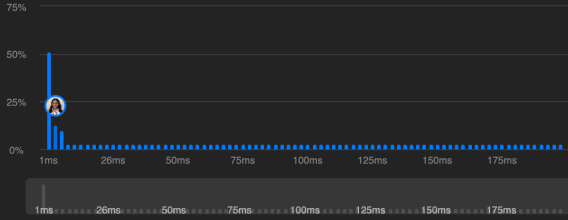
Runtime

4 ms | Beats 48.89%

Analyze Complexity

Memory

18.69 MB | Beats 40.18%



Code | C++

#include <vector>
using namespace std;

Code

```
18     ListNode* mergeKLists(vector<ListNode*>& lists) {  
19         if (lists.empty()) return nullptr;  
20         return divideAndConquer(lists, 0, lists.size() - 1);  
21     }  
22  
23     ListNode* divideAndConquer(vector<ListNode*>& lists, int left, int right) {  
24         if (left == right) return lists[left];  
25  
26         int mid = left + (right - left) / 2;  
27         ListNode* l1 = divideAndConquer(lists, left, mid);  
28         ListNode* l2 = divideAndConquer(lists, mid + 1, right);  
29         return mergeTwoLists(l1, l2);  
30     }  
31 };
```

Saved

Ln 31, Col 3

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

lists =