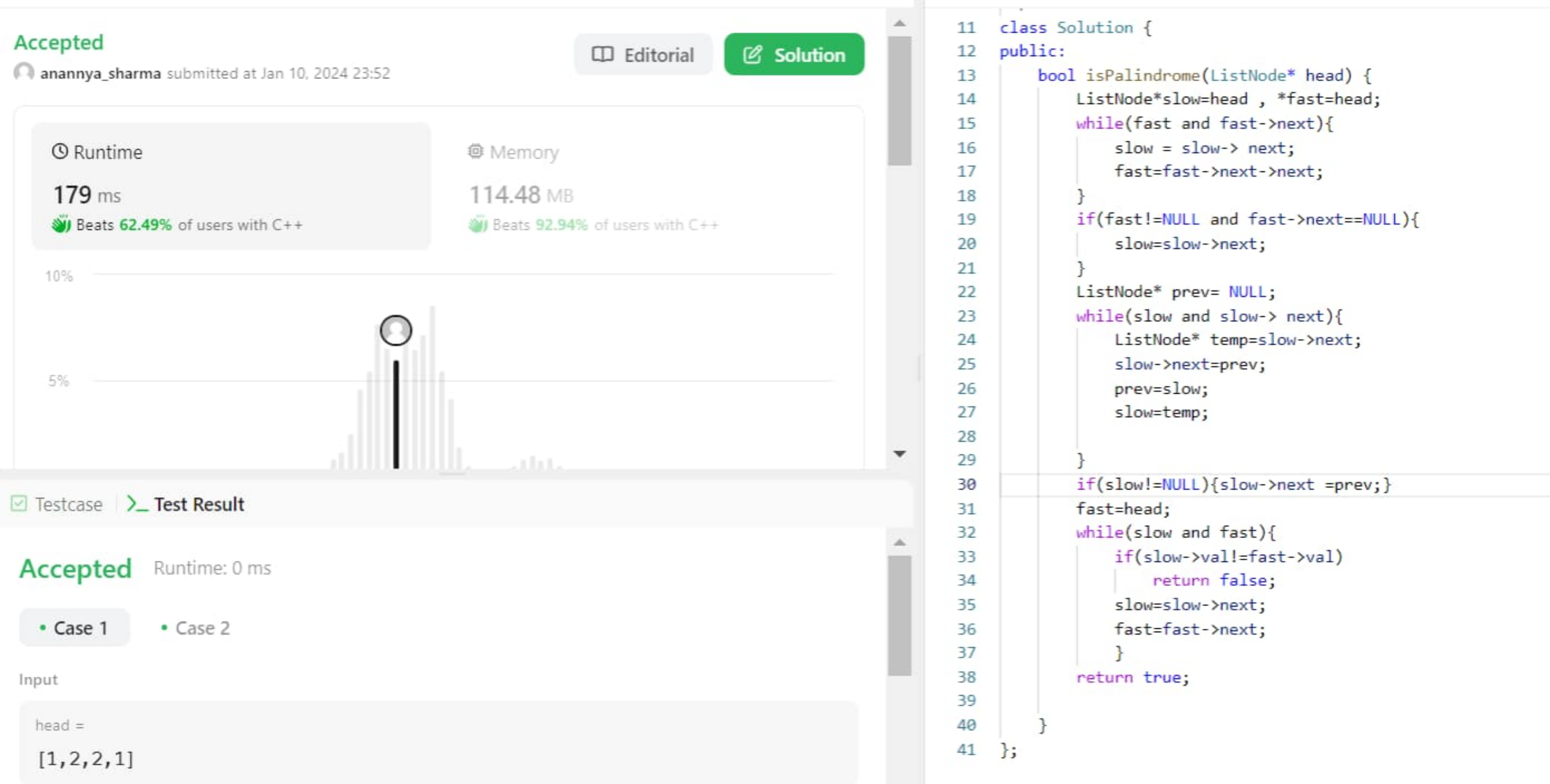


Accepted

anannya_sharma submitted at Jan 10, 2024 23:52

Runtime

179 ms

Beats 62.49% of users with C++

Memory

114.48 MB

Beats 92.94% of users with C++

Testcase | Test Result

Accepted Runtime: 0 ms

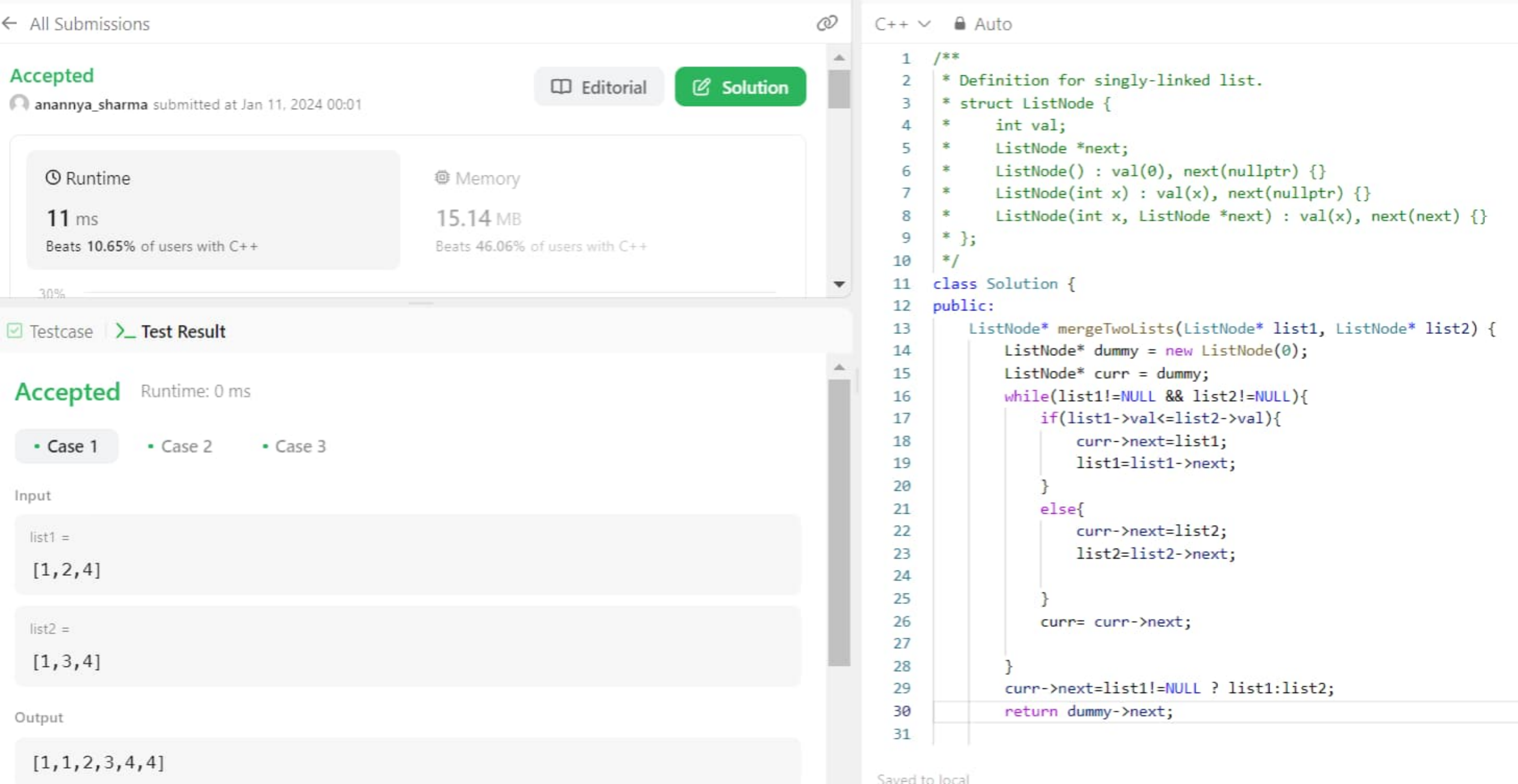
- Case 1
- Case 2

Input

head =

[1,2,2,1]

```
class Solution {
public:
    bool isPalindrome(ListNode* head) {
        ListNode*slow=head , *fast=head;
        while(fast and fast->next){
            slow = slow-> next;
            fast=fast->next->next;
        }
        if(fast!=NULL and fast->next==NULL){
            slow=slow->next;
        }
        ListNode* prev= NULL;
        while(slow and slow-> next){
            ListNode* temp=slow->next;
            slow->next=prev;
            prev=slow;
            slow=temp;

        }
        if(slow!=NULL){slow->next =prev;}
        fast=head;
        while(slow and fast){
            if(slow->val!=fast->val)
                return false;
            slow=slow->next;
            fast=fast->next;
            }
        return true;

    }
};
```



Accepted

anannya_sharma submitted at Jan 11, 2024 00:01

Runtime

11 ms

Beats 10.65% of users with C++

Memory

15.14 MB

Beats 46.06% of users with C++

Accepted Runtime: 0 ms

Input

list1 =

[1,2,4]

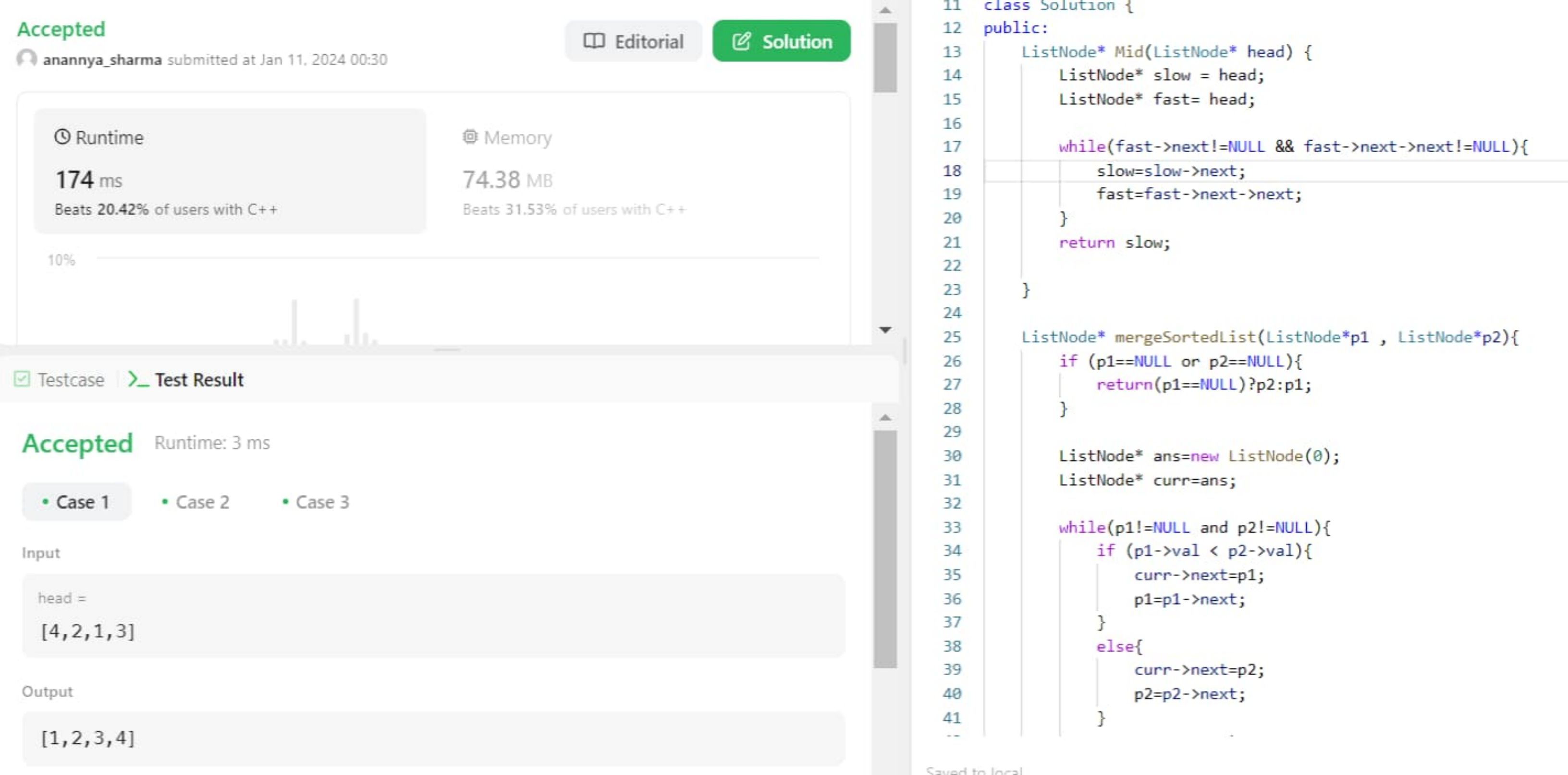
list2 =

[1,3,4]

Output

[1,1,2,3,4,4]

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

            }
            curr= curr->next;

        }
        curr->next=list1!=NULL ? list1:list2;
        return dummy->next;

```



Accepted

anannya_sharma submitted at Jan 11, 2024 00:30

Runtime

174 ms

Beats 20.42% of users with C++

Memory

74.38 MB

Beats 31.53% of users with C++

Accepted Runtime: 3 ms

Input

head =

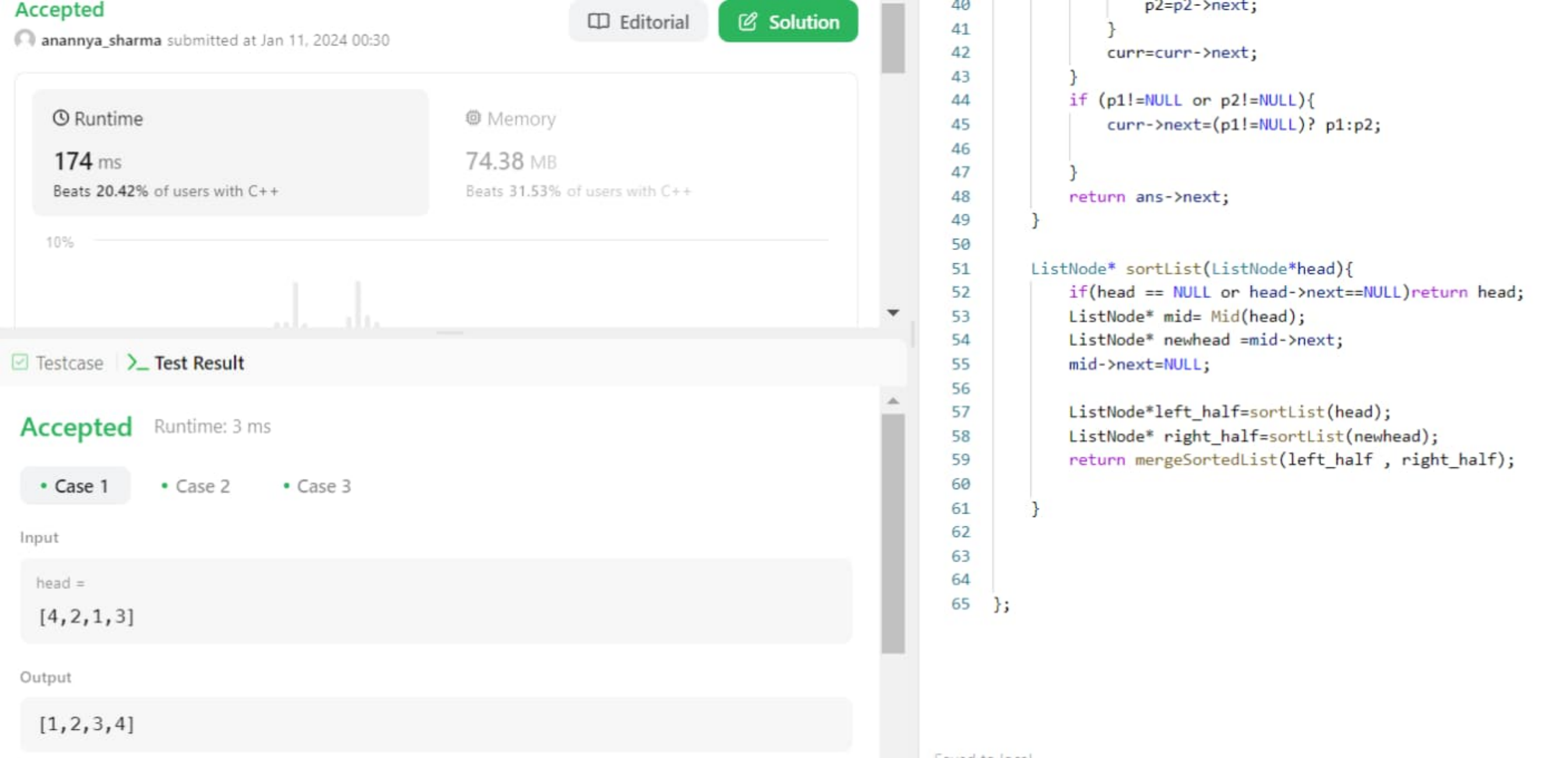
```
[4,2,1,3]
```

Output

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

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

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

    }

    ListNode* mergeSortedList(ListNode*p1 , ListNode*p2){
        if (p1==NULL or p2==NULL){
            return(p1==NULL)?p2:p1;
        }

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

        while(p1!=NULL and p2!=NULL){
            if (p1->val < p2->val){
                curr->next=p1;
                p1=p1->next;
            }
            else{
                curr->next=p2;
                p2=p2->next;
            }
```

Saved to local



Accepted

anannya_sharma submitted at Jan 11, 2024 00:30

Runtime

174 ms

Beats 20.42% of users with C++

Memory

74.38 MB

Beats 31.53% of users with C++

Accepted Runtime: 3 ms

Input

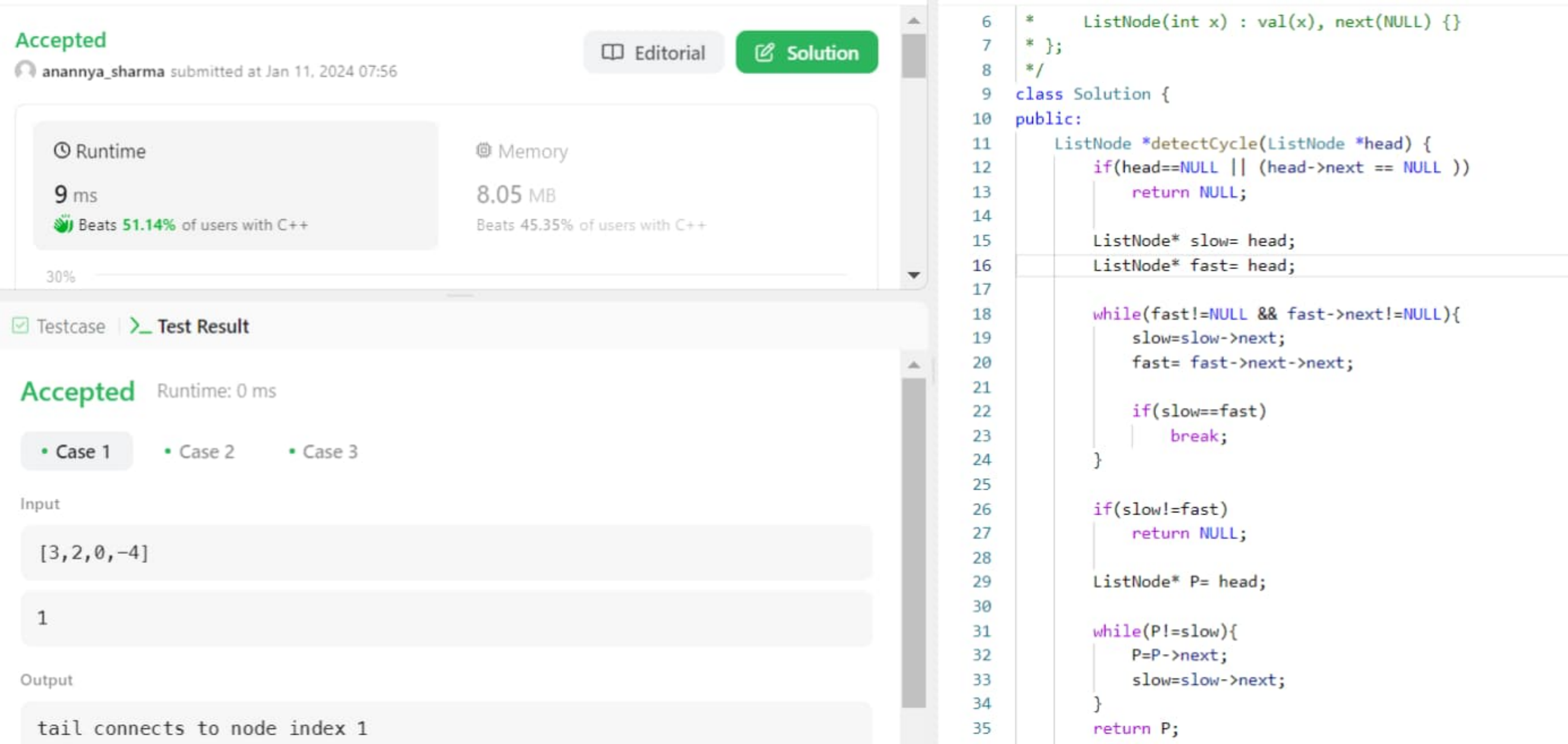
head =

[4,2,1,3]

Output

[1,2,3,4]

```
                p2=p2->next;
            }
            curr=curr->next;
        }
        if (p1!=NULL or p2!=NULL){
            curr->next=(p1!=NULL)? p1:p2;

        }
        return ans->next;
    }

    ListNode* sortList(ListNode*head){
        if(head == NULL or head->next==NULL)return head;
        ListNode* mid= Mid(head);
        ListNode* newhead =mid->next;
        mid->next=NULL;

        ListNode*left_half=sortList(head);
        ListNode* right_half=sortList(newhead);
        return mergeSortedList(left_half , right_half);

    }



};
```



Accepted

anannya_sharma submitted at Jan 11, 2024 07:56

Runtime

9 ms

Beats 51.14% of users with C++

Memory

8.05 MB

Beats 45.35% of users with C++

Accepted Runtime: 0 ms

Input

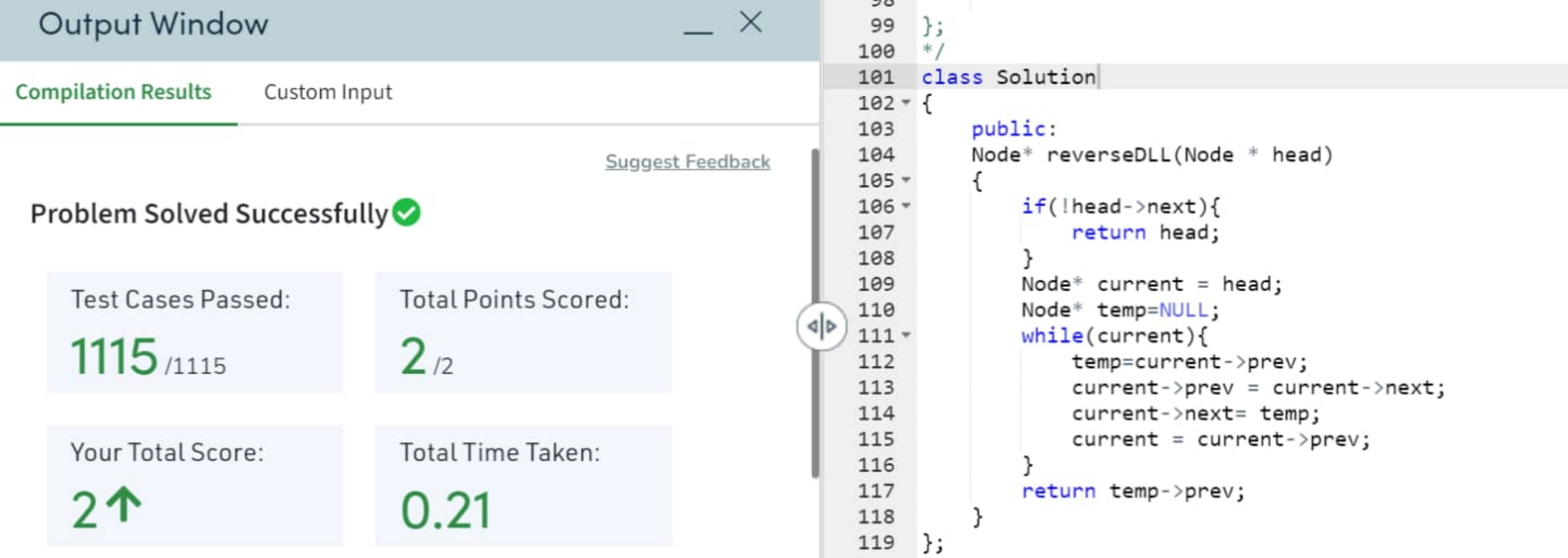
```
[3,2,0,-4]
```

```
1
```

Output

```
tail connects to node index 1
```

```
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        if(head==NULL || (head->next == NULL ))
            return NULL;

        ListNode* slow= head;
        ListNode* fast= head;

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

            if(slow==fast)
                break;
        }

        if(slow!=fast)
            return NULL;

        ListNode* P= head;

        while(P!=slow){
            P=P->next;
            slow=slow->next;
        }
        return P;
```



Problem Solved Successfully ✓

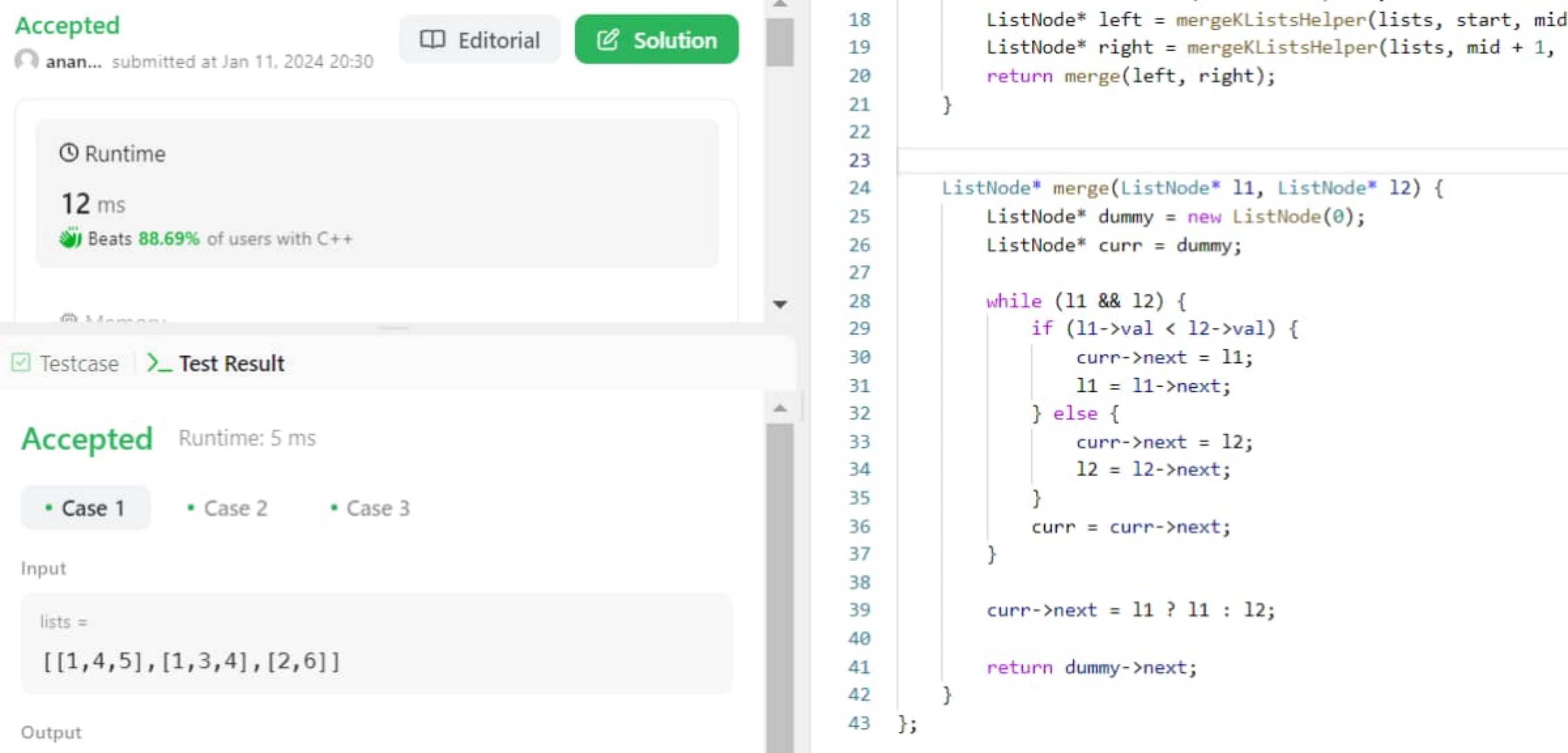
Test Cases Passed: 1115 /1115

Total Points Scored: 2 /2

Your Total Score: 2 ↑

Total Time Taken: 0.21

```
};
*/
class Solution
{
    public:
    Node* reverseDLL(Node * head)
    {
        if(!head->next){
            return head;
        }
        Node* current = head;
        Node* temp=NULL;
        while(current){
            temp=current->prev;
            current->prev = current->next;
            current->next= temp;
            current = current->prev;
        }
        return temp->prev;
    }
};
```



Accepted

anan... submitted at Jan 11, 2024 20:30

Runtime

12 ms

Beats 88.69% of users with C++

Memory

Accepted Runtime: 5 ms

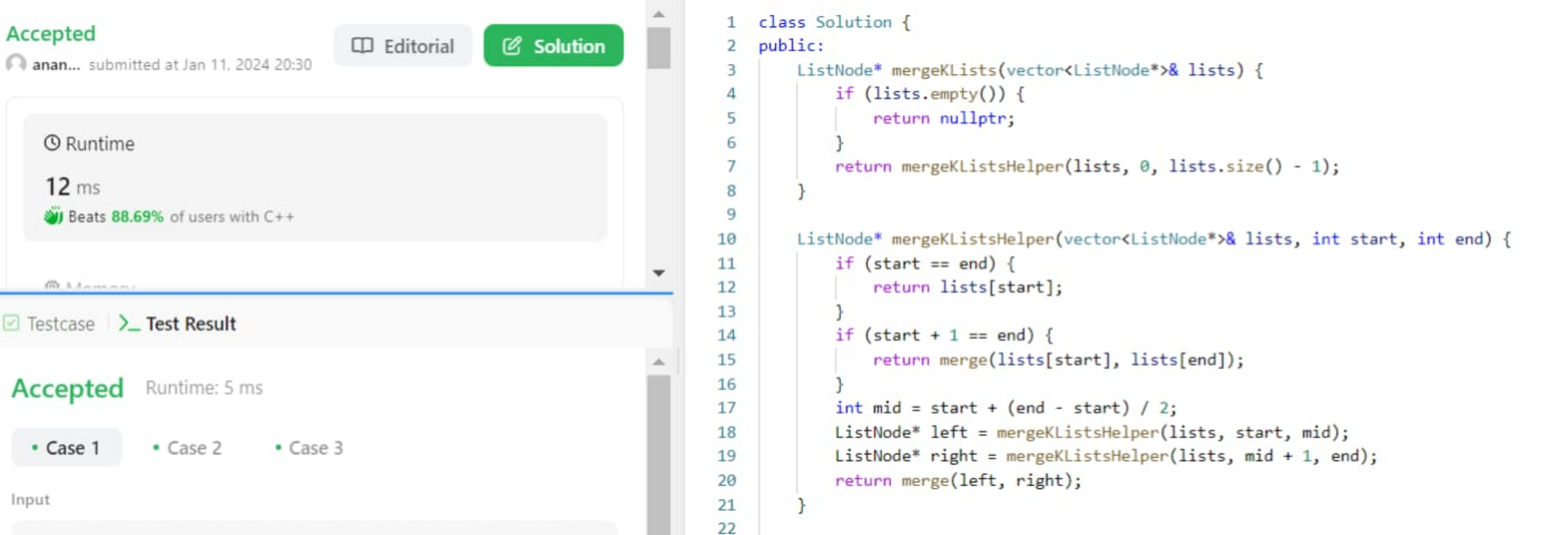
Input

lists =

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

Output

```cpp
        ListNode* left = mergeKListsHelper(lists, start, mid
        ListNode* right = mergeKListsHelper(lists, mid + 1,
        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

anan... submitted at Jan 11, 2024 20:30

Runtime

12 ms

Beats 88.69% of users with C++

Memory

Accepted Runtime: 5 ms

Input

```cpp
class Solution {
public:
    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);
    }

```