Assignment 3

Advanced Programming Lab – II

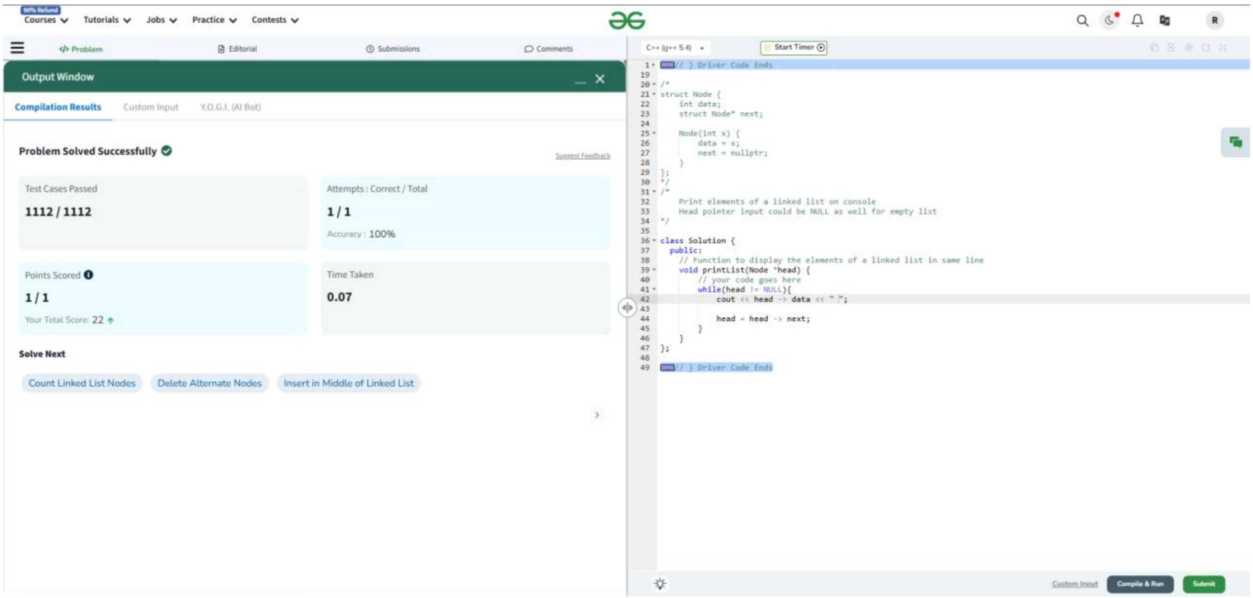
Submitted By – Ratnesh Tiwari

UID – 22BCS17201

Section – 22BCS_IOT – 609

Submitted To – Pratima Sonali Horo (E18304)

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



Problem Solved Successfully

Test Cases Passed
1112 / 1112

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

Points Scored
1 / 1
Your Total Score: 22

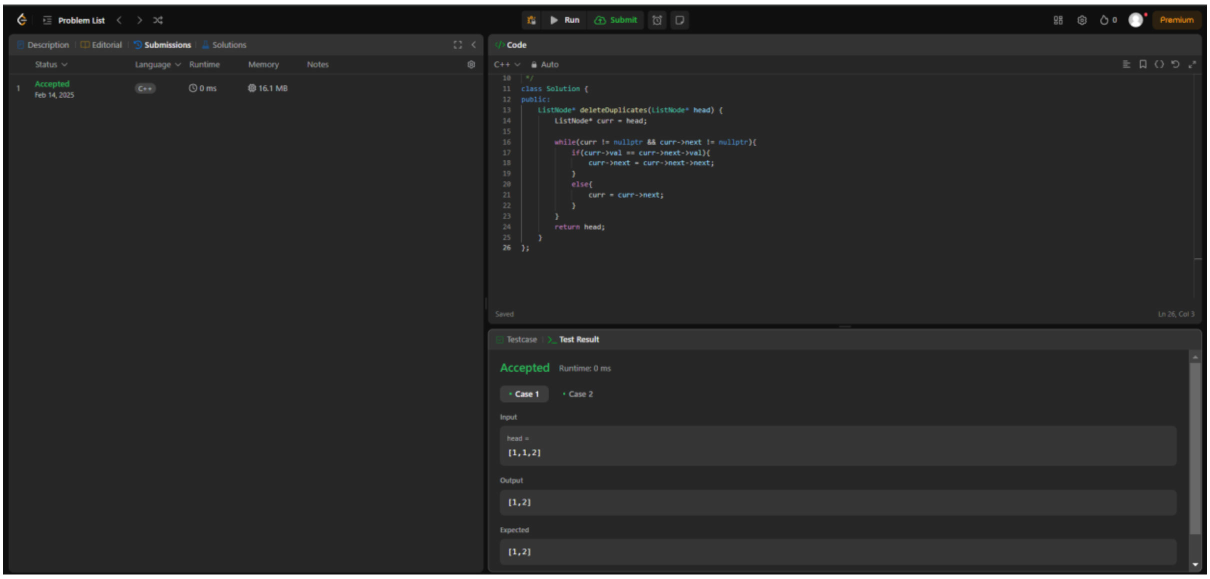
Time Taken
0.07

Solve Next

Count Linked List Nodes
Delete Alternate Nodes
Insert in Middle of Linked List

```
// } Driver Code Ends

/*
struct Node {
    int data;
    struct Node* next;

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

class Solution {
  public:
    // Function to display the elements of a linked list in same line
    void printList(Node *head) {
        // your code goes here
        while(head != NULL){
            cout << head -> data << " ";

            head = head -> next;
        }
    }
};

// } Driver Code Ends
```

2. Remove Duplicates from a Sorted List: https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/



	Status	Language	Runtime	Memory	Notes
1	Accepted Feb 14, 2025	C++	0 ms	16.1 MB	

```cpp
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* curr = head;

        while(curr != nullptr && curr->next != nullptr){
            if(curr->val == curr->next->val){
                curr->next = curr->next->next;
            }
            else{
                curr = curr->next;
            }
        }
        return head;
    }
};
```

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

```
head =
[1,1,2]
```

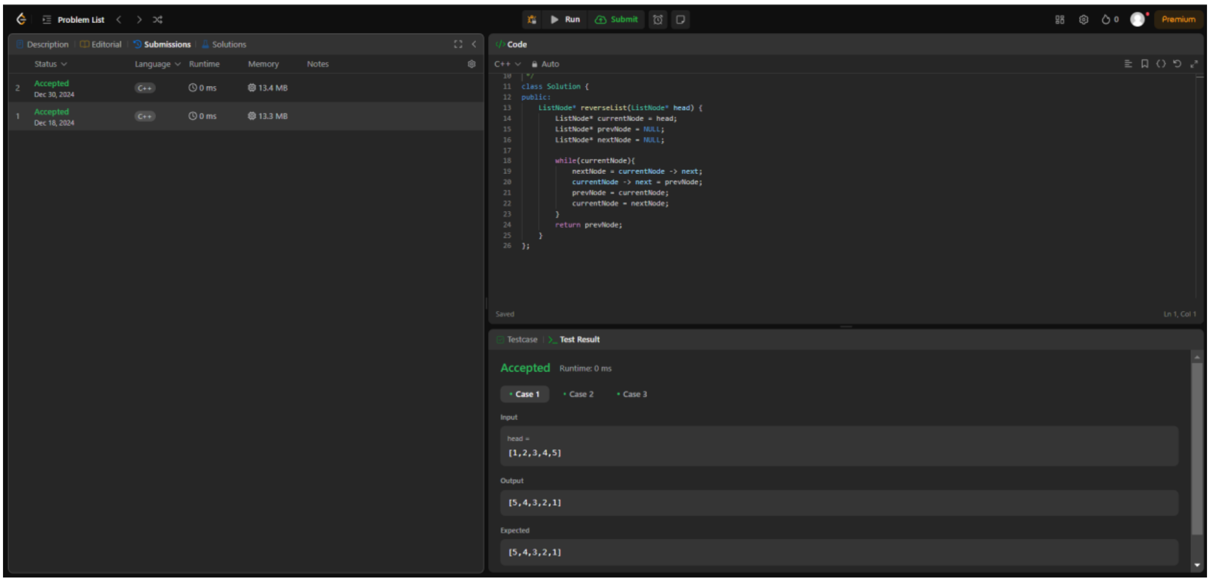
Output

```
[1,2]
```

Expected

```
[1,2]
```

3. Reverse a Linked List: https://leetcode.com/problems/reverse-linked-list/description/



	Status	Language	Runtime	Memory	Notes
2	Accepted Dec 30, 2024	C++	0 ms	13.4 MB	
1	Accepted Dec 18, 2024	C++	0 ms	13.3 MB	

```cpp
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* currentNode = head;
        ListNode* prevNode = NULL;
        ListNode* nextNode = NULL;

        while(currentNode){
            nextNode = currentNode -> next;
            currentNode -> next = prevNode;
            prevNode = currentNode;
            currentNode = nextNode;
        }
        return prevNode;
    }
};
```

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

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

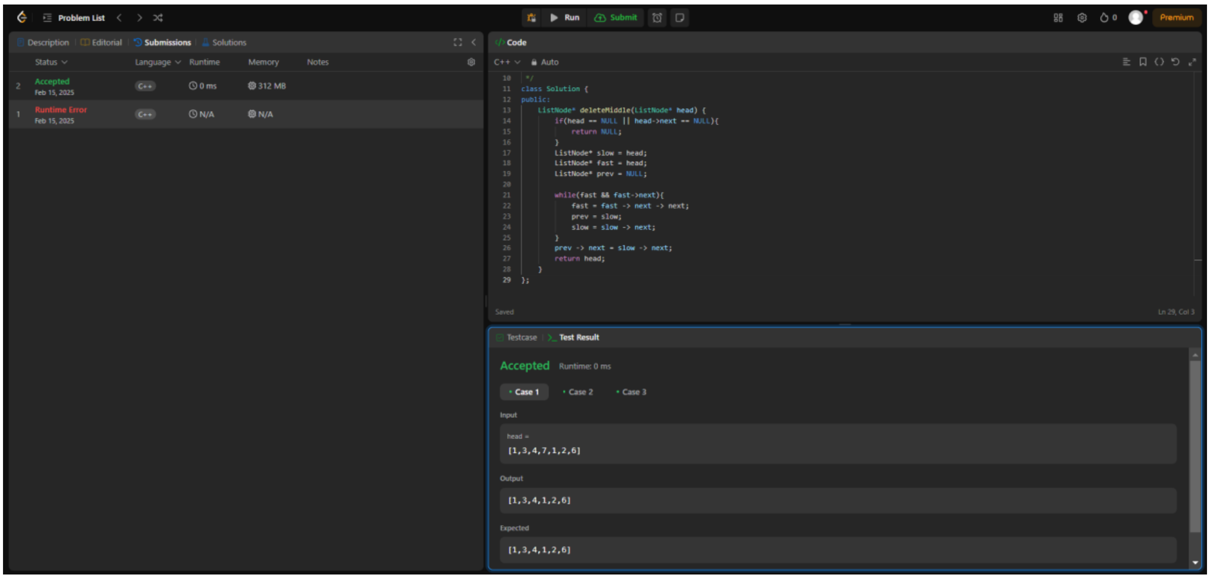
Output

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

Expected

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

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



	Status	Language	Runtime	Memory	Notes
2	Accepted Feb 15, 2025	C++	0 ms	312 MB	
1	Runtime Error Feb 15, 2025	C++	N/A	N/A	

```cpp
 */
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 && fast->next){
            fast = fast -> next -> next;
            prev = slow;
            slow = slow -> next;
        }
        prev -> next = slow -> next;
        return head;
    }
};
```

Accepted Runtime: 0 ms

Input

head =

```
[1,3,4,7,1,2,6]
```

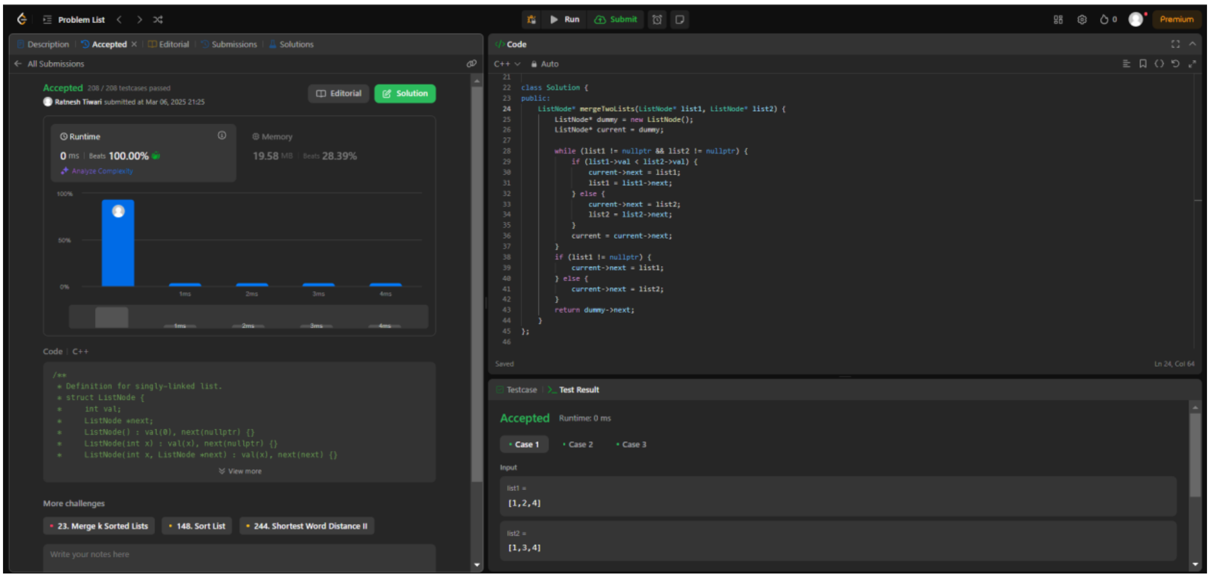
Output

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

Expected

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

5. Merge two sorted linked lists: https://leetcode.com/problems/merge-two-sorted-lists/description/



Accepted 208 / 208 testcases passed

Ratnesh Tiwari submitted at Mar 06, 2025 21:25

Runtime: 0 ms | Beats 100.00%

Memory: 19.58 MB | Beats 28.39%

Code | C++

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
```

```cpp

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

        while (list1 != nullptr && list2 != nullptr) {
            if (list1->val < list2->val) {
                current->next = list1;
                list1 = list1->next;
            } else {
                current->next = list2;
                list2 = list2->next;
            }
            current = current->next;
        }
        if (list1 != nullptr) {
            current->next = list1;
        } else {
            current->next = list2;
        }
        return dummy->next;
    }
};

```

Accepted Runtime: 0 ms

Input

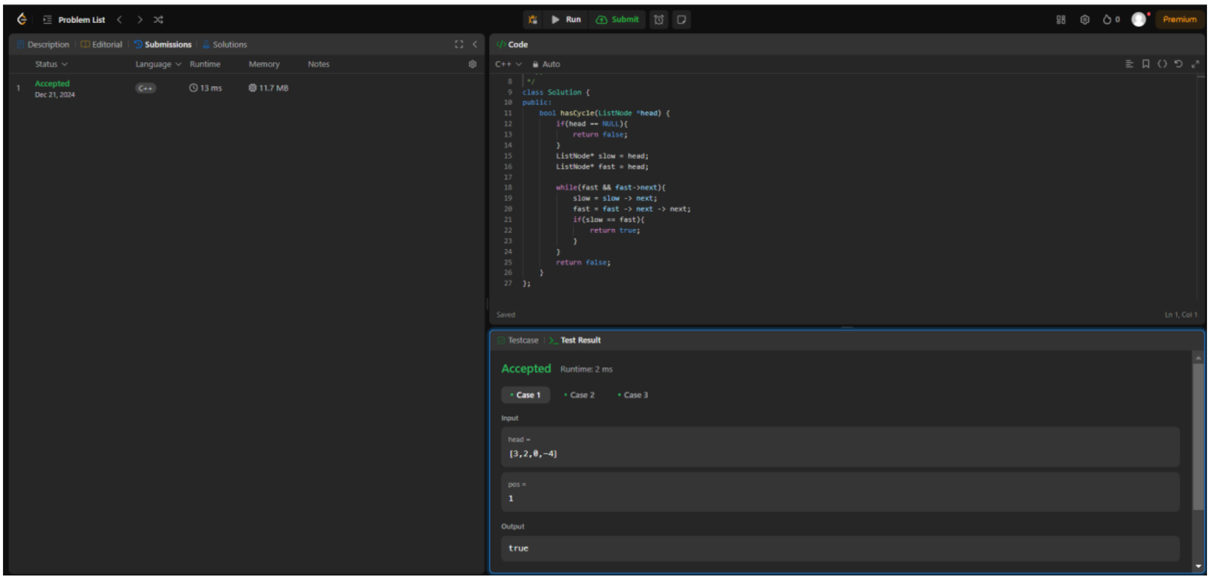
list1 =

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

list2 =

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

6. Detect a cycle in a linked list: https://leetcode.com/problems/linked-list-cycle/description/



	Status	Language	Runtime	Memory	Notes
1	Accepted Dec 21, 2024	C++	13 ms	11.7 MB	

Code
C++ Auto

```
*/
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if(head == NULL){
            return false;
        }
        ListNode* slow = head;
        ListNode* fast = head;

        while(fast && fast->next){
            slow = slow -> next;
            fast = fast -> next -> next;
            if(slow == fast){
                return true;
            }
        }
        return false;
    }
};
```

Saved Ln 1, Col 1

Testcase | Test Result

Accepted Runtime: 2 ms

Case 1 Case 2 Case 3

Input

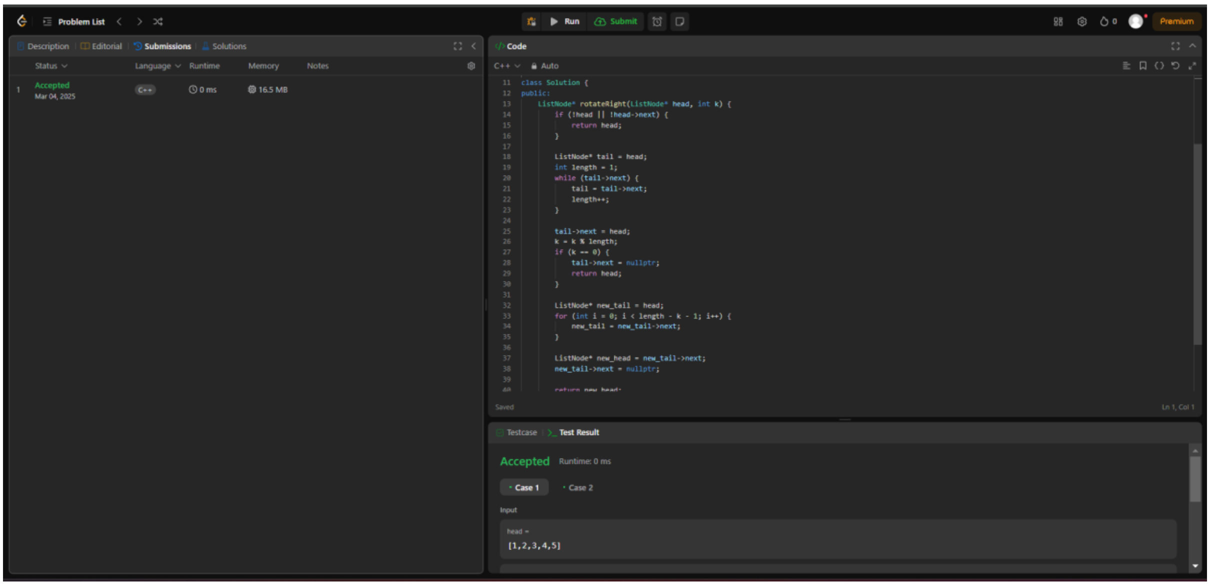
head =
[3,2,0,-4]

pos =
1

Output

true

7. Rotate a list: https://leetcode.com/problems/rotate-list/description/



	Status	Language	Runtime	Memory	Notes
1	Accepted Mar 04, 2025	C++	0 ms	16.5 MB	

Code
C++ Auto

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next) {
            return head;
        }

        ListNode* tail = head;
        int length = 1;
        while (tail->next) {
            tail = tail->next;
            length++;
        }

        tail->next = head;
        k = k % length;
        if (k == 0) {
            tail->next = nullptr;
            return head;
        }

        ListNode* new_tail = head;
        for (int i = 0; i < length - k - 1; i++) {
            new_tail = new_tail->next;
        }

        ListNode* new_head = new_tail->next;
        new_tail->next = nullptr;

        return new_head;
```

Saved Ln 1, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

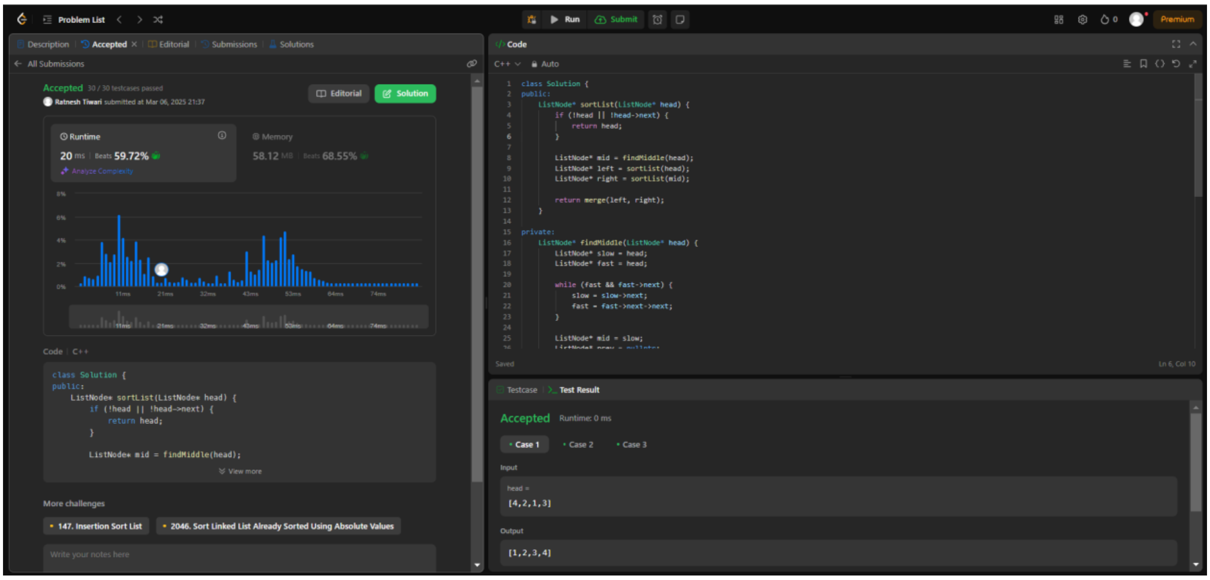
Case 1 Case 2

Input

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

8. Sort List: https://leetcode.com/problems/sort-list/description/

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) {
            return head;
        }

        ListNode* mid = findMiddle(head);
        ListNode* left = sortList(head);
        ListNode* right = sortList(mid);

        return merge(left, right);
    }

private:
    ListNode* findMiddle(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

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

        ListNode* mid = slow;
        ListNode* prev = nullptr;

        while (head != mid) {
            prev = head;
            head = head->next;
        }

        if (prev) {
            prev->next = nullptr;
        }

        return mid;
    }

    ListNode* merge(ListNode* left, ListNode* right) {
        if (!left) return right;
        if (!right) return left;
        if (left->val < right->val) {
            left->next = merge(left->next, right);
            return left;
        } else {
            right->next = merge(left, right->next);
            return right;
        }
    }
};
```



Accepted 30 / 30 testcases passed

Ratnesh Tiwari submitted at Mar 06, 2025 21:37

Runtime 20 ms | Beats 59.72%

Memory 58.12 MB | Beats 68.55%

Code | C++

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) {
            return head;
        }

        ListNode* mid = findMiddle(head);
```

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) {
            return head;
        }

        ListNode* mid = findMiddle(head);
        ListNode* left = sortList(head);
        ListNode* right = sortList(mid);

        return merge(left, right);
    }

private:
    ListNode* findMiddle(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

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

        ListNode* mid = slow;
```

Accepted Runtime: 0 ms

Input

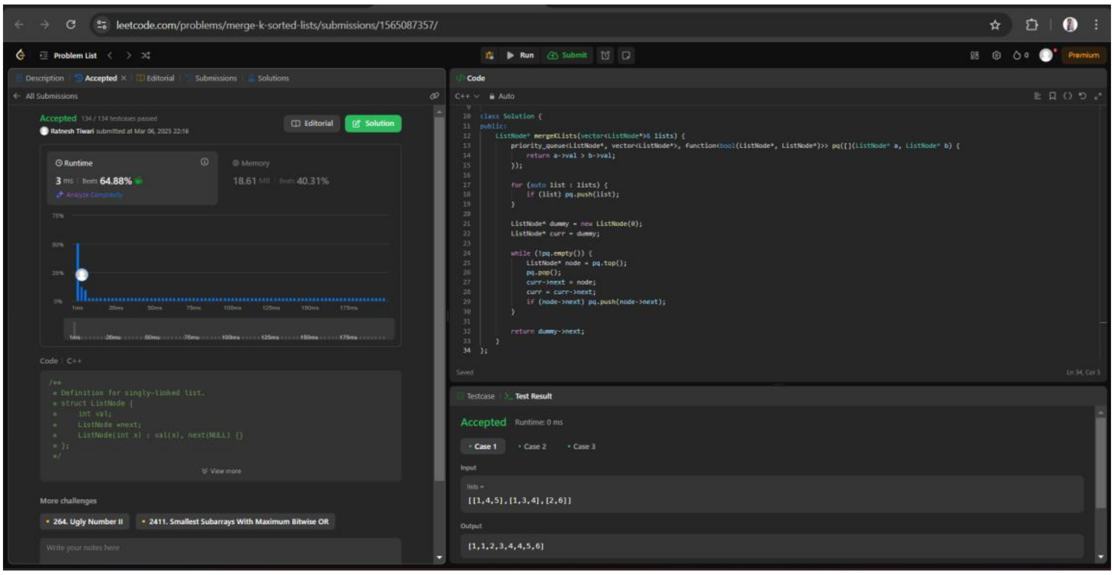
head =

[4,2,1,3]

Output

[1,2,3,4]

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



leetcode.com/problems/merge-k-sorted-lists/submissions/1565087357/

Accepted 134 / 134 testcases passed

Ratnesh Tiwari submitted at Mar 06, 2025 22:16

Runtime 3 ms | Beats 64.88%

Memory 18.61 MB | Beats 40.31%

Code | C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
```

```
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        priority_queue<ListNode*, vector<ListNode*>, function<bool(ListNode*, ListNode*)>> pq([](ListNode* a, ListNode* b) {
            return a->val > b->val;
        });

        for (auto list : lists) {
            if (list) pq.push(list);
        }

        ListNode* dummy = new ListNode(0);
        ListNode* curr = dummy;

        while (!pq.empty()) {
            ListNode* node = pq.top();
            pq.pop();
            curr->next = node;
            curr = curr->next;
            if (node->next) pq.push(node->next);
        }

        return dummy->next;
    }
};
```

Accepted Runtime: 0 ms

Input

lists =

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

Output

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