

## Assignment 3

Name- Rahul Gupta

UID- 22BCS10469

Class- 609-B

### Question 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){  
            cout<<temp->data<<" ";  
            temp=temp->next; } } };
```

The screenshot displays a coding platform interface. At the top, there's a navigation bar with links for Courses, Tutorials, Jobs, Practice, and Contests. The main header features the platform's logo and a user profile icon. Below the header, the 'Submissions' tab is active, showing a 'Problem Solved Successfully' message with a green checkmark. The submission details include: Test Cases Passed (1112 / 1112), Attempts: Correct / Total (1 / 3), Accuracy (33%), Points Scored (0 / 1), and Time Taken (0.05). The 'Compilation Results' section shows the C++ code for the 'Print Linked List' problem. The code defines a 'Node' struct with 'data' and 'next' fields, and a 'Solution' class with a 'printList' method that iterates through the linked list and prints the data of each node on the same line. The code is written in C++ (g++ 5.4) and includes a 'Start Timer' button.

90% Refund

Courses ▾ Tutorials ▾ Jobs ▾ Practice ▾ Contests ▾

⌵ Problem Editorial Submissions Comments

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  
**1 / 3**

Accuracy: 33%

Points Scored ⓘ  
**0 / 1**

Time Taken  
**0.05**

Your Total Score: 6

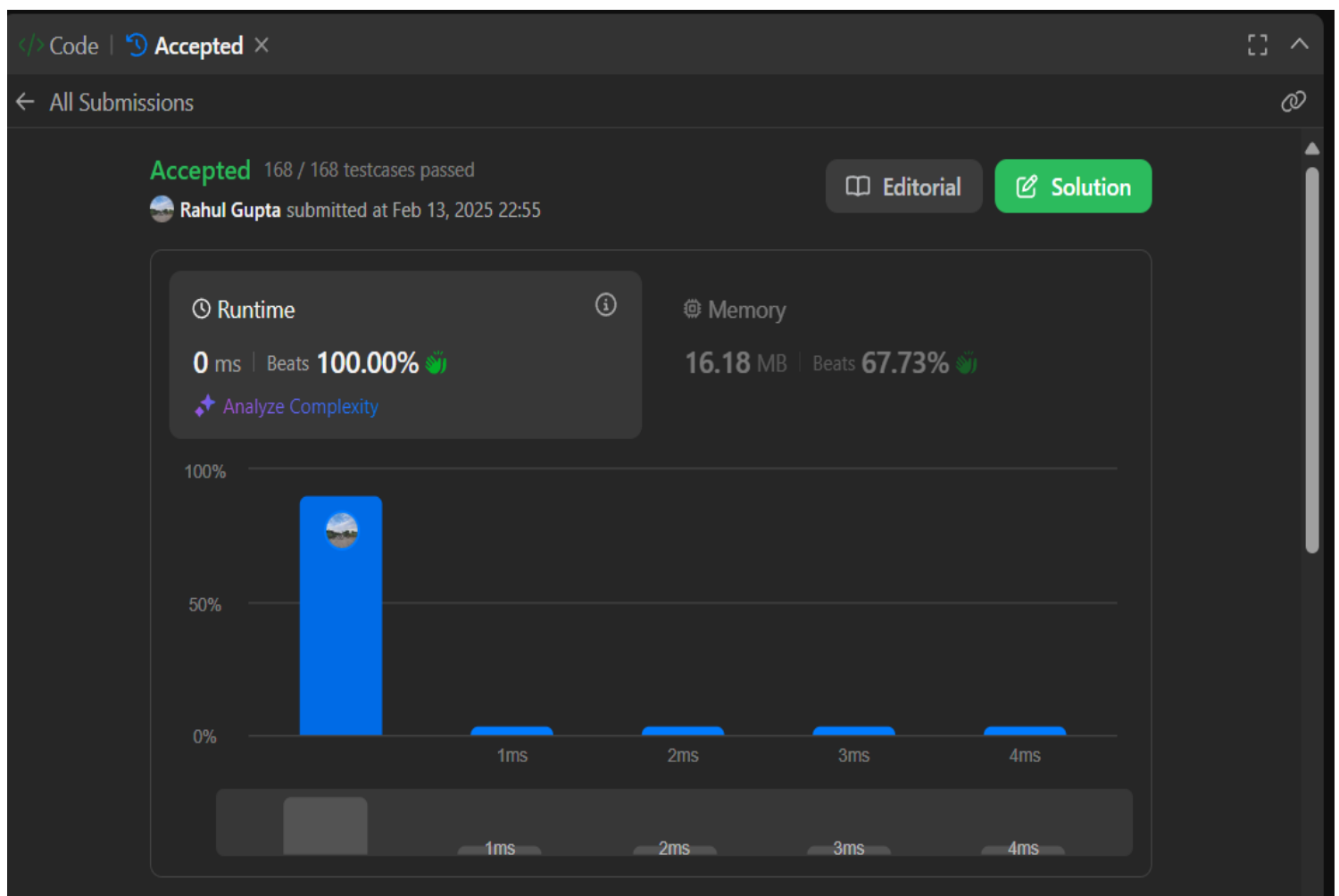
C++ (g++ 5.4) Start Timer

```
1 // } Driver Code Ends  
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 /*  
31     Print elements of a linked list on console  
32     Head pointer input could be NULL as well for empty list  
33 /*  
34  
35 class Solution {  
36 public:  
37     // Function to display the elements of a linked list in same line  
38     void printList(Node *head) {  
39         Node *temp=head;  
40         while(temp){  
41             cout<<temp->data<<" ";  
42             temp=temp->next;  
43         }  
44     }  
45 };  
46  
47 // } Driver Code Ends
```

## Question 2) [Remove Duplicates from Sorted List](#)

Code)

```
class Solution {  
public:  
  
    ListNode* deleteDuplicates(ListNode* head) {  
  
        ListNode* current = head;  
  
        while (current && current->next) {  
  
            if (current->val == current->next->val) {  
  
                current->next = current->next->next; // Skip duplicate node  
  
            } else {  
  
                current = current->next; // Move to the next node  
  
            }  
  
        }  
  
        return head; } };
```



### Question 3) [Reverse Linked List](#)

#### Code)

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

The screenshot displays a coding platform interface with a dark theme. The top navigation bar includes a 'Problem List' tab, a 'Run' button, a 'Submit' button, and a 'Premium' badge. The main content area is divided into two panels. The left panel, titled 'All Submissions', shows the submission status 'Accepted' with '28 / 28 testcases passed'. It also displays the user 'Rahul Gupta' and the submission time 'Mar 05, 2025 14:41'. Below this, there are performance metrics: 'Runtime: 0 ms | Beats 100.00%' and 'Memory: 13.47 MB | Beats 39.77%'. A bar chart shows the user's performance relative to others. The right panel, titled 'Code', shows the C++ code for the 'reverseList' function. The code is as follows:

```
1 class Solution {  
2 public:  
3     ListNode* reverseList(ListNode* head) {  
4         ListNode* prev = nullptr;  
5         ListNode* curr = head;  
6  
7         while (curr) {  
8             ListNode* nextNode = curr->next;  
9             curr->next = prev;  
10            prev = curr;  
11            curr = nextNode;  
12        }  
13  
14        return prev;  
15    }  
16 };  
17
```

The bottom of the interface shows a 'Testcase' tab with 'Case 1', 'Case 2', and 'Case 3' selected.

## Question 4) [Delete the Middle Node of a Linked List](#) Code)

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

The screenshot displays a coding platform interface with the following components:

- Problem List:** A navigation bar at the top left with icons for problem list, accepted solutions, editorials, and submissions.
- Accepted Solutions:** A section showing that 70 out of 70 testcases passed. The user 'Rahul Gupta' is listed as the submitter, with a submission time of Mar 05, 2025 14:48.
- Runtime and Memory:** The solution has a runtime of 2 ms (Beats 49.98%) and a memory usage of 312.08 MB (Beats 55.20%).
- Bar Chart:** A bar chart showing the distribution of runtime times for other solutions. The x-axis represents runtime in milliseconds (1ms to 7ms), and the y-axis represents the percentage of solutions. The highest bar is at 1ms, followed by 4ms.
- Code Editor:** A C++ code editor showing the implementation of the 'deleteMiddle' function. The code is as follows:

```
1 class Solution {  
2 public:  
3     ListNode* deleteMiddle(ListNode* head) {  
4         if (!head || !head->next) {  
5             return nullptr;  
6         }  
7         ListNode* slow = head;  
8         ListNode* fast = head;  
9         ListNode* prev = nullptr;  
10        while (fast && fast->next) {  
11            prev = slow;  
12            slow = slow->next;  
13            fast = fast->next->next; }  
14        prev->next = slow->next;  
15        return head;  
16    }  
17 }  
18 };
```
- Testcase Results:** A section at the bottom showing the results for three test cases. Case 1 is selected, and the result is 'Test Result'.

## Question 5) [Merge Two Sorted Lists](#)

### Code)

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

The screenshot displays a code editor interface with a dark theme. The top navigation bar includes icons for problem list, run, submit, and other utilities. The main editor area shows the C++ code for the 'Merge Two Sorted Lists' problem. The code is as follows:

```
1 class Solution {  
2 public:  
3     ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {  
4         ListNode* dummy = new ListNode();  
5         ListNode* current = dummy;  
6         while (list1 && list2) {  
7             if (list1->val <= list2->val) {  
8                 current->next = list1;  
9                 list1 = list1->next;  
10            } else {  
11                current->next = list2;  
12                list2 = list2->next;  
13            }  
14            current = current->next;  
15        }  
16        if (list1) {  
17            current->next = list1;  
18        } else if (list2) {  
19            current->next = list2;  
20        }  
21        return dummy->next;  
22    }  
23 };
```

On the left side of the editor, there is a sidebar showing the problem status. It indicates that the solution is 'Accepted' with 208/208 testcases passed. The runtime is 3 ms, which is 4.22% better than the average. The memory usage is 19.40 MB, which is 86.51% better than the average. A bar chart shows the performance of the solution compared to other submissions.

At the bottom of the editor, there is a 'Testcase' tab with a 'Test Result' section. It shows the results for 'Case 1', 'Case 2', and 'Case 3'.

## Question 6) [Linked List Cycle](#)

```
class Solution {
```

```
public:
```

```
    bool hasCycle(ListNode* head) {
```

```
        if (head == NULL || head->next == NULL) {
```

```
            return false; }
```

```
        ListNode* slow = head;
```

```
        ListNode* fast = head->next;
```

```
        while (fast != slow) {
```

```
            if (fast->next == NULL || fast->next->next == NULL) {
```

```
                return false; }
```

```
            slow = slow->next;
```

```
            fast = fast->next->next; }
```

```
        return true; } };
```

The screenshot displays a coding platform interface with the following components:

- Top Bar:** Includes navigation icons, a 'Problem List' link, and a 'Premium' badge.
- Left Panel:**
  - Accepted:** 29 / 29 testcases passed.
  - Runtime:** 8 ms, Beats 80.86%.
  - Memory:** 11.98 MB, Beats 24.25%.
  - Bar Chart:** A performance comparison chart showing the user's performance (blue bar) relative to other submissions (grey bars) for various runtime and memory thresholds.
- Code Editor:** Displays the C++ code for the 'hasCycle' function, which uses Floyd's Cycle-Finding Algorithm (slow and fast pointers).
- Testcase Panel:** Shows a 'Testcase' tab and a 'Test Result' section with tabs for 'Case 1', 'Case 2', and 'Case 3'.

## Question 7) [Rotate List](#)

### Code)

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

The screenshot displays a coding platform interface for the 'Rotate List' problem. The left sidebar shows submission statistics: 'Accepted 232 / 232 testcases passed', 'Runtime 0 ms | Beats 100.00%', and 'Memory 16.23 MB | Beats 93.87%'. A bar chart visualizes the user's performance relative to others. The main editor area shows the C++ code for the 'rotateRight' function, which rotates a linked list to the right by k places. The bottom panel shows the 'Testcase' section with 'Case 1' selected.

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

## Question 8) [Sort List](#)

### Code)

```
#include <iostream>

using namespace std;

class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) return head;
        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;
        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; } };
```

Problem List

Run
Submit

42
Premium

Description
Editorial
Solutions
Accepted
Submissions

All Submissions

Accepted 30 / 30 testcases passed  
Rahul Gupta submitted at Mar 05, 2025 14:59

Editorial
Solution

Runtime
12 ms | Beats 78.58%

Memory
56.90 MB | Beats 93.44%

Code
C++

```

1 #include <iostream>
2 using namespace std;
3 class Solution {
4 public:
5     ListNode* sortList(ListNode* head) {
6         if (!head || !head->next) return head;
7         ListNode* slow = head;
8         ListNode* fast = head->next;
9         while (fast && fast->next) {
10             slow = slow->next;
11             fast = fast->next->next;
12         }
13         ListNode* mid = slow->next;
14         slow->next = nullptr;
15         ListNode* left = sortList(head);
16         ListNode* right = sortList(mid);
17         return merge(left, right);
18     }
19     ListNode* merge(ListNode* l1, ListNode* l2) {
20         ListNode dummy(0);
21         ListNode* tail = &dummy;
22         while (l1 && l2) {
23             if (l1->val < l2->val) {
24                 tail->next = l1;
25                 l1 = l1->next;
26             } else {
27                 tail->next = l2;
28                 l2 = l2->next;
29             }
30             tail = tail->next;
31         }
32         if (l1) tail->next = l1;
33         if (l2) tail->next = l2;
34         return dummy.next;
35     }
36 };

```

Testcase
Test Result

Case 1 Case 2 Case 3 +

Source

## Question 9) [Merge k Sorted Lists](#)

### Code)

```
#include <vector>

using namespace std;

class Solution {
public:

    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        if (!l1) return l2;
        if (!l2) return l1;

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

    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) return nullptr;
        return divideAndConquer(lists, 0, lists.size() - 1);
    }

    ListNode* divideAndConquer(vector<ListNode*>& lists, int left, int right) {
        if (left == right) return lists[left];
```

```
int mid = left + (right - left) / 2;
```

```
ListNode* l1 = divideAndConquer(lists, left, mid);
```

```
ListNode* l2 = divideAndConquer(lists, mid + 1, right);
```

```
return mergeTwoLists(l1, l2);
```

```
}
```

```
};
```

The screenshot displays a C++ code editor with a dark theme. The top navigation bar shows 'Problem List', 'Run', 'Submit', and 'Premium' status. The left sidebar contains tabs for 'Description', 'Editorial', 'Solutions', 'Submissions', and 'Accepted'. The main area is divided into three sections: a submission summary, a runtime/memory analysis panel, and a code editor.

**Submission Summary:** Shows 'Accepted' status with 134/134 testcases passed. The user 'Rahul Gupta' submitted the solution on Mar 05, 2025 at 15:01. Buttons for 'Editorial' and 'Solution' are visible.

**Runtime/Memory Analysis:** The 'Runtime' section shows 0 ms execution time, beating 100.00% of other solutions. The 'Memory' section shows 18.57 MB usage, beating 50.76% of other solutions. A bar chart below these metrics shows the distribution of execution times across various test cases.

**Code Editor:** Contains the following C++ code:

```
1 #include <vector>
2 using namespace std;
3 class Solution {
4 public:
5     ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
6         if (!l1) return l2;
7         if (!l2) return l1;
8
9         if (l1->val < l2->val) {
10             l1->next = mergeTwoLists(l1->next, l2);
11             return l1;
12         } else {
13             l2->next = mergeTwoLists(l1, l2->next);
14             return l2;
15         }
16     }
17
18     ListNode* mergeKLists(vector<ListNode*>& lists) {
19         if (lists.empty()) return nullptr;
20         return divideAndConquer(lists, 0, lists.size() - 1);
21     }
22
23     ListNode* divideAndConquer(vector<ListNode*>& lists, int left, int right) {
24         if (left == right) return lists[left];
```

The bottom of the editor shows a 'Testcase' tab with 'Case 1', 'Case 2', and 'Case 3' selected. The source code is saved, and the cursor is at line 23, column 69.