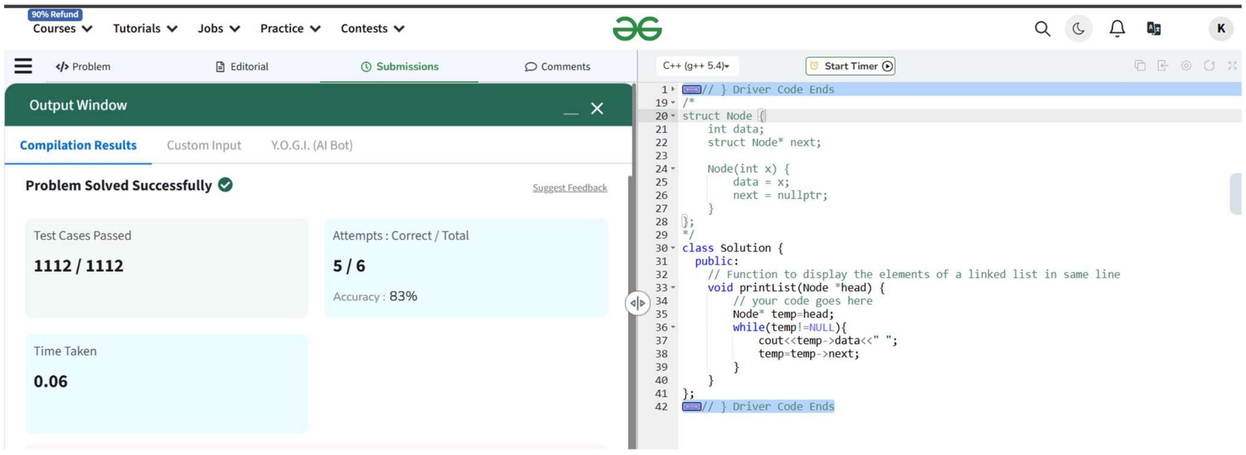
Name: Shivam Kumar

Uid: 22BCS11736

Q1} [Print linked list]()



Problem Solved Successfully

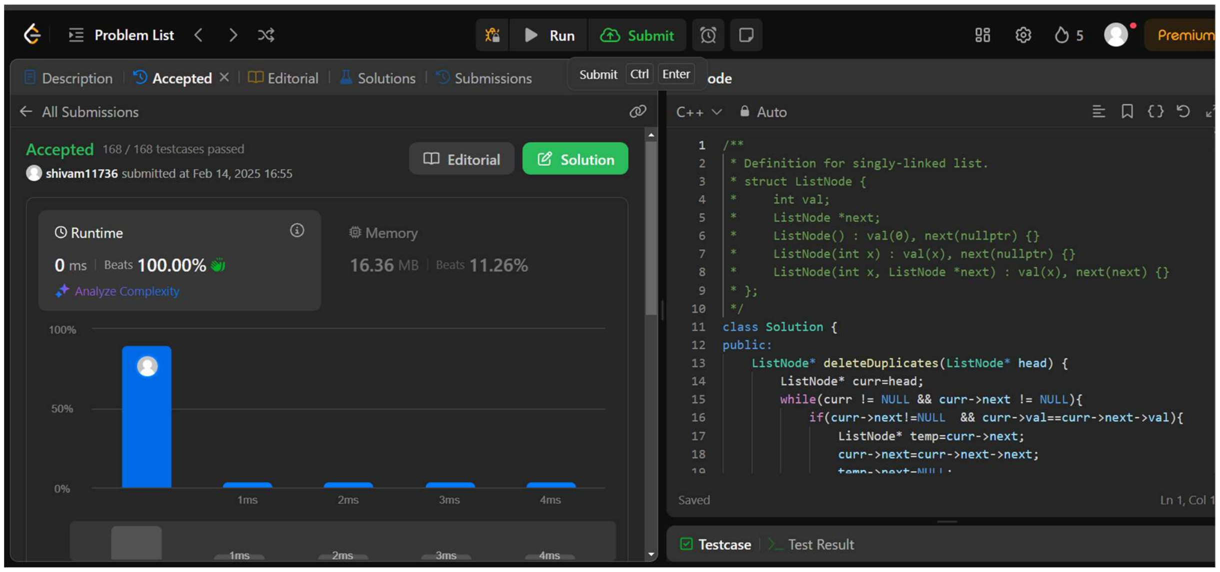
Test Cases Passed
1112 / 1112

Attempts : Correct / Total
5 / 6
Accuracy : 83%

Time Taken
0.06

```cpp
// } Driver Code Ends
/*
struct Node {
    int data;
    struct Node* next;

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

Q2} [Remove duplicates from a sorted list]()



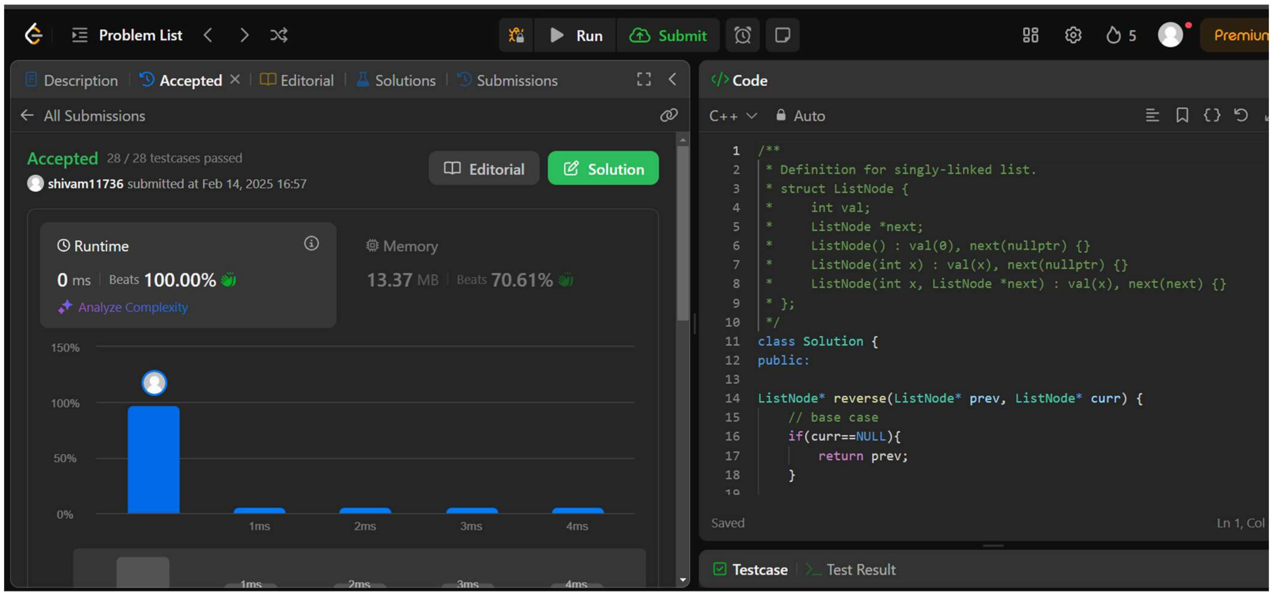
Accepted 168 / 168 testcases passed
shivam11736 submitted at Feb 14, 2025 16:55

Runtime
0 ms | Beats 100.00%

Memory
16.36 MB | Beats 11.26%

```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* deleteDuplicates(ListNode* head) {
        ListNode* curr=head;
        while(curr != NULL && curr->next != NULL){
            if(curr->next!=NULL  && curr->val==curr->next->val){
                ListNode* temp=curr->next;
                curr->next=curr->next->next;
                temp->next=NULL;
```

Q3} Reverse a linked list] (https://leetcode.com/problems/reverse-linked-list/)



Accepted 28 / 28 testcases passed
shivam11736 submitted at Feb 14, 2025 16:57

Runtime
0 ms | Beats 100.00%

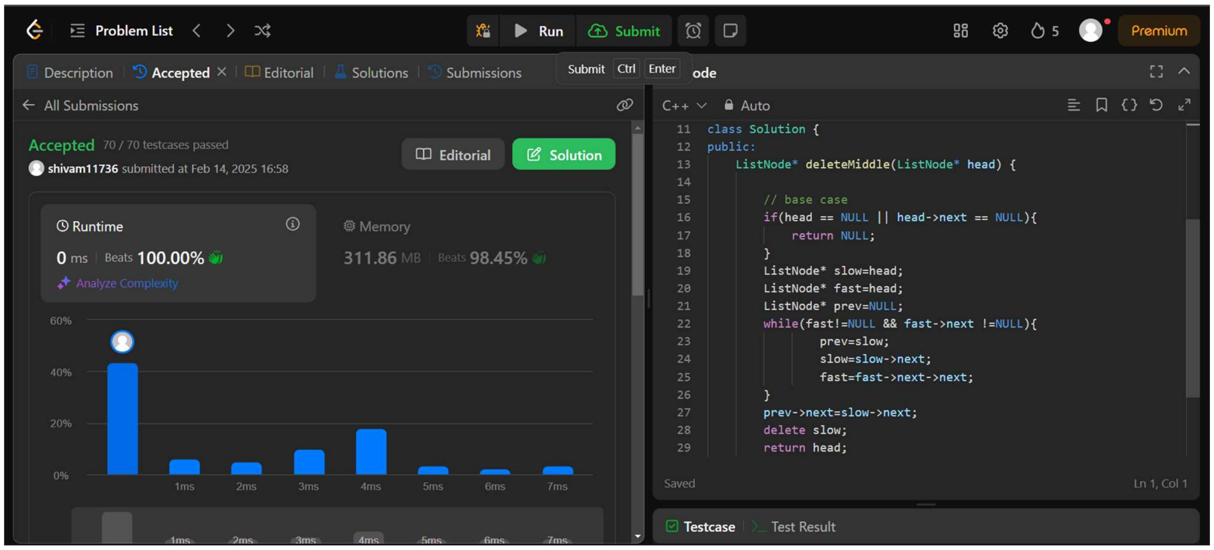
Memory
13.37 MB | Beats 70.61%

C++ Auto

```
/**
 * 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* reverse(ListNode* prev, ListNode* curr) {
    // base case
    if(curr==NULL){
        return prev;
    }
```

Saved   Ln 1, Col

Q4} Delete middle node of a list



Accepted 70 / 70 testcases passed
shivam11736 submitted at Feb 14, 2025 16:58

Runtime
0 ms | Beats 100.00%

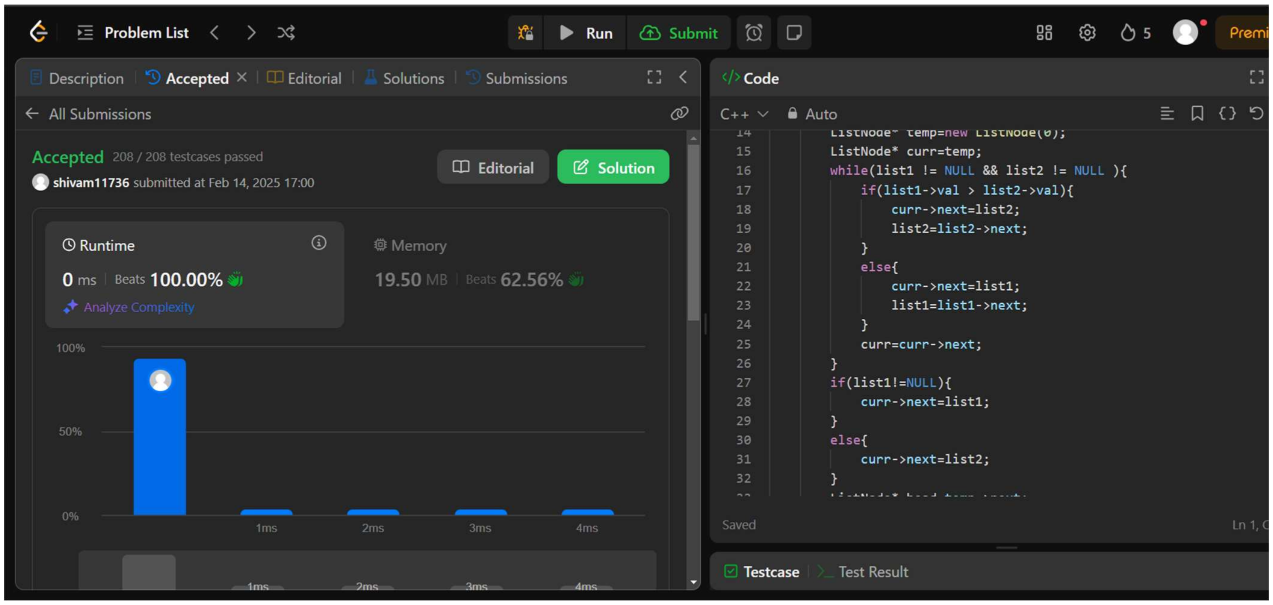
Memory
311.86 MB | Beats 98.45%

C++ Auto

```
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {

        // base case
        if(head == NULL || head->next == NULL){
            return NULL;
        }
        ListNode* slow=head;
        ListNode* fast=head;
        ListNode* prev=NULL;
        while(fast!=NULL && fast->next !=NULL){
                prev=slow;
                slow=slow->next;
                fast=fast->next->next;
        }
        prev->next=slow->next;
        delete slow;
        return head;
```

Saved   Ln 1, Col 1

Q5} Merge two sorted linked lists


Accepted 208 / 208 testcases passed

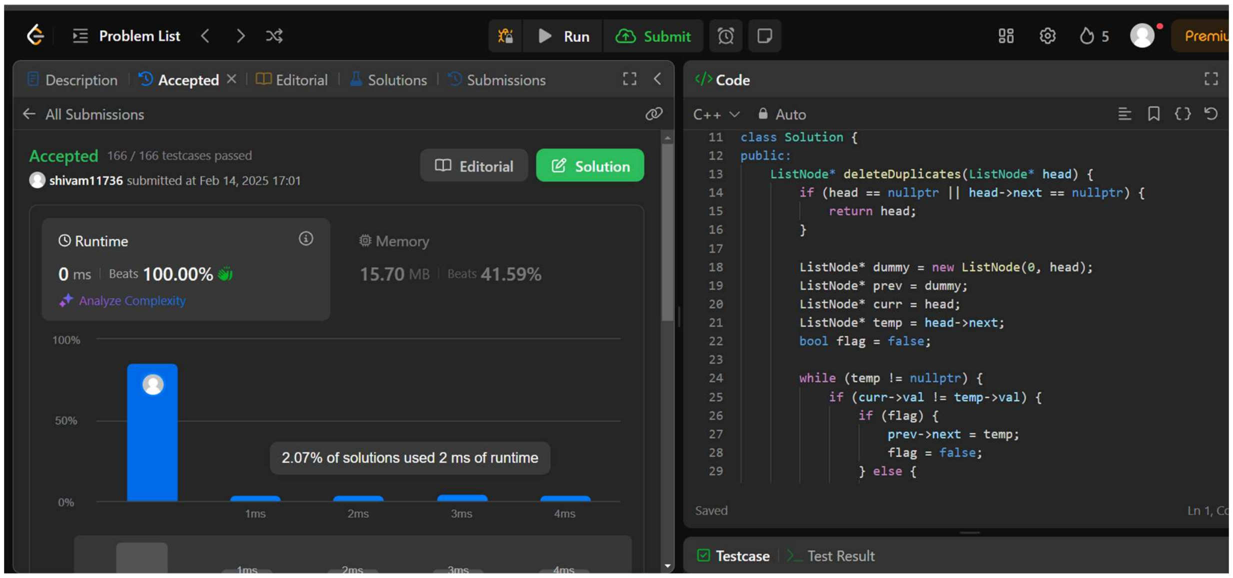
shivam11736 submitted at Feb 14, 2025 17:00

Runtime 0 ms | Beats 100.00%

Memory 19.50 MB | Beats 62.56%

```
        ListNode* temp=new ListNode(0);
        ListNode* curr=temp;
        while(list1 != NULL && list2 != NULL ){
            if(list1->val > list2->val){
                curr->next=list2;
                list2=list2->next;
            }
            else{
                curr->next=list1;
                list1=list1->next;
            }
            curr=curr->next;
        }
        if(list1!=NULL){
            curr->next=list1;
        }
        else{
            curr->next=list2;
        }
```

Q6} Remove duplicates from sorted lists 2


Accepted 166 / 166 testcases passed

shivam11736 submitted at Feb 14, 2025 17:01

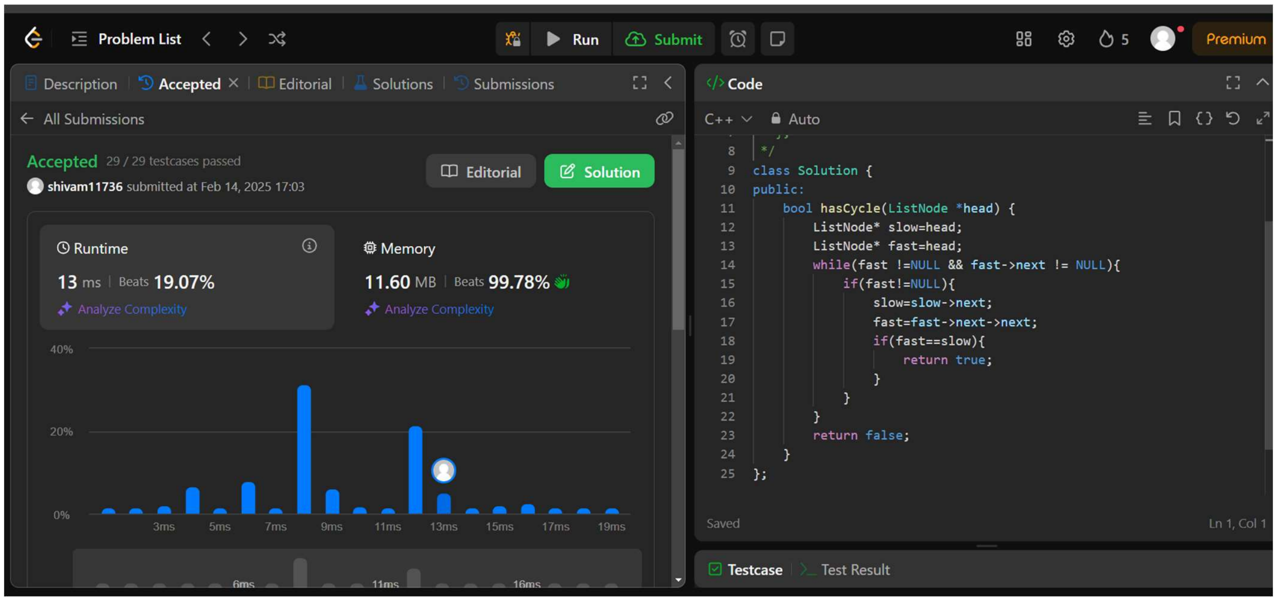
Runtime 0 ms | Beats 100.00%

Memory 15.70 MB | Beats 41.59%

2.07% of solutions used 2 ms of runtime

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (head == nullptr || head->next == nullptr) {
            return head;
        }

        ListNode* dummy = new ListNode(0, head);
        ListNode* prev = dummy;
        ListNode* curr = head;
        ListNode* temp = head->next;
        bool flag = false;

        while (temp != nullptr) {
            if (curr->val != temp->val) {
                if (flag) {
                    prev->next = temp;
                    flag = false;
                } else {
```

Q7} Detect a cycle in a linked list



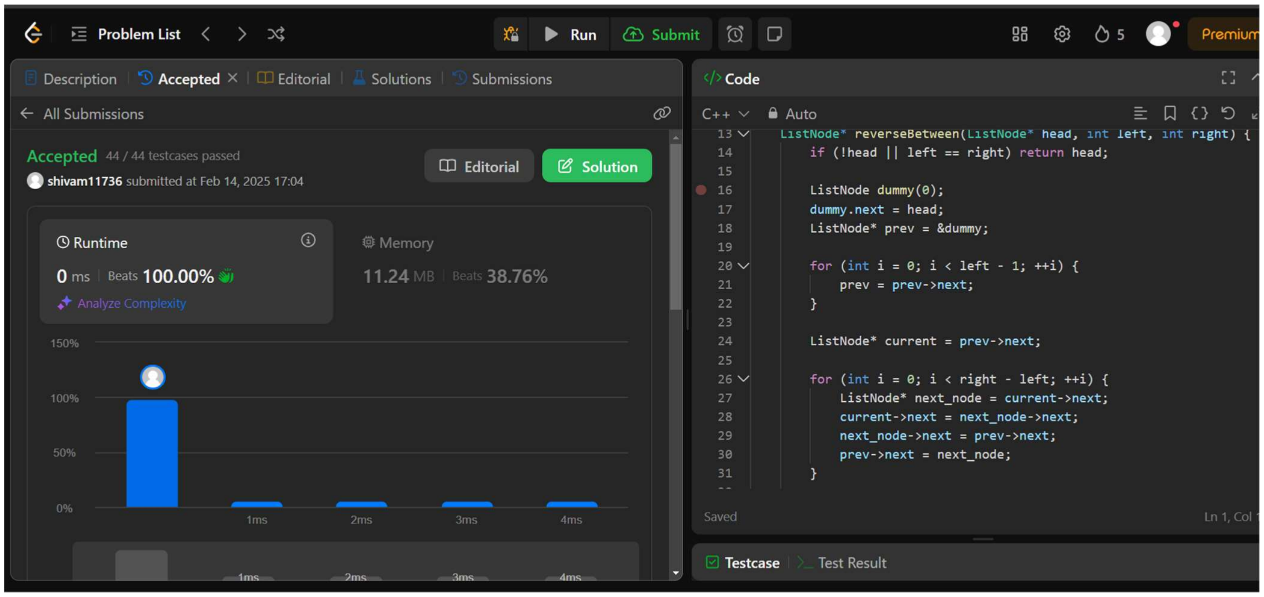
Accepted 29 / 29 testcases passed
shivam11736 submitted at Feb 14, 2025 17:03

Runtime
13 ms | Beats 19.07%

Memory
11.60 MB | Beats 99.78%

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

Q8} Reverse linked list 2



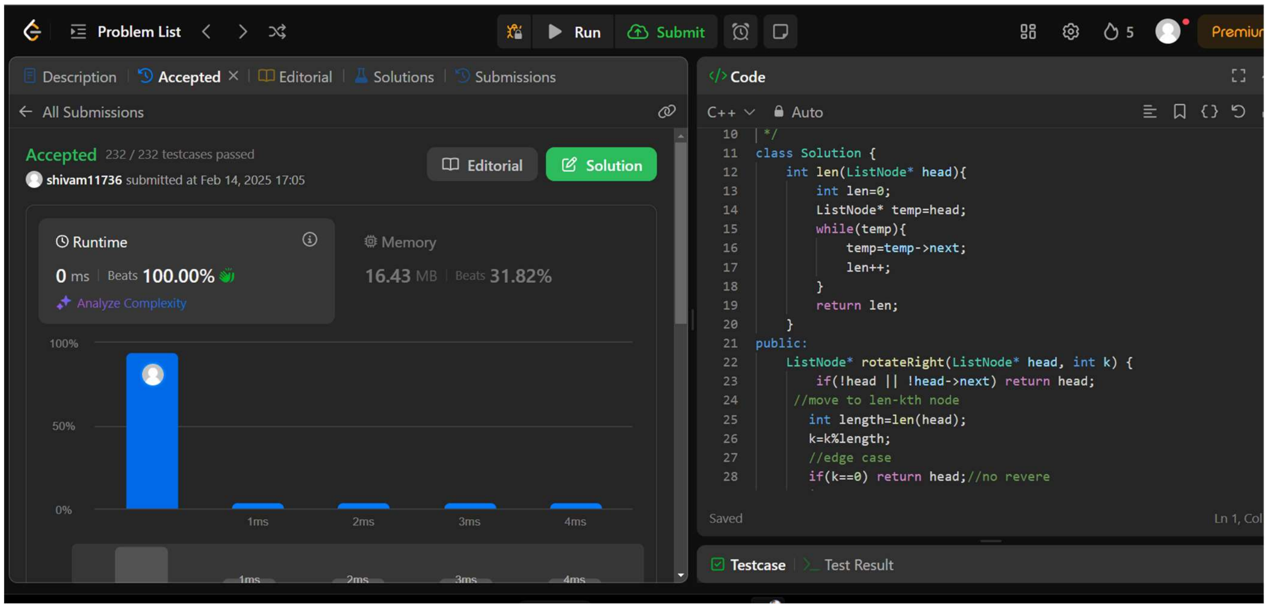
Accepted 44 / 44 testcases passed
shivam11736 submitted at Feb 14, 2025 17:04

Runtime
0 ms | Beats 100.00%

Memory
11.24 MB | Beats 38.76%

```
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (!head || left == right) return head;

        ListNode dummy(0);
        dummy.next = head;
        ListNode* prev = &dummy;

        for (int i = 0; i < left - 1; ++i) {
            prev = prev->next;
        }

        ListNode* current = prev->next;

        for (int i = 0; i < right - left; ++i) {
            ListNode* next_node = current->next;
            current->next = next_node->next;
            next_node->next = prev->next;
            prev->next = next_node;
        }
```

Q9} rotate a list



Accepted 232 / 232 testcases passed

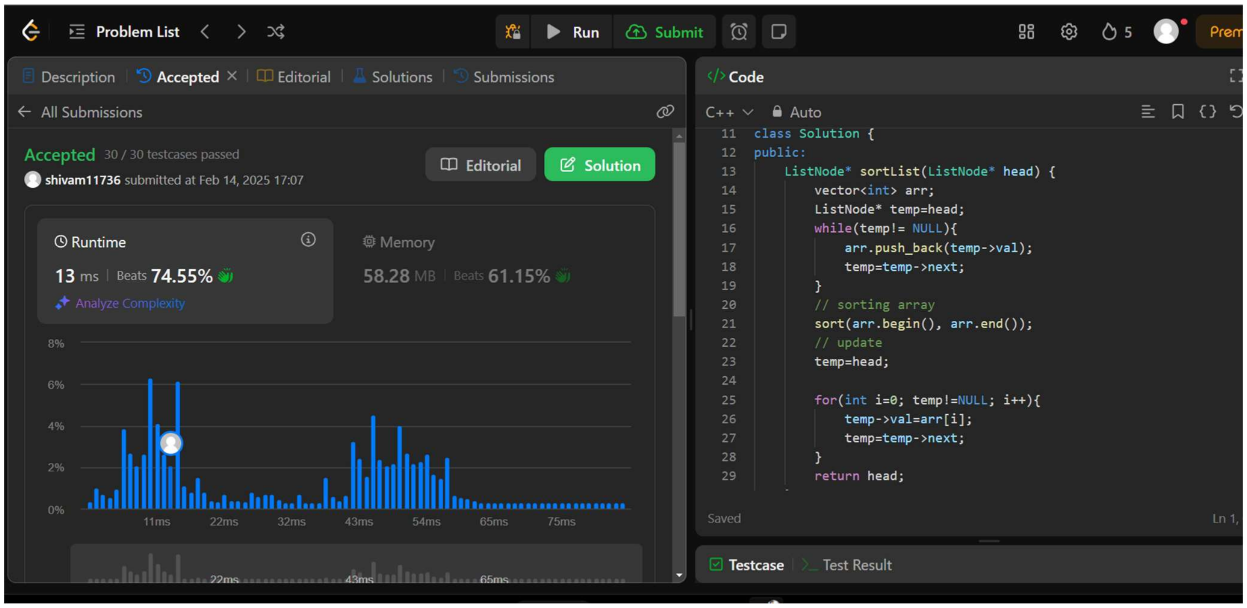
shivam11736 submitted at Feb 14, 2025 17:05

Runtime
0 ms | Beats 100.00%

Memory
16.43 MB | Beats 31.82%

```
 */
class Solution {
    int len(ListNode* head){
        int len=0;
        ListNode* temp=head;
        while(temp){
            temp=temp->next;
            len++;
        }
        return len;
    }
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if(!head || !head->next) return head;
     //move to len-kth node
       int length=len(head);
       k=k%length;
       //edge case
       if(k==0) return head;//no revere
```

Q10} Sort List



Accepted 30 / 30 testcases passed

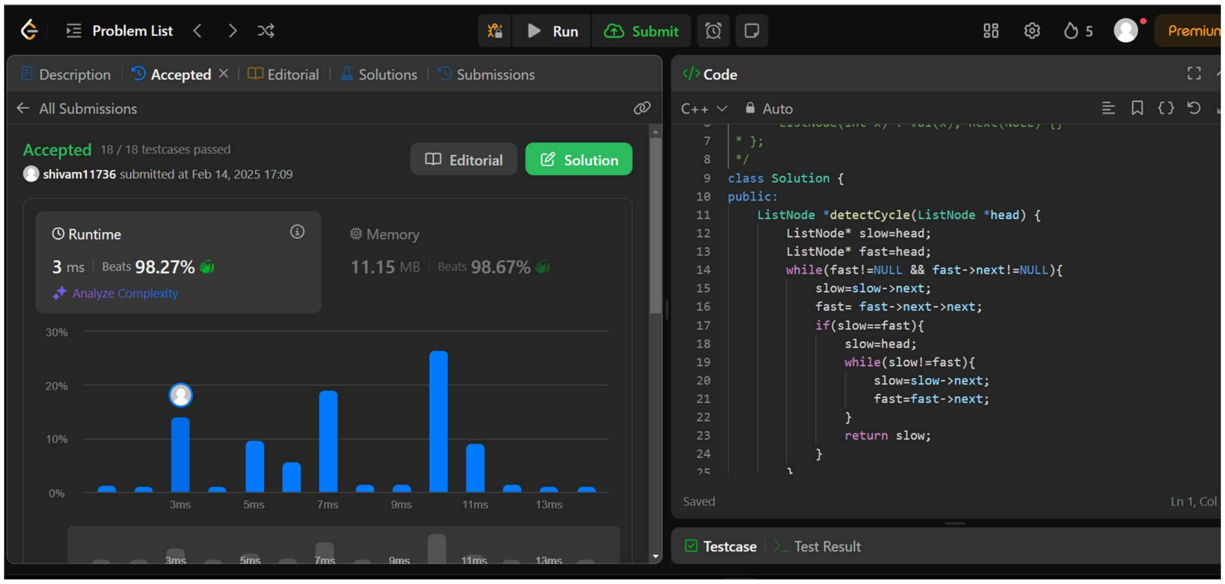
shivam11736 submitted at Feb 14, 2025 17:07

Runtime
13 ms | Beats 74.55%

Memory
58.28 MB | Beats 61.15%

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        vector<int> arr;
        ListNode* temp=head;
        while(temp!= NULL){
            arr.push_back(temp->val);
            temp=temp->next;
        }
        // sorting array
        sort(arr.begin(), arr.end());
        // update
        temp=head;

        for(int i=0; temp!=NULL; i++){
            temp->val=arr[i];
            temp=temp->next;
        }
        return head;
```

Q11} [Detect a cycle in a linked list 2]()



Accepted 18 / 18 testcases passed

shivam11736 submitted at Feb 14, 2025 17:09

Runtime 3 ms | Beats 98.27%

Memory 11.15 MB | Beats 98.67%

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