

Name: Shivam Kumar

Uid: 22BCS11736

Q1} [Print linked list](#)

The screenshot shows a C++ IDE with a solution for the 'Print linked list' problem. The code defines a `Node` struct and a `printList` function. The output window shows 'Problem Solved Successfully' with 1112/1112 test cases passed, 5/6 attempts, 83% accuracy, and a time taken of 0.06 seconds.

```
1 // Driver Code Starts
19 /*
20 struct Node {
21     int data;
22     struct Node* next;
23 }
24 Node(int x) {
25     data = x;
26     next = nullptr;
27 }
28 };
29 */
30 class Solution {
31 public:
32     // Function to display the elements of a linked list in same line
33     void printList(Node* head) {
34         // your code goes here
35         Node* temp=head;
36         while(temp!=NULL){
37             cout<<temp->data<<" ";
38             temp=temp->next;
39         }
40     }
41 };
42 // Driver Code Ends
```

Output Window: Problem Solved Successfully ✓

Test Cases Passed: 1112 / 1112

Attempts : Correct / Total: 5 / 6

Accuracy : 83%

Time Taken: 0.06

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

The screenshot shows a C++ IDE with a solution for the 'Remove duplicates from a sorted list' problem. The code defines a `ListNode` struct and a `deleteDuplicates` function. The submission status is 'Accepted' with 168/168 testcases passed. The runtime is 0 ms (Beats 100.00%) and memory is 16.36 MB (Beats 11.26%).

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode() : val(0), next(nullptr) {}
7  *     ListNode(int x) : val(x), next(nullptr) {}
8  *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* deleteDuplicates(ListNode* head) {
14         ListNode* curr=head;
15         while(curr != NULL && curr->next != NULL){
16             if(curr->next!=NULL && curr->val==curr->next->val){
17                 ListNode* temp=curr->next;
18                 curr->next=temp->next;
19                 temp->next=NULL;
20             }
21             curr=curr->next;
22         }
23         return head;
24     }
25 };
26
```

Submission Status: Accepted 168 / 168 testcases passed

Runtime: 0 ms | Beats 100.00%

Memory: 16.36 MB | Beats 11.26%

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

The screenshot shows the LeetCode interface for the 'Reverse a linked list' problem. The submission is accepted, with 28/28 test cases passed. The runtime is 0 ms, beating 100.00% of other submissions. The memory usage is 13.37 MB, beating 70.61% of other submissions. A bar chart shows the runtime distribution, with the majority of submissions falling between 0 and 1 ms. The code is written in C++ and defines a singly-linked list structure and a solution class.

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode() : val(0), next(nullptr) {}
7  *     ListNode(int x) : val(x), next(nullptr) {}
8  *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13
14     ListNode* reverse(ListNode* prev, ListNode* curr) {
15         // base case
16         if(curr==NULL){
17             return prev;
18         }
19     }
20 }
```

Q4) Delete middle node of a list

The screenshot shows the LeetCode interface for the 'Delete middle node of a list' problem. The submission is accepted, with 70/70 test cases passed. The runtime is 0 ms, beating 100.00% of other submissions. The memory usage is 311.86 MB, beating 98.45% of other submissions. A bar chart shows the runtime distribution, with the majority of submissions falling between 0 and 4 ms. The code is written in C++ and defines a solution class with a deleteMiddle method.

```
11 class Solution {
12 public:
13     ListNode* deleteMiddle(ListNode* head) {
14
15         // base case
16         if(head == NULL || head->next == NULL){
17             return NULL;
18         }
19         ListNode* slow=head;
20         ListNode* fast=head;
21         ListNode* prev=NULL;
22         while(fast!=NULL && fast->next !=NULL){
23             prev=slow;
24             slow=slow->next;
25             fast=fast->next->next;
26         }
27         prev->next=slow->next;
28         delete slow;
29         return head;
30 }
```

Q5) [Merge two sorted linked lists](#)

The screenshot displays a code editor interface for a problem titled "Merge two sorted linked lists". The left sidebar shows the submission status: "Accepted", "208 / 208 testcases passed", and "shivam11736 submitted at Feb 14, 2025 17:00". Below this, the runtime and memory usage are shown: "Runtime: 0 ms | Beats 100.00%", "Memory: 19.50 MB | Beats 62.56%". A bar chart indicates the runtime performance, with a single bar at 0 ms. The main editor shows the C++ code for merging two sorted linked lists. The code uses a dummy node and a while loop to merge the two lists.

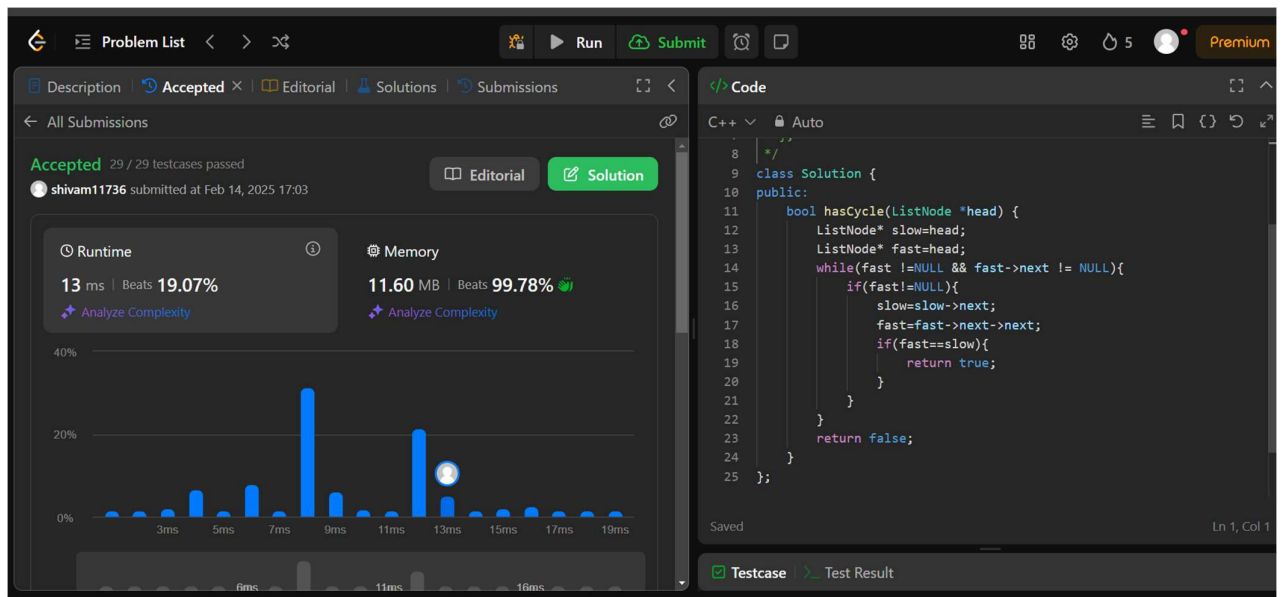
```
14  ListNode* temp=new ListNode(0);
15  ListNode* curr=temp;
16  while(list1 != NULL && list2 != NULL ){
17      if(list1->val > list2->val){
18          curr->next=list2;
19          list2=list2->next;
20      }
21      else{
22          curr->next=list1;
23          list1=list1->next;
24      }
25      curr=curr->next;
26  }
27  if(list1!=NULL){
28      curr->next=list1;
29  }
30  else{
31      curr->next=list2;
32  }
33  return temp->next;
```

Q6) [Remove duplicates from sorted lists 2](#)

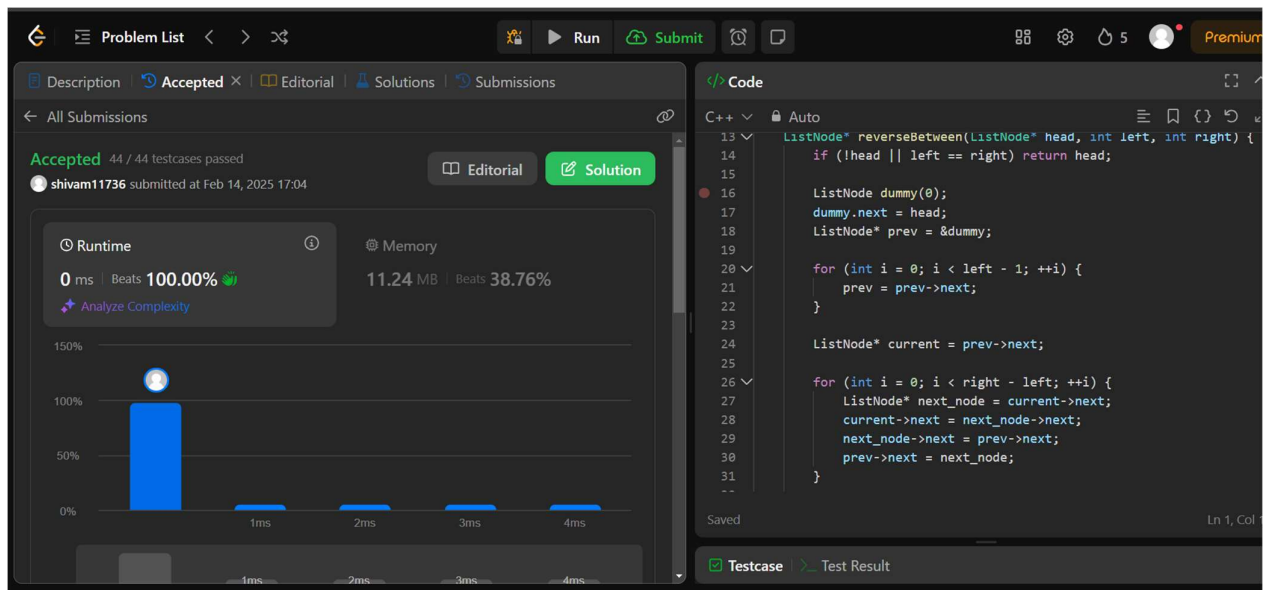
The screenshot displays a code editor interface for a problem titled "Remove duplicates from sorted lists 2". The left sidebar shows the submission status: "Accepted", "166 / 166 testcases passed", and "shivam11736 submitted at Feb 14, 2025 17:01". Below this, the runtime and memory usage are shown: "Runtime: 0 ms | Beats 100.00%", "Memory: 15.70 MB | Beats 41.59%". A bar chart indicates the runtime performance, with a single bar at 0 ms. The main editor shows the C++ code for removing duplicates from a sorted linked list. The code uses a dummy node and a while loop to traverse the list and remove duplicates.

```
11  class Solution {
12  public:
13      ListNode* deleteDuplicates(ListNode* head) {
14          if (head == nullptr || head->next == nullptr) {
15              return head;
16          }
17
18          ListNode* dummy = new ListNode(0, head);
19          ListNode* prev = dummy;
20          ListNode* curr = head;
21          ListNode* temp = head->next;
22          bool flag = false;
23
24          while (temp != nullptr) {
25              if (curr->val != temp->val) {
26                  if (flag) {
27                      prev->next = temp;
28                      flag = false;
29                  } else {
```

Q7) [Detect a cycle in a linked list](#)



Q8) [Reverse linked list 2](#)



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%

Code

```
10  */
11  class Solution {
12      int len(ListNode* head){
13          int len=0;
14          ListNode* temp=head;
15          while(temp){
16              temp=temp->next;
17              len++;
18          }
19          return len;
20      }
21  public:
22      ListNode* rotateRight(ListNode* head, int k) {
23          if(!head || !head->next) return head;
24          //move to len-kth node
25          int length=len(head);
26          k=k%length;
27          //edge case
28          if(k==0) return head; //no reverse
```

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%

Code

```
11  class Solution {
12  public:
13      ListNode* sortList(ListNode* head) {
14          vector<int> arr;
15          ListNode* temp=head;
16          while(temp!= NULL){
17              arr.push_back(temp->val);
18              temp=temp->next;
19          }
20          // sorting array
21          sort(arr.begin(), arr.end());
22          // update
23          temp=head;
24
25          for(int i=0; temp!=NULL; i++){
26              temp->val=arr[i];
27              temp=temp->next;
28          }
29          return head;
30      }
```

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

Problem List < > < > Run Submit < > 5 Premium

Description Accepted x Editorial Solutions Submissions

All Submissions


Accepted 18 / 18 testcases passed

shivam11736 submitted at Feb 14, 2025 17:09

Editorial Solution

Runtime 3 ms Beats 98.27% Analyze Complexity

Memory 11.15 MB Beats 98.67%



Runtime (ms)	Percentage
3ms	15%
5ms	10%
7ms	20%
9ms	10%
11ms	25%
13ms	10%

```
C++ Auto
7 * };
8 */
9 class Solution {
10 public:
11     ListNode* detectCycle(ListNode *head) {
12         ListNode* slow=head;
13         ListNode* fast=head;
14         while(fast!=NULL && fast->next!=NULL){
15             slow=slow->next;
16             fast= fast->next->next;
17             if(slow==fast){
18                 slow=head;
19                 while(slow!=fast){
20                     slow=slow->next;
21                     fast=fast->next;
22                 }
23                 return slow;
24             }
25         }
26     }
27 }
```

Saved Ln 1, Col 1

Testcase Test Result