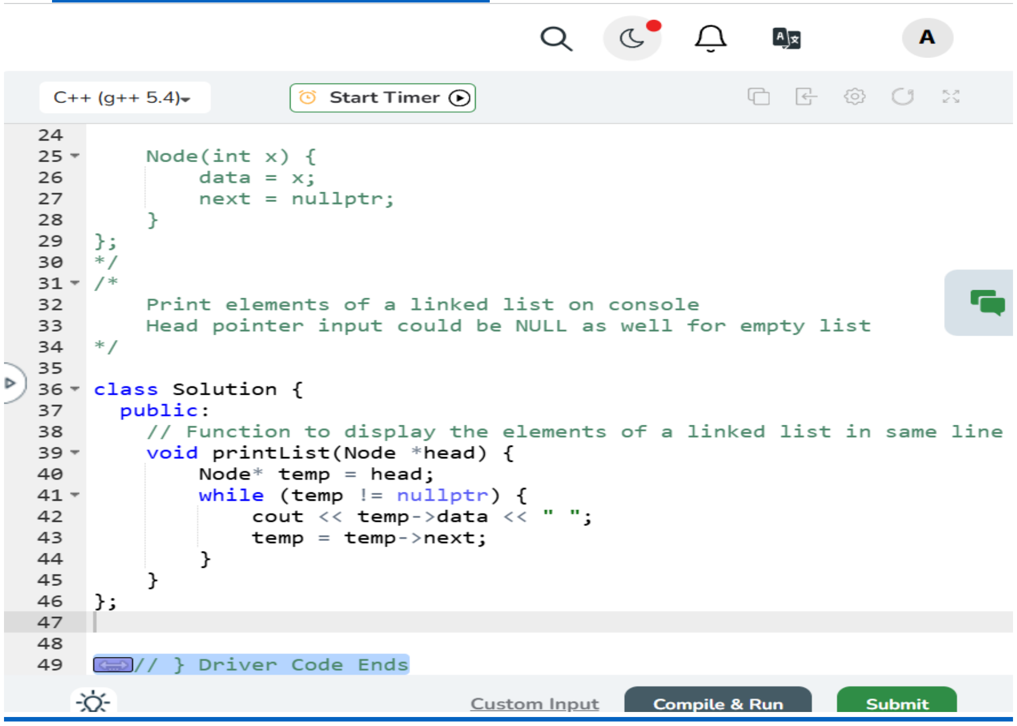
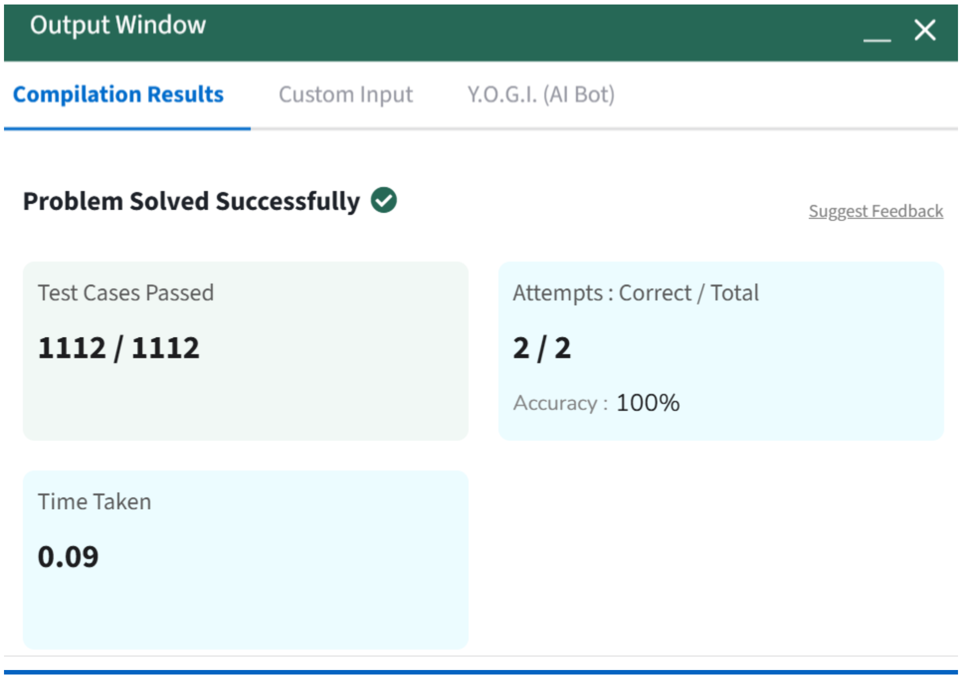
ASSIGNMENT-3

NAME: Aryadeep
UID: 22BCS10915
SUBJECT: AP-2
CLASS- IOT-610/B

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



```
    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) {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next;
        }
    }
};


// } Driver Code Ends
```


Output Window

Compilation Results   Custom Input   Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓   Suggest Feedback

Test Cases Passed

1112 / 1112

Attempts : Correct / Total

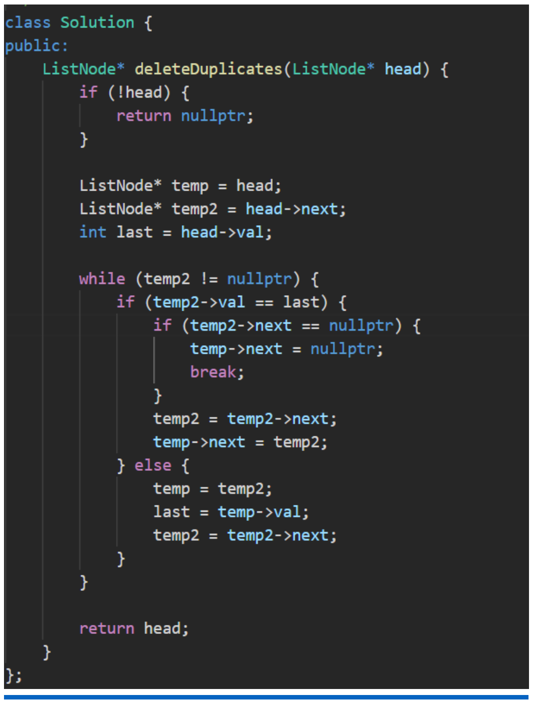
2 / 2

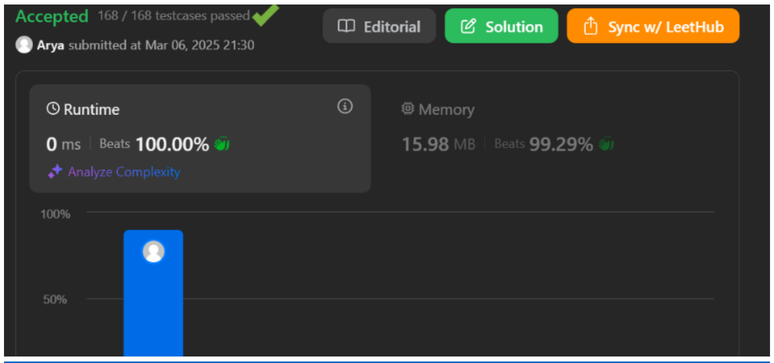
Accuracy : 100%

Time Taken

0.09

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


```
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;
    }
};
```



Accepted 168 / 168 testcases passed

Arya submitted at Mar 06, 2025 21:30

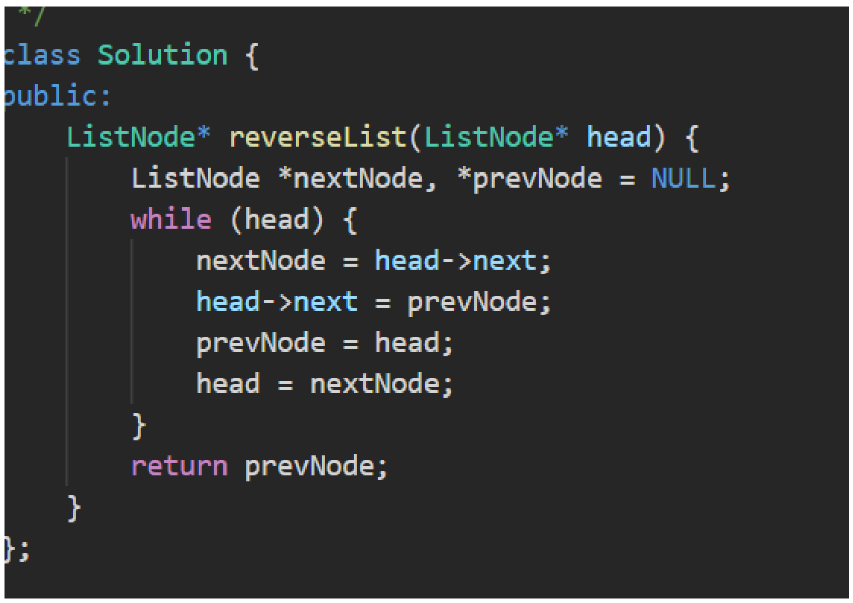
Runtime

0 ms | Beats 100.00%

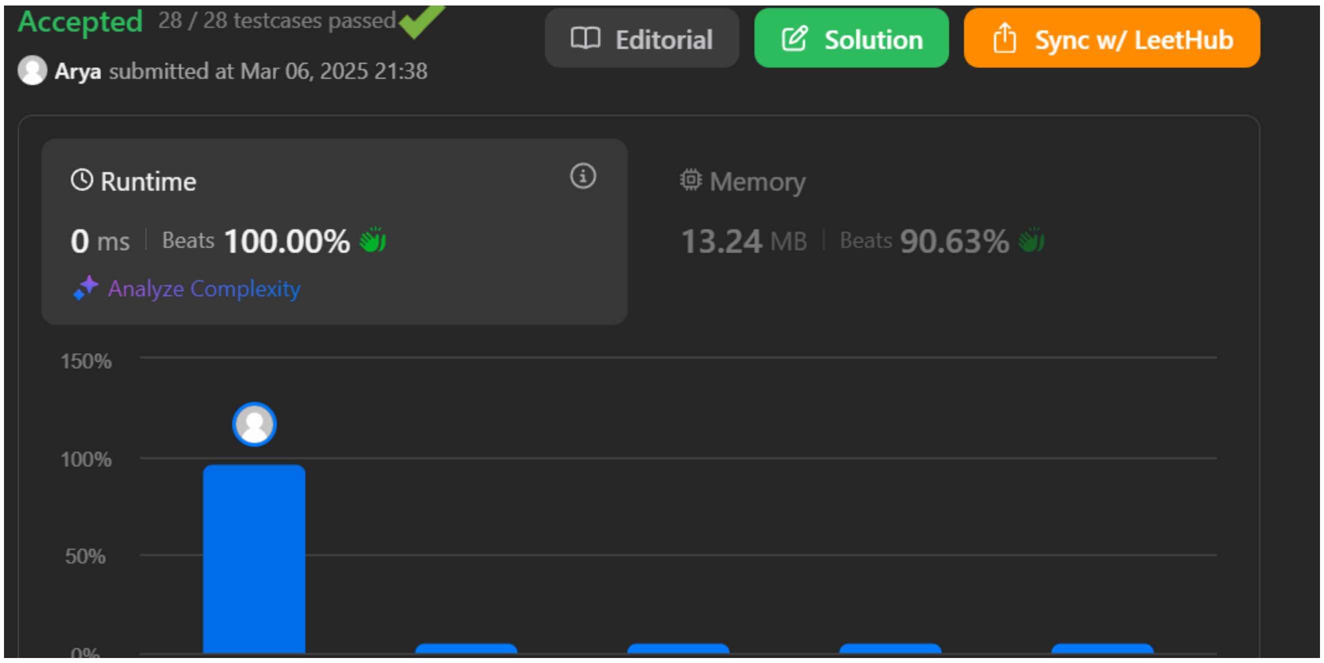
Memory

15.98 MB | Beats 99.29%

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

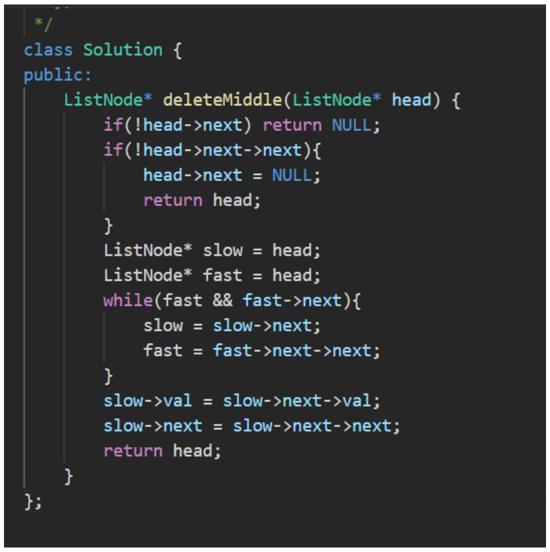


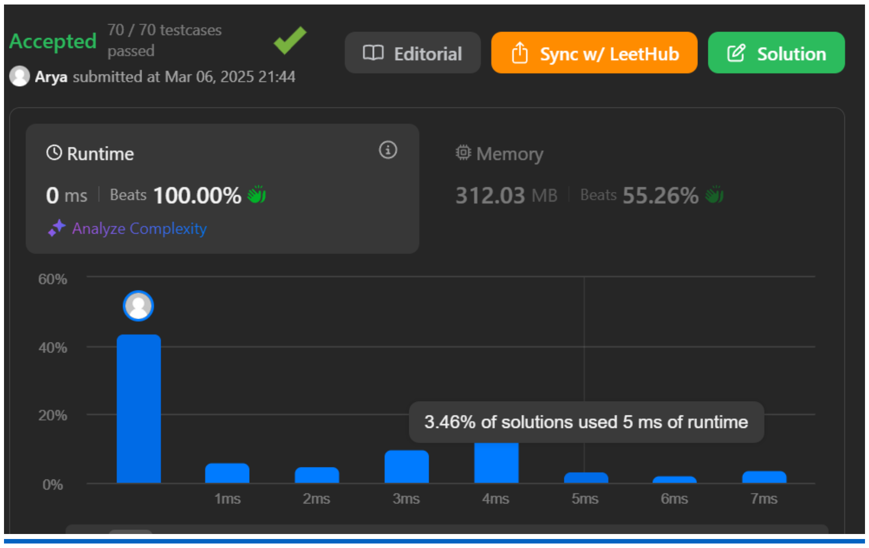
```
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode *nextNode, *prevNode = NULL;
        while (head) {
            nextNode = head->next;
            head->next = prevNode;
            prevNode = head;
            head = nextNode;
        }
        return prevNode;
    }
};
```


Accepted 28 / 28 testcases passed

Arya submitted at Mar 06, 2025 21:38

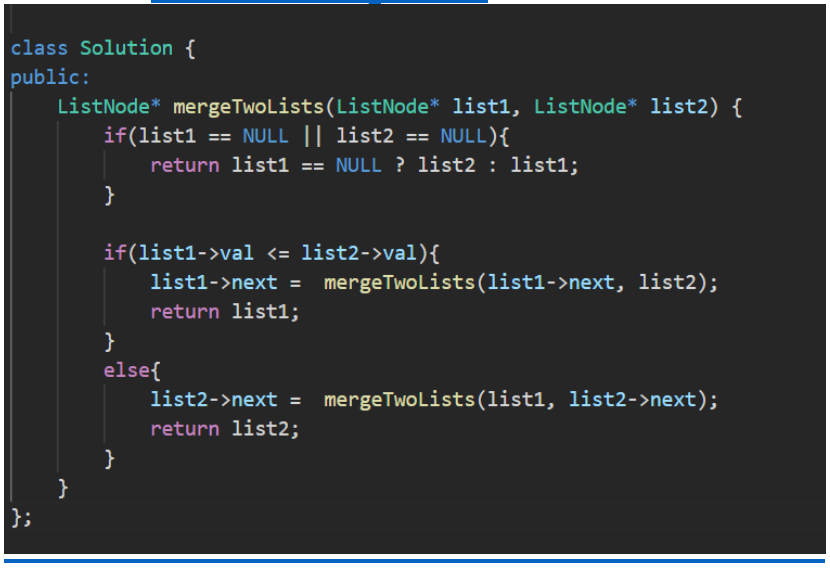
4. **Delete middle node of a list: https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/**


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

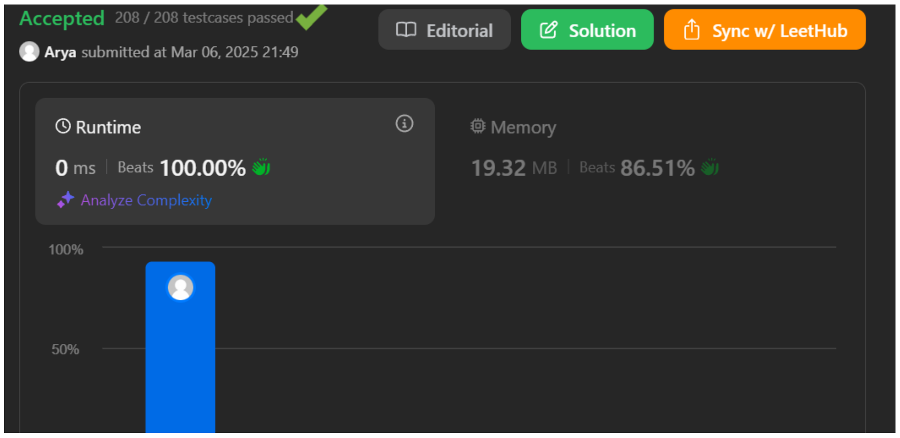

Accepted 70 / 70 testcases passed

Arya submitted at Mar 06, 2025 21:44

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

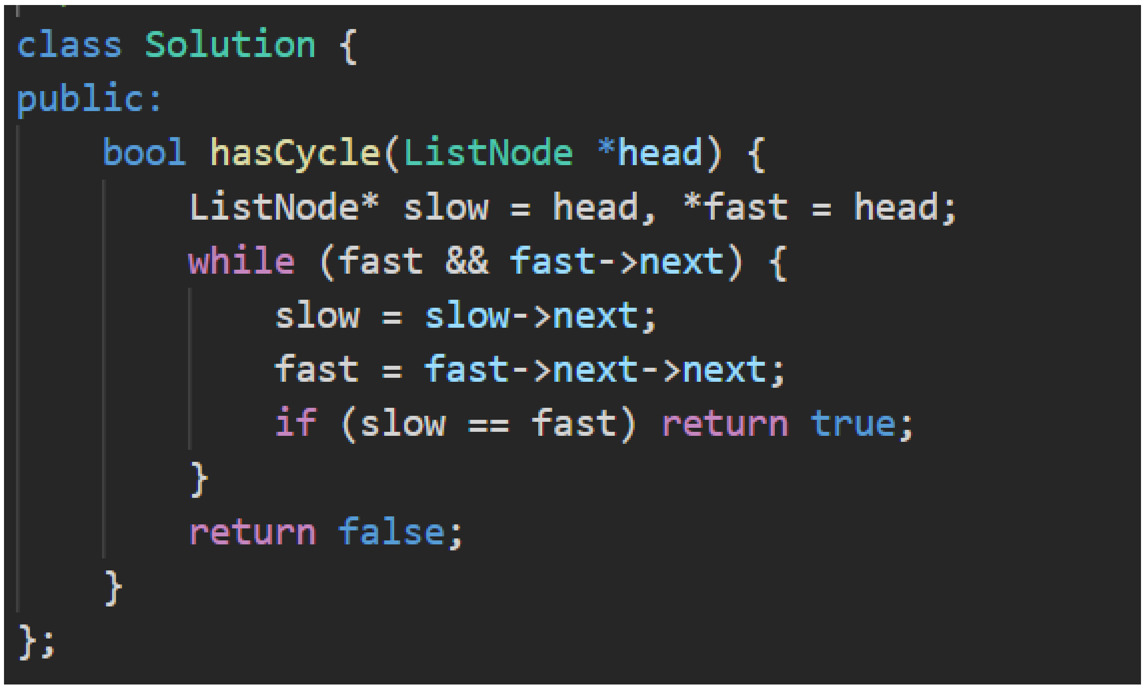


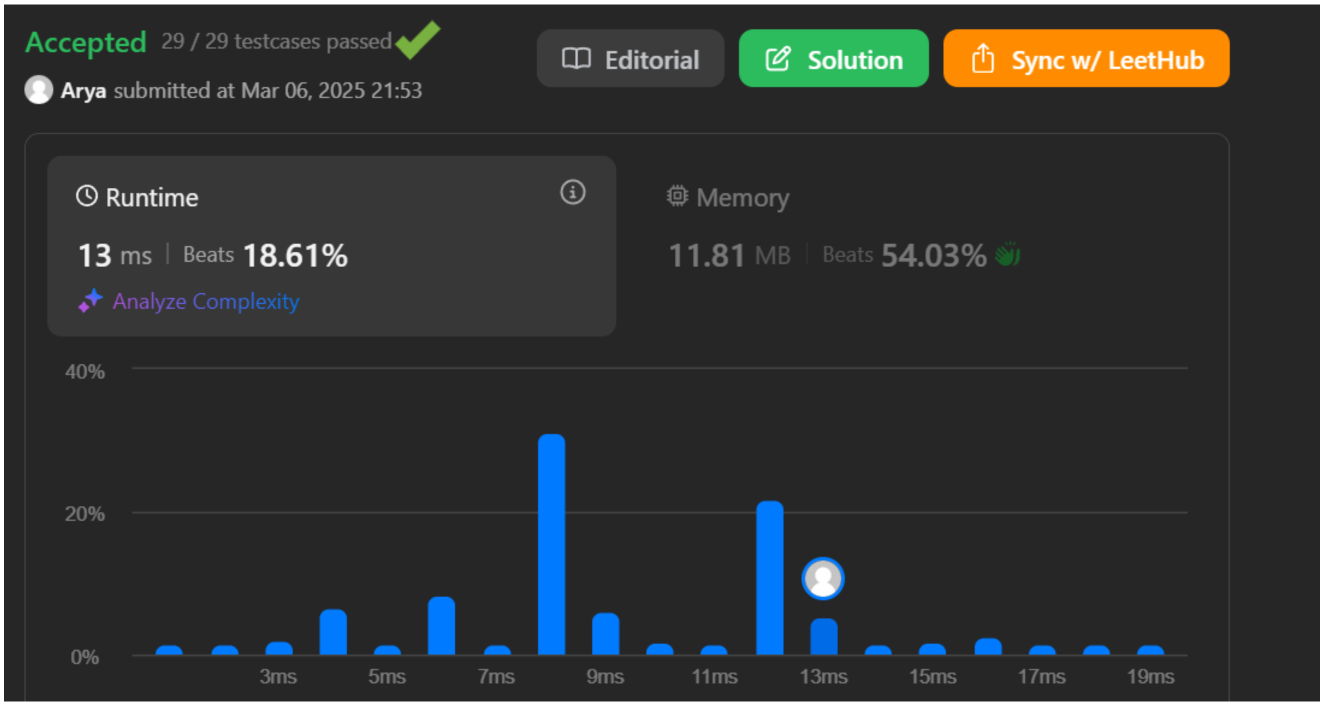
```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if(list1 == NULL || list2 == NULL){
            return list1 == NULL ? list2 : list1;
        }

        if(list1->val <= list2->val){
            list1->next =  mergeTwoLists(list1->next, list2);
            return list1;
        }
        else{
            list2->next =  mergeTwoLists(list1, list2->next);
            return list2;
        }
    }
};
```


Accepted 208 / 208 testcases passed

Arya submitted at Mar 06, 2025 21:49

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


```
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 29 / 29 testcases passed

Arya submitted at Mar 06, 2025 21:53

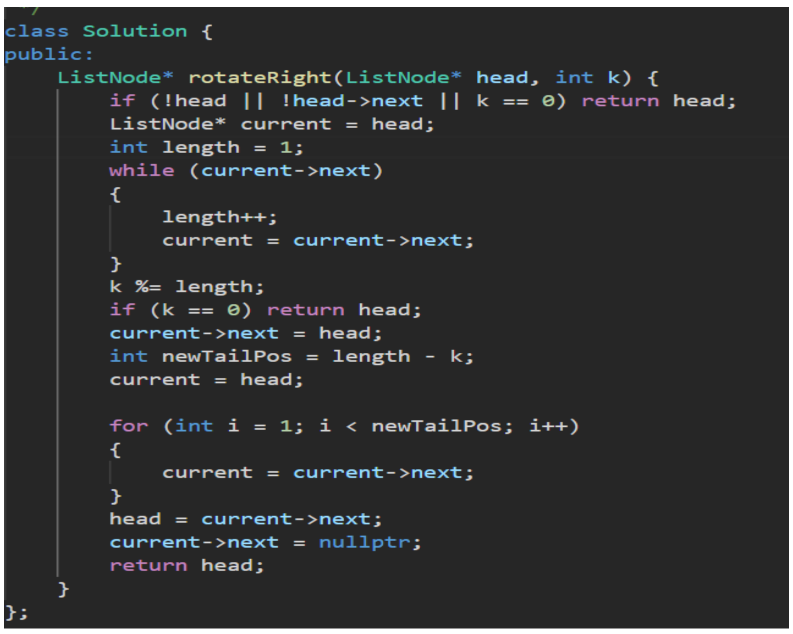
Runtime

13 ms | Beats 18.61%

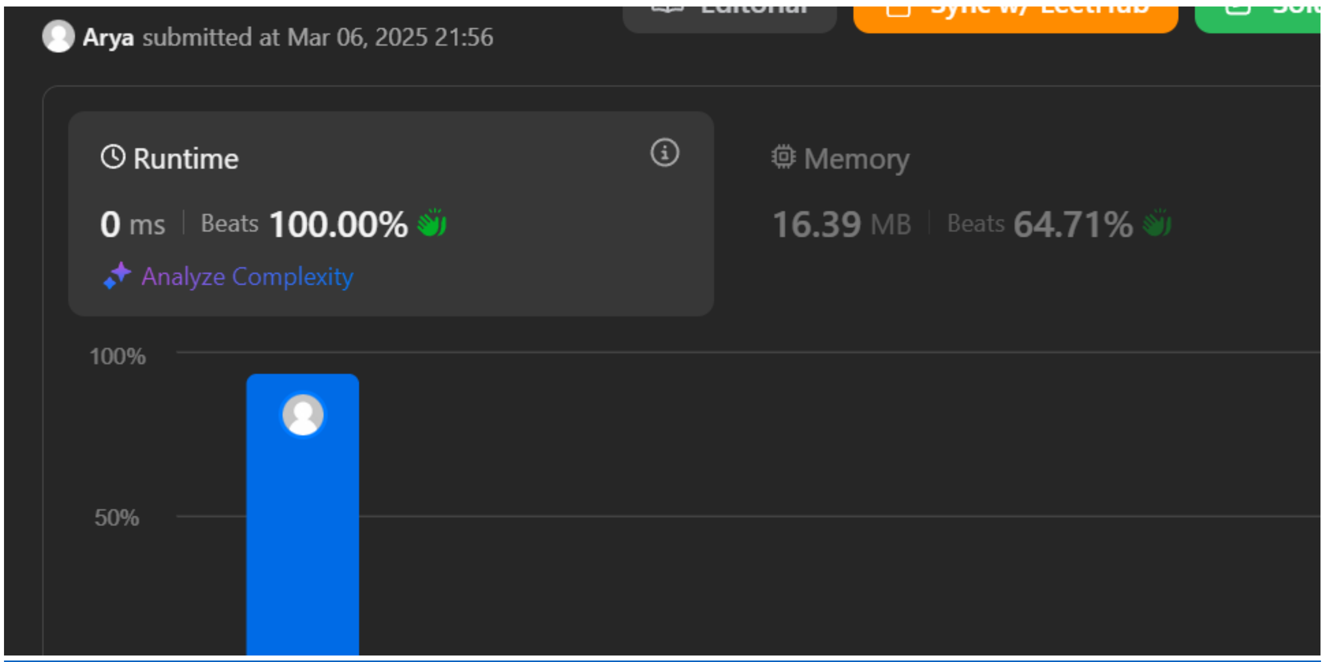
Memory

11.81 MB | Beats 54.03%

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



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

        for (int i = 1; i < newTailPos; i++)
        {
            current = current->next;
        }
        head = current->next;
        current->next = nullptr;
        return head;
    }
};
```


Arya submitted at Mar 06, 2025 21:56

Runtime

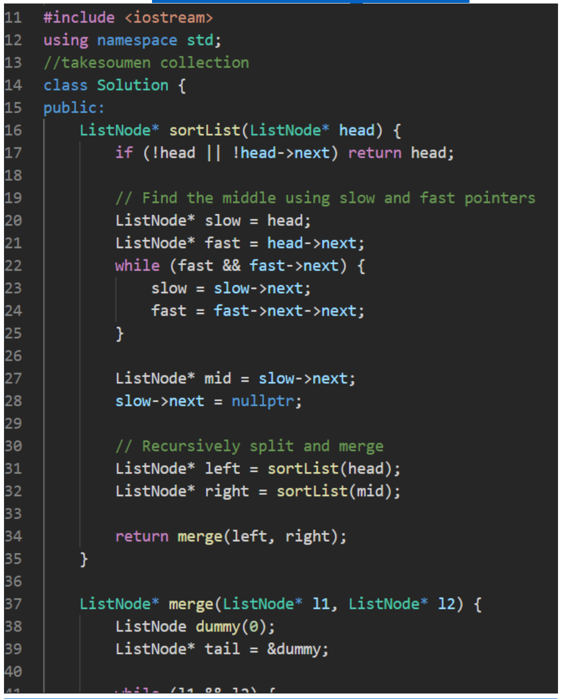
0 ms | Beats 100.00%

Analyze Complexity

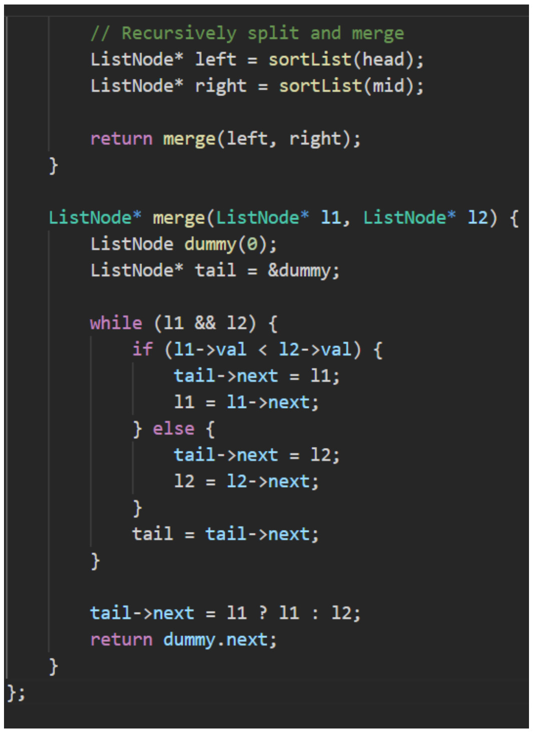
Memory

16.39 MB | Beats 64.71%

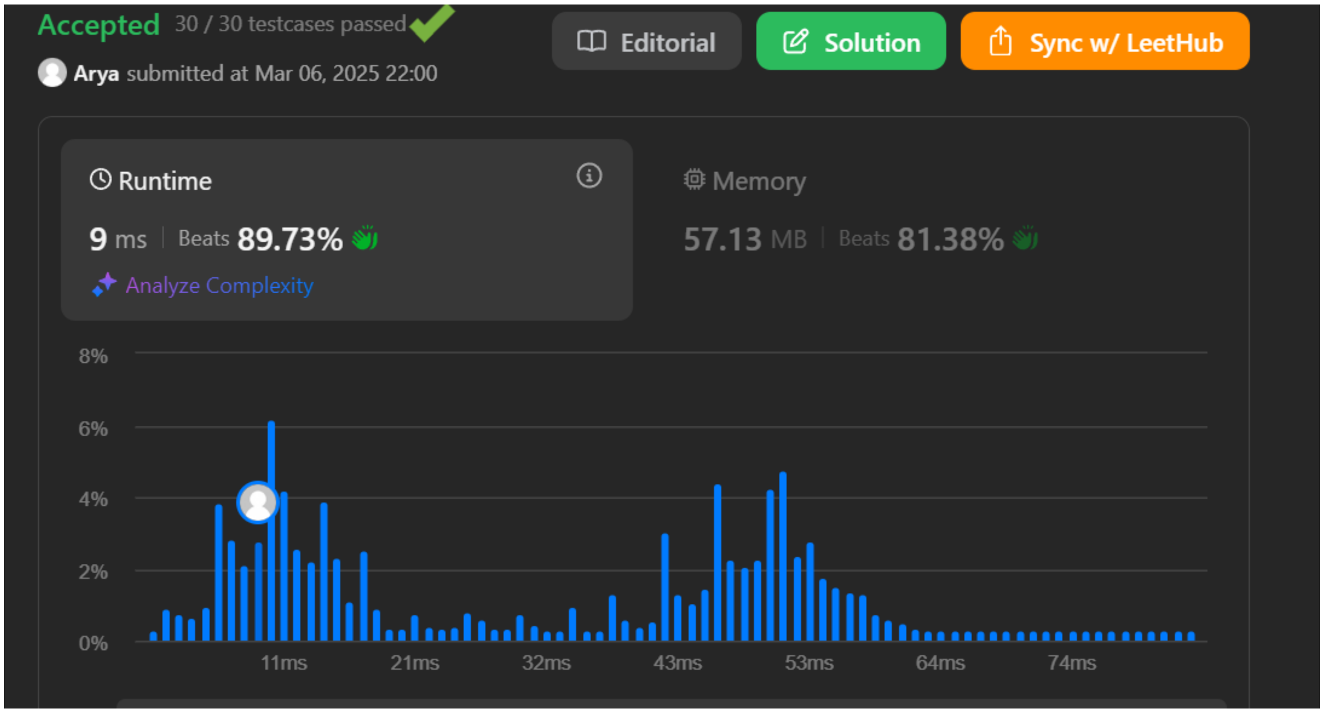
8. **Sort List: https://leetcode.com/problems/sort-list/description/**



```cpp
#include <iostream>
using namespace std;
//takesoumen collection
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) return head;

        // Find the middle using slow and fast pointers
        ListNode* slow = head;
        ListNode* fast = head->next;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
        }

        ListNode* mid = slow->next;
        slow->next = nullptr;

        // Recursively split and merge
        ListNode* left = sortList(head);
        ListNode* right = sortList(mid);

        return merge(left, right);
    }

    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;

```



```
        // Recursively split and merge
        ListNode* left = sortList(head);
        ListNode* right = sortList(mid);

        return merge(left, right);
    }

    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;
    }
};
```



Accepted 30 / 30 testcases passed

Arya submitted at Mar 06, 2025 22:00

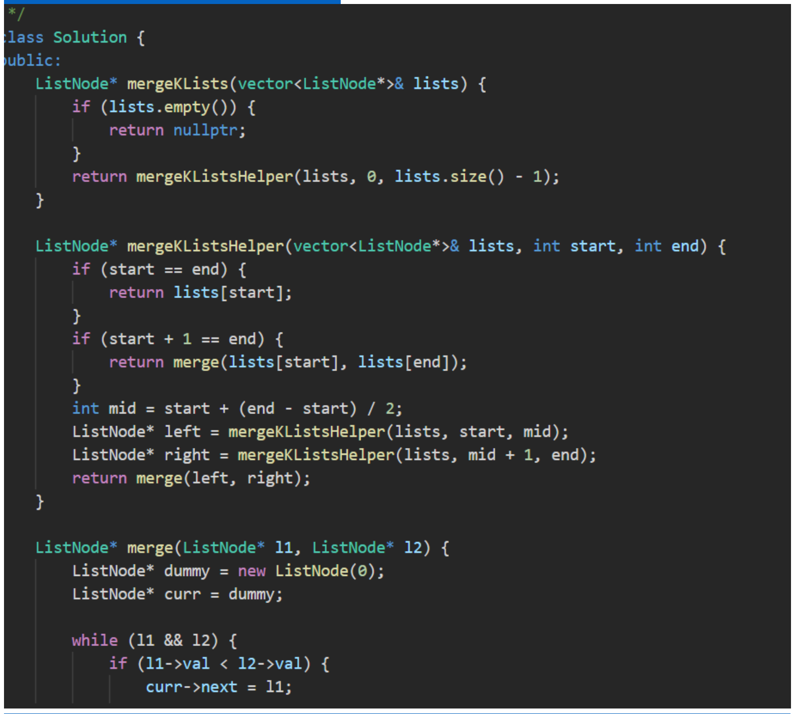
Runtime

9 ms | Beats 89.73%

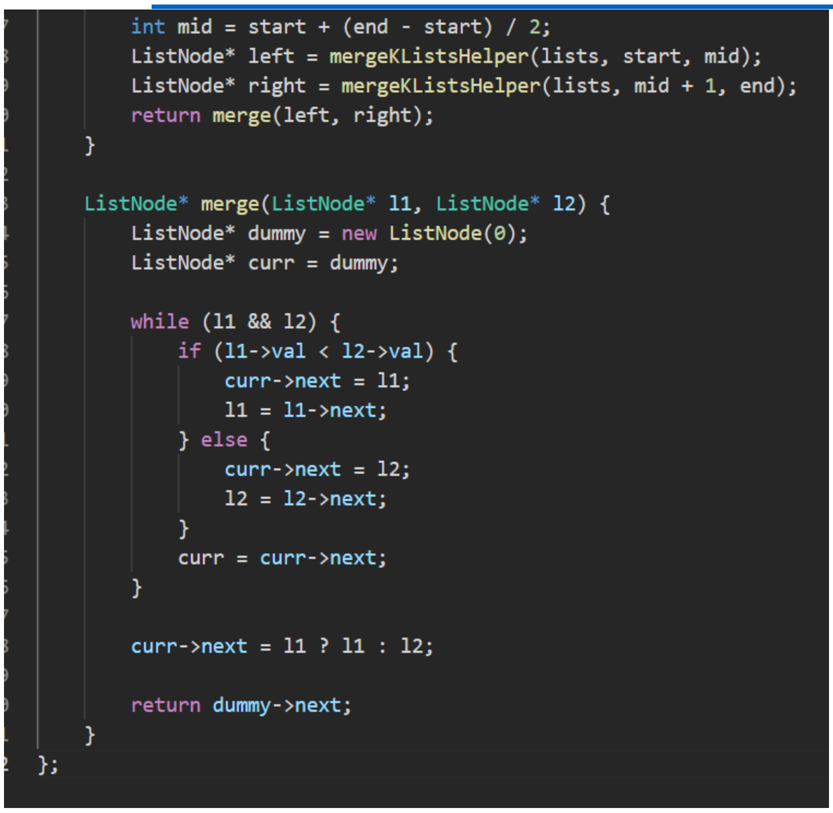
Memory

57.13 MB | Beats 81.38%

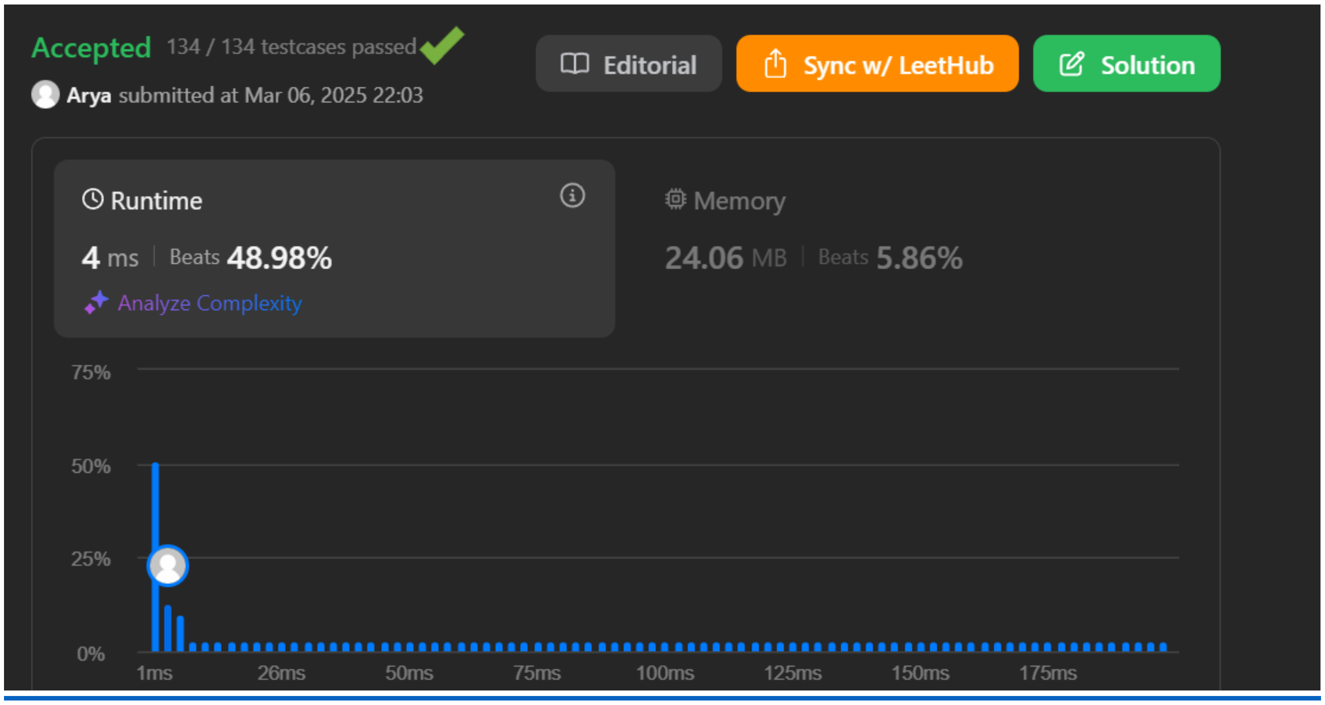
9. **Merge k sorted lists: https://leetcode.com/problems/merge-k-sorted-lists/description/**



```
*/
lass Solution {
ublic:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) {
            return nullptr;
        }
        return mergeKListsHelper(lists, 0, lists.size() - 1);
    }

    ListNode* mergeKListsHelper(vector<ListNode*>& lists, int start, int end) {
        if (start == end) {
            return lists[start];
        }
        if (start + 1 == end) {
            return merge(lists[start], lists[end]);
        }
        int mid = start + (end - start) / 2;
        ListNode* left = mergeKListsHelper(lists, start, mid);
        ListNode* right = mergeKListsHelper(lists, mid + 1, end);
        return merge(left, right);
    }

    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode* dummy = new ListNode(0);
        ListNode* curr = dummy;

        while (l1 && l2) {
            if (l1->val < l2->val) {
                curr->next = l1;
```



```
        int mid = start + (end - start) / 2;
        ListNode* left = mergeKListsHelper(lists, start, mid);
        ListNode* right = mergeKListsHelper(lists, mid + 1, end);
        return merge(left, right);
    }

    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode* dummy = new ListNode(0);
        ListNode* curr = dummy;

        while (l1 && l2) {
            if (l1->val < l2->val) {
                curr->next = l1;
                l1 = l1->next;
            } else {
                curr->next = l2;
                l2 = l2->next;
            }
            curr = curr->next;
        }

        curr->next = l1 ? l1 : l2;

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



Accepted 134 / 134 testcases passed

Arya submitted at Mar 06, 2025 22:03

Runtime

4 ms | Beats 48.98%

Memory

24.06 MB | Beats 5.86%