Assignment-03

Student Name: Deepraj Patel **UID: 22BCS14957**
Branch: CSE **Section/Group: 610-B**
Semester: 6th **Date of Performance: 05-03-2025**
Subject Name: AP **Subject Code: 22CSP-351**

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

Problem Solved Successfully

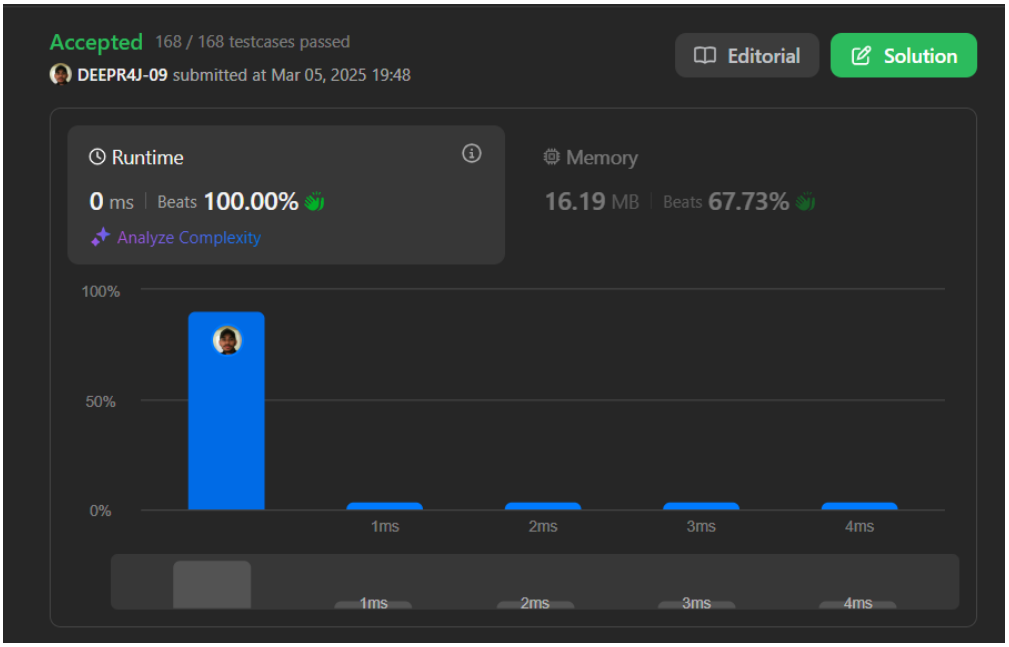
Suggest Feedback

Test Cases Passed	Attempts : Correct / Total
1112 / 1112	**1 / 1**
	Accuracy : 33%

Points Scored	Time Taken
1 / 1	**0.07**
Your Total Score: 7 ↑	

```
class Solution {
  public:
    // Function to display the elements of a linked list in same line
    void printList(Node *head) {
        while (head) {
        cout << head->data << " ";
        head = head->next;
    }

    }
};
```

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

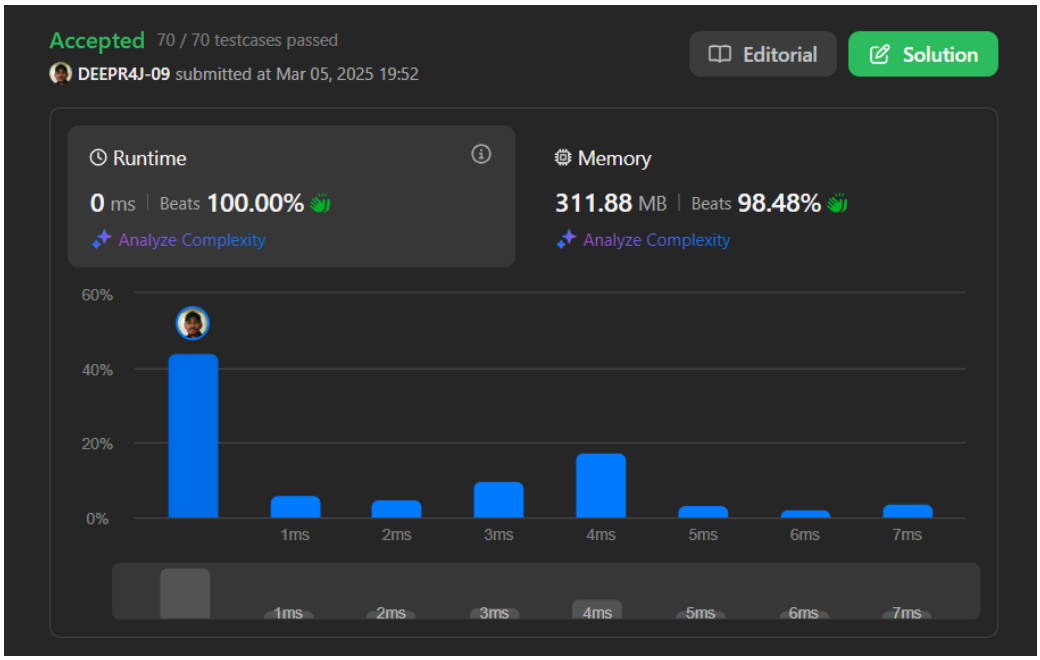



```cpp
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (!head) {
            return nullptr;
        }

        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;
    }
};
```

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

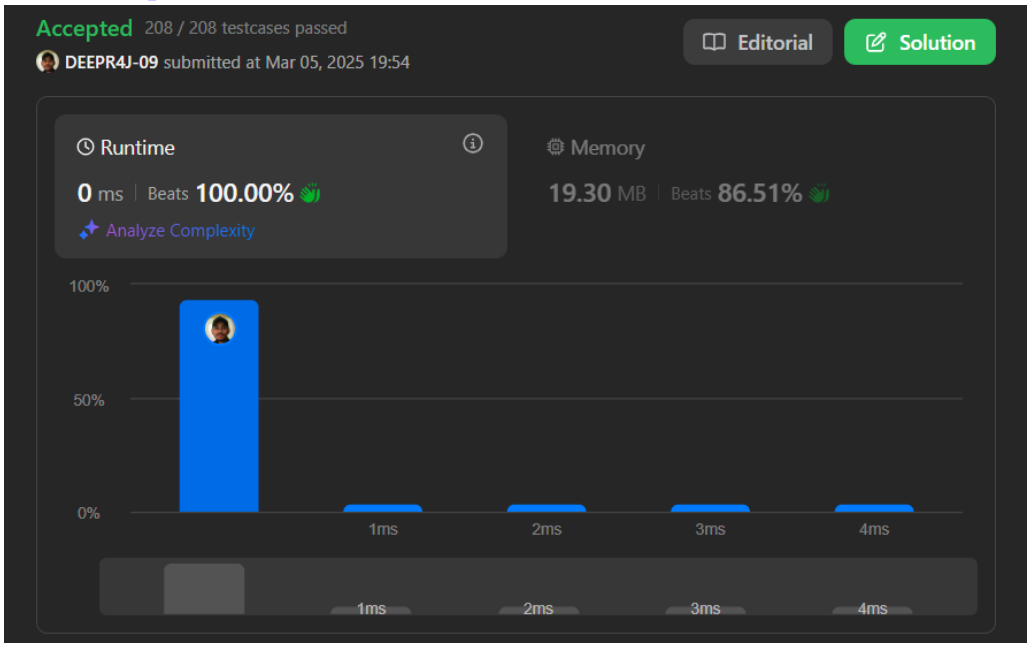

```cpp
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* next = nullptr;
        ListNode* curr = head;

        while (curr != nullptr) {

            next = curr->next;

            curr->next = prev;

            prev = curr;
            curr = next;
        }

        return prev;
    }
};
```

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



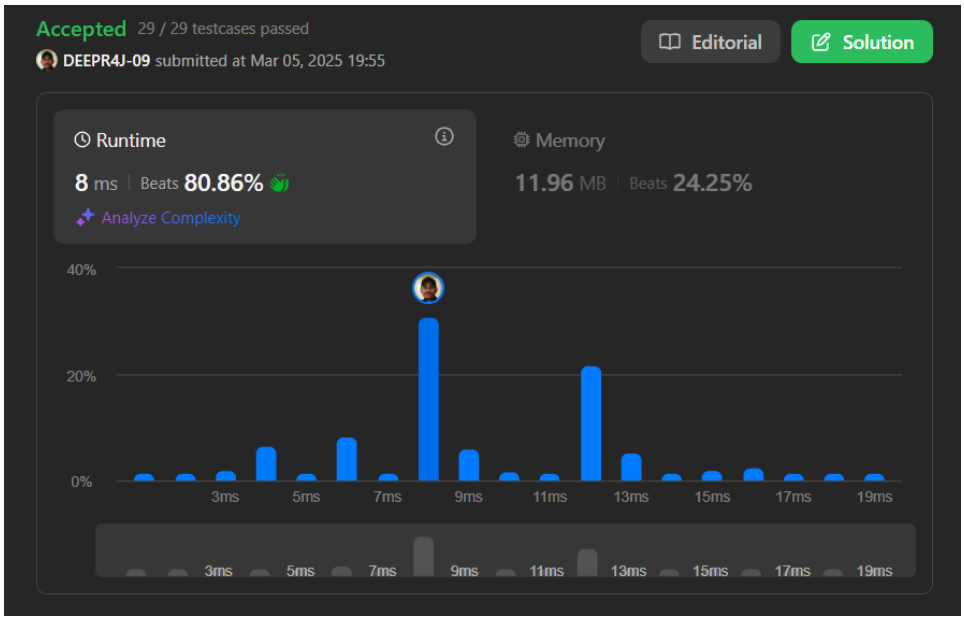

```cpp
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if(!head->next) return NULL;
        if(!head->next->next){
            head->next = NULL;
            return head;
        }
        ListNode* slow = head;
        ListNode* fast = head;
        while(fast && fast->next){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

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



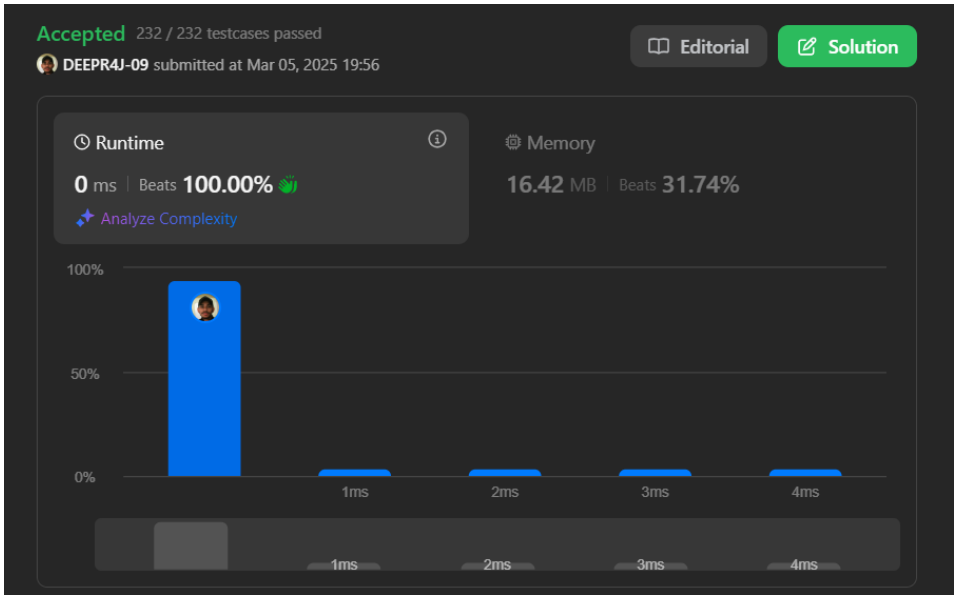

```cpp

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;
        }
    }
};
```

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



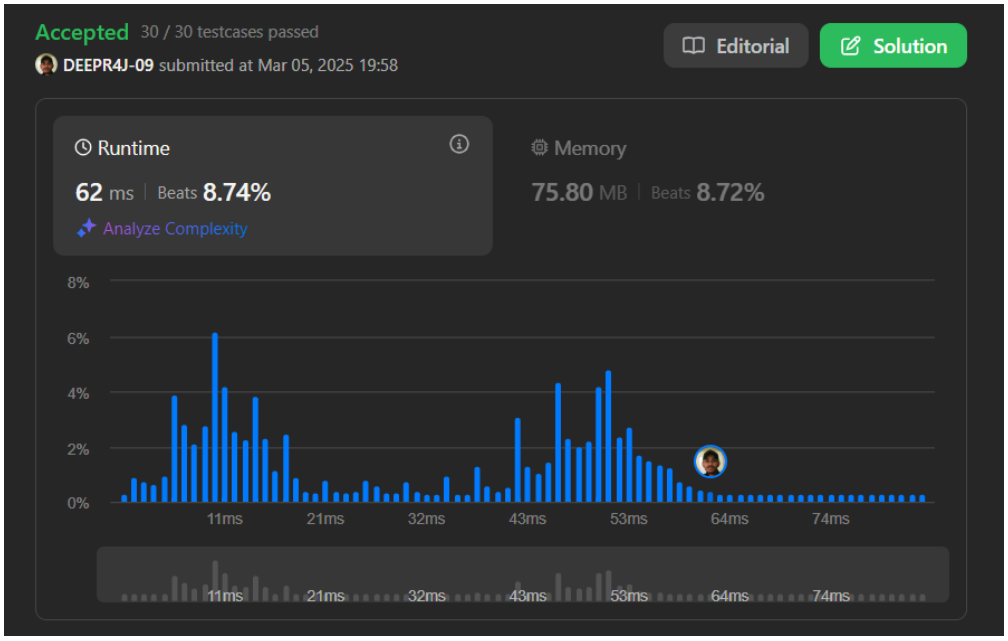

```cpp
class Solution {
public:
    bool hasCycle(ListNode* head) {
        if (head == NULL || head->next == NULL) {
            return false;
        }
        ListNode* slow = head;
        ListNode* fast = head->next;
        while (fast != slow) {
            if (fast->next == NULL || fast->next->next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

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




```cpp
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        // base condition
        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;  // curr points to tail node sorta

        return head;

    }
};
```

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

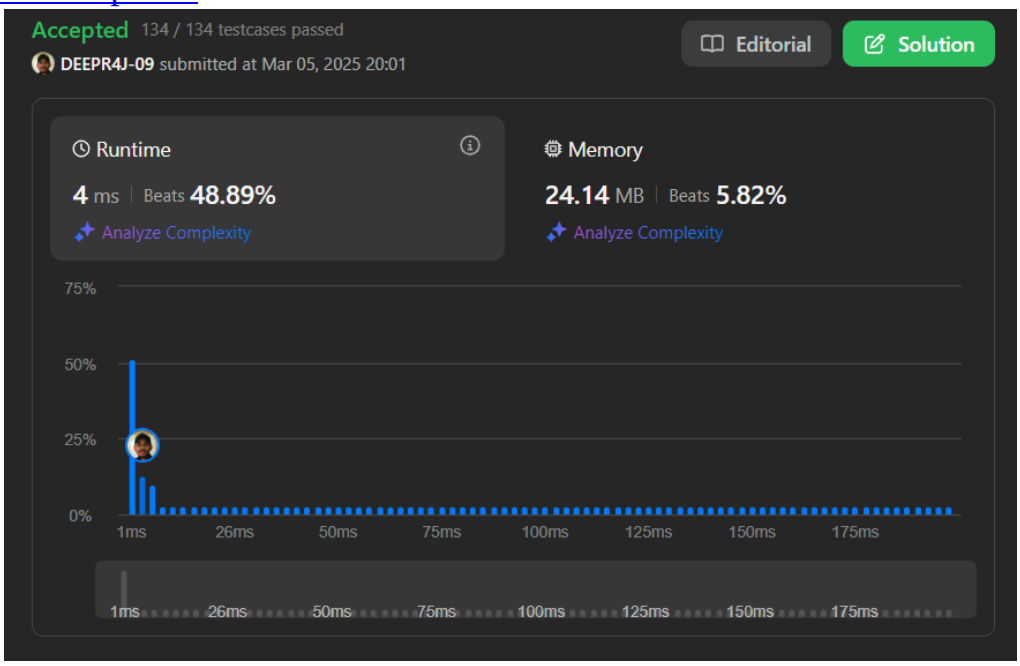



```cpp
class Solution {
public:
    ListNode* getmid(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head->next;

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

    ListNode* merge(ListNode* left, ListNode* right) {
        if (left == NULL)
            return right;
        if (right == NULL)
            return left;

        ListNode* dummy = new ListNode(0);
        ListNode* temp = dummy;

        while (left != NULL && right != NULL) {
            if (left->val < right->val) {
                temp->next = left;
                temp = left;
                left = left->next;
            } else {
                temp->next = right;
```

```
29                  temp->next = right;
30                  temp = right;
31                  right = right->next;
32              }
33          }
34          while (left != NULL) {
35              temp->next = left;
36              temp = left;
37              left = left->next;
38          }
39          while (right != NULL) {
40              temp->next = right;
41              temp = right;
42              right = right->next;
43          }
44          dummy = dummy->next;
45          return dummy;
46      }
47
48      ListNode* sortList(ListNode* head) {
49          // using merge sort
50
51          // base case
52          if (head == NULL || head->next == NULL)
53              return head;
54
55          ListNode* mid = getmid(head);
56
57          ListNode* left = head;
```

```
57          ListNode* left = head;
58          ListNode* right = mid->next;
59          mid->next = NULL;
60
61          left = sortList(left);
62          right = sortList(right);
63
64          ListNode* result = merge(left, right);
65
66          return result;
67      }
68  };
```

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




```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);
    }

    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:
```

```
29                  curr->next = l1;
30                  l1 = l1->next;
31              } else {
32                  curr->next = l2;
33                  l2 = l2->next;
34              }
35              curr = curr->next;
36          }
37
38          curr->next = l1 ? l1 : l2;
39
40          return dummy->next;
41      }
42  };
```