Assignment-3

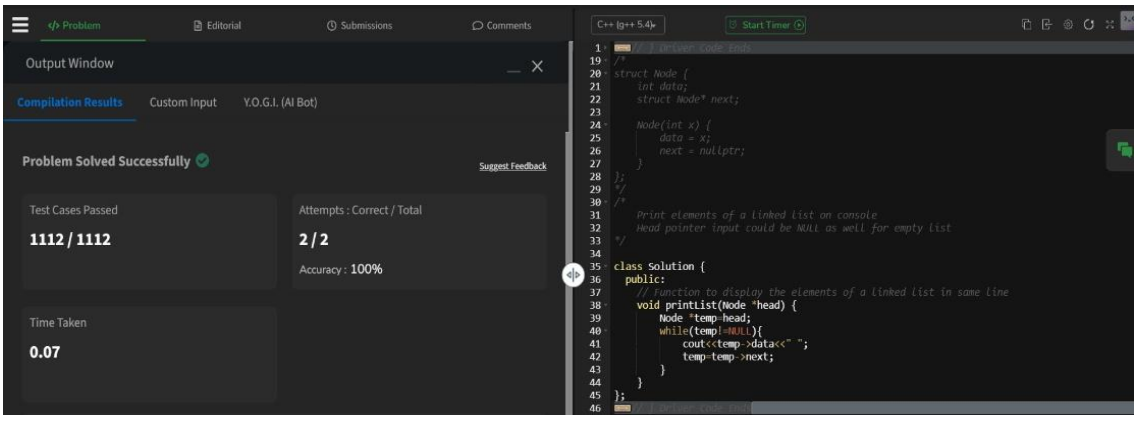
Student Name: Deepa Kumari **UID: 22BCS10272**
Branch: CSE **Section/Group: IOT-609-B**
Semester: 6th **Date: 05-03-2025**
Subject Name: AP Lab **Subject Code: 22CSP-351**

Ques 1: Print Linked List

Code: -

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

Submission: -

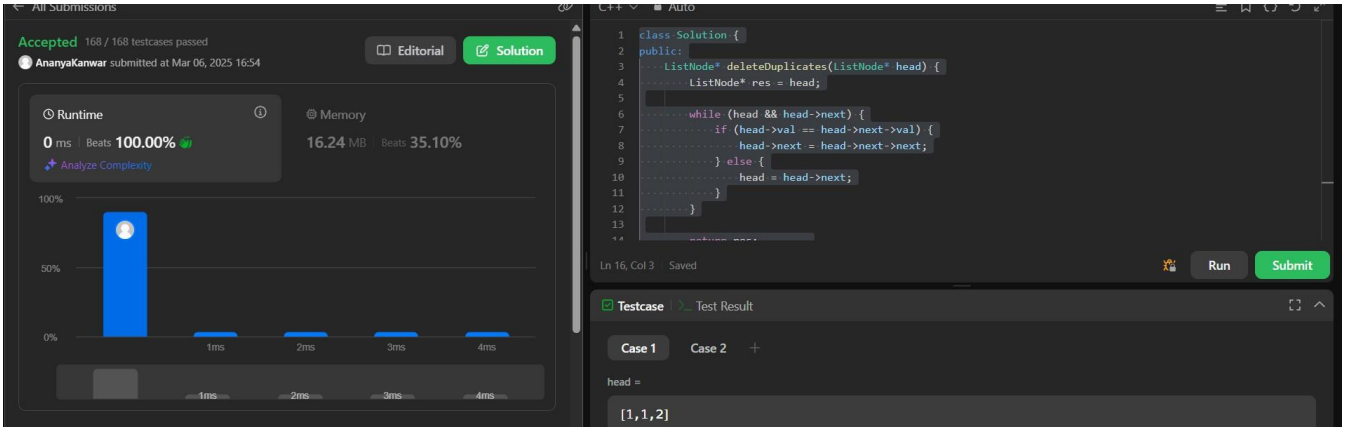


Ques 2: Remove Duplicates from Sorted List

Code: -

```
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: -

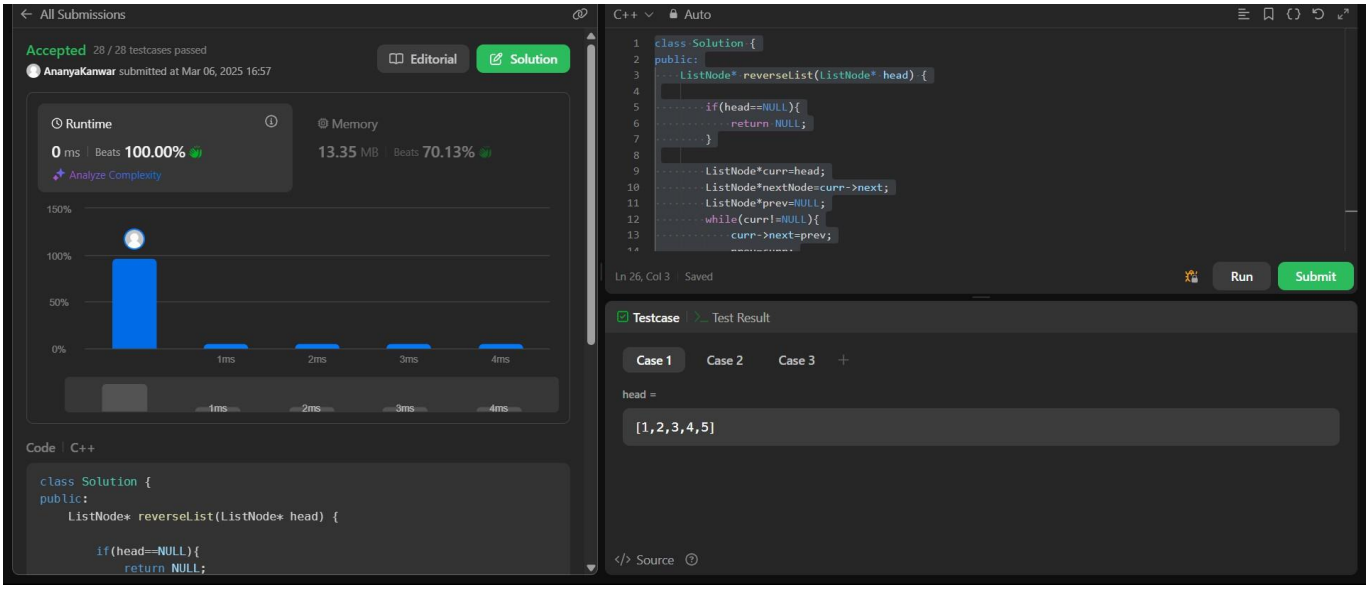


Ques 3: Reverse Linked List

Code: -

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

Submission: -

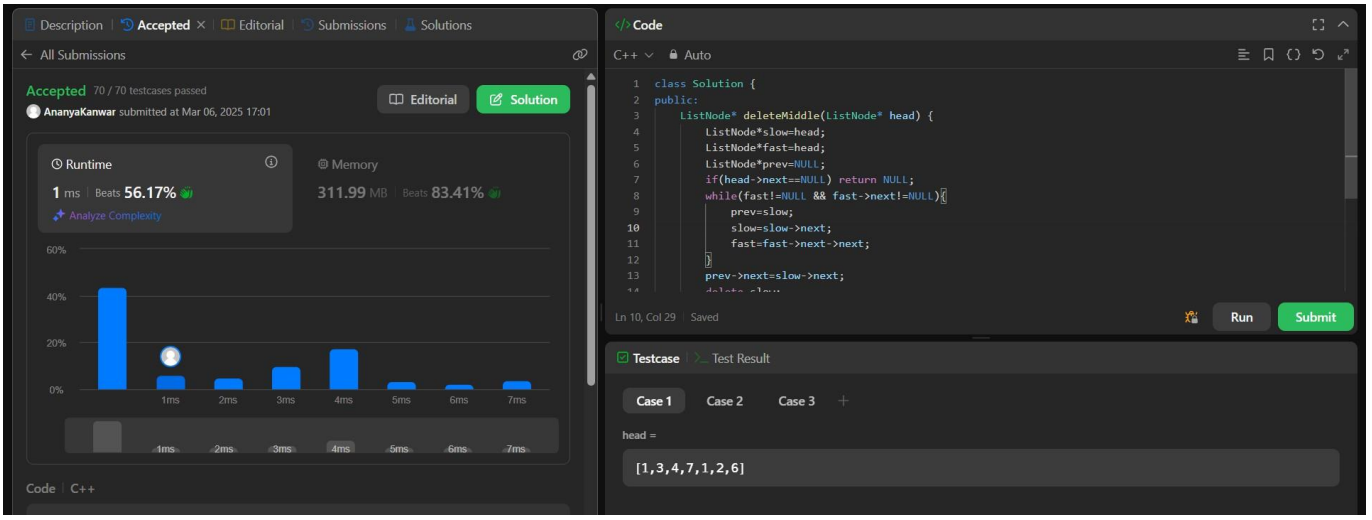


Ques 4: Delete the Middle Node of a Linked List

Code: -

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

Submission: -

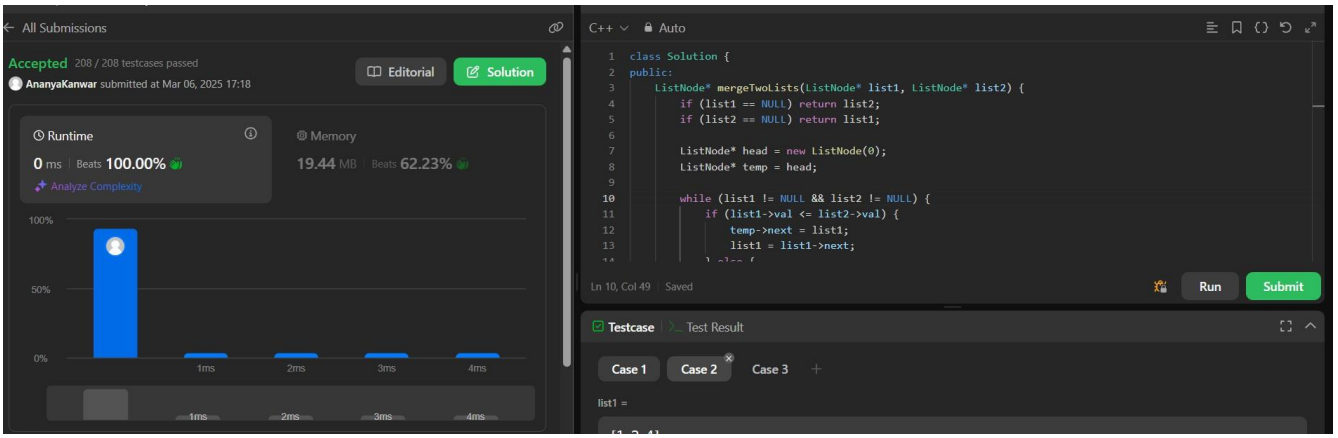


Ques 5: Merge Two Sorted Lists

Code: -

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if (list1 == NULL) return list2;
        if (list2 == NULL) return list1;
        ListNode* head = new ListNode(0);
        ListNode* temp = head;
        while (list1 != NULL && list2 != NULL) {
            if (list1->val <= list2->val) {
                temp->next = list1;
                list1 = list1->next;
            } else {
                temp->next = list2;
                list2 = list2->next;
            }
            temp = temp->next;
        }
        if (list1 != NULL) temp->next = list1;
        if (list2 != NULL) temp->next = list2;
        return head->next;
    }
};
```

Submission: -

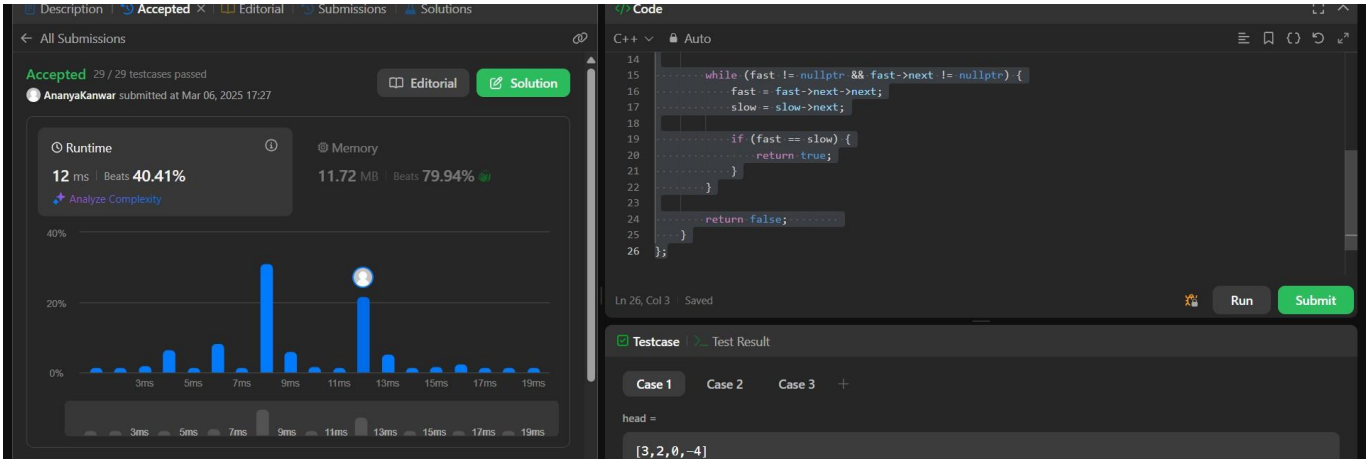


Ques 6: Linked List Cycle

Code: -

```
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;
    }
};
```

Submission: -

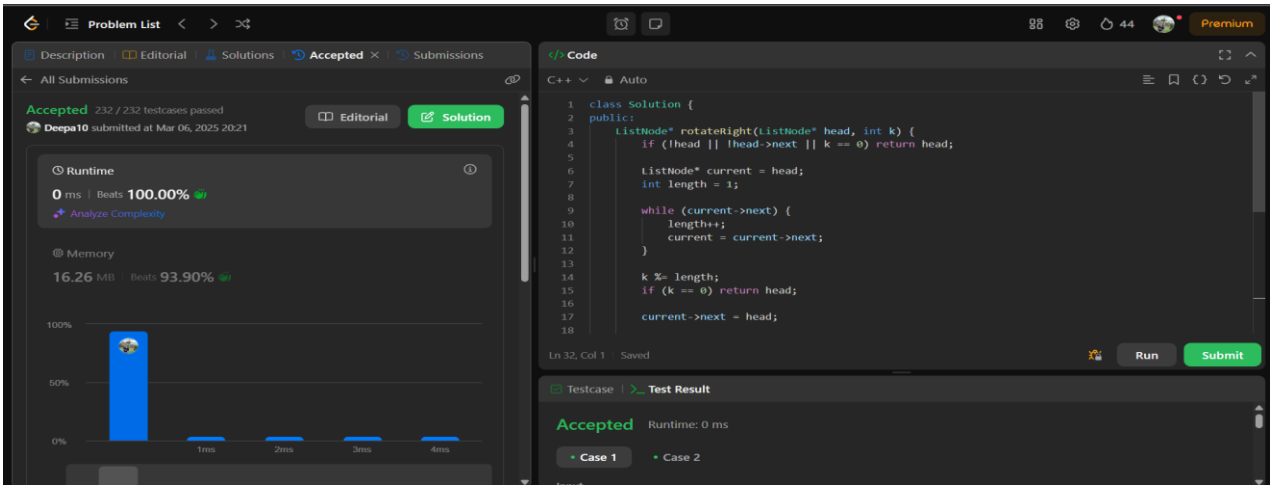


Ques 7: Rotate Lists

Code: -

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

Submission: -



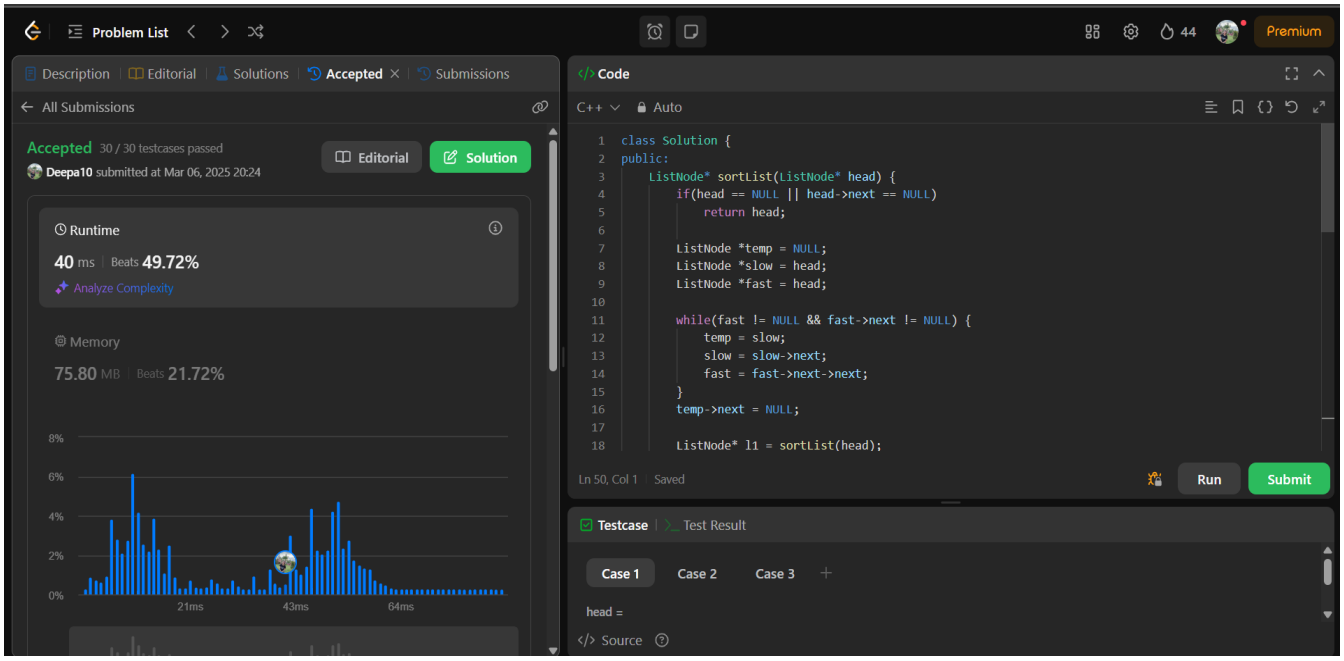
Ques 8: Sort Lists

Code: -

```
class Solution {
public:
    ListNode* merge(ListNode* l1, ListNode* l2) {
        if (!l1) return l2;
        if (!l2) return l1;

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

    ListNode* findMid(ListNode* head) {
        ListNode* slow = head, *fast = head->next;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
        }
        return slow;
    }

    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) return head;

        ListNode* mid = findMid(head);
        ListNode* right = mid->next;
        mid->next = nullptr;

        ListNode* leftSorted = sortList(head);
        ListNode* rightSorted = sortList(right);
```

```
        return merge(leftSorted, rightSorted);
    }
};
```

Submission: -

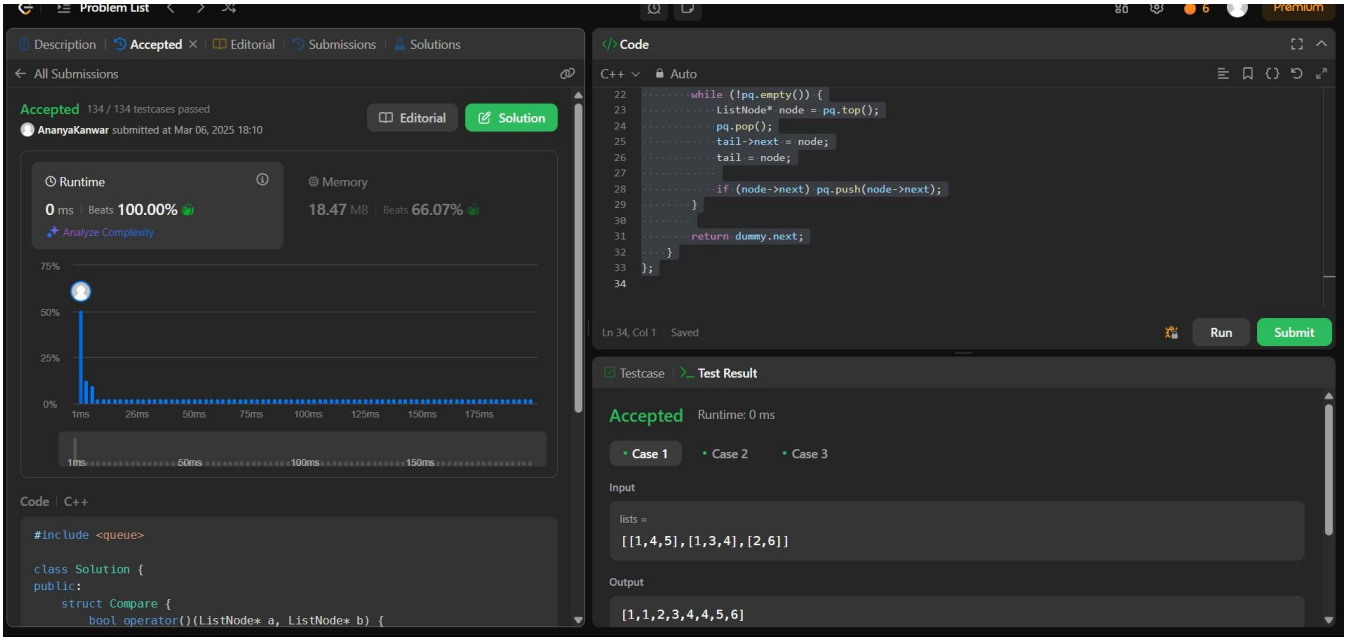


Ques 9: Merge k sorted lists

Code: -

```
#include <queue>

class Solution {
public:
    struct Compare {
        bool operator()(ListNode* a, ListNode* b) {
            return a->val > b->val;  // Min-heap based on node values
        }
    };

    ListNode* mergeKLists(vector<ListNode*>& lists) {
        priority_queue<ListNode*, vector<ListNode*>, Compare> pq;

        // Push all non-null list heads into the priority queue
        for (auto list : lists) {
            if (list) pq.push(list);
        }

        ListNode dummy(0);
        ListNode* tail = &dummy;

        while (!pq.empty()) {
            ListNode* node = pq.top();
            pq.pop();
            tail->next = node;
            tail = node;

            if (node->next) pq.push(node->next);
        }

        return dummy.next;
    }
};
```

Submission: -


Problem List
Description
Accepted
Editorial
Submissions
Solutions
All Submissions
Accepted 134 / 134 testcases passed
AnanyaKanwar submitted at Mar 06, 2025 18:10
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
18.47 MB | Beats 66.07%
Code | C++
#include <queue>
class Solution {
public:
struct Compare {
bool operator()(ListNode* a, ListNode* b) {
Code
C++
Auto
while (!pq.empty()) {
ListNode* node = pq.top();
pq.pop();
tail->next = node;
tail = node;
if (node->next) pq.push(node->next);
}
return dummy.next;
}
};
Ln 34, Col 1 | Saved
Run
Submit
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
lists =
[[1,4,5],[1,3,4],[2,6]]
Output
[1,1,2,3,4,4,5,6]