



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

AP Assignment

Student Name: Shreya Singh

Branch: BE-CSE

Semester: 6th

Subject Name: AP Lab-2

UID: 22BCS12423

Section/Group: 606-B

Date : 06/03/25

Subject Code: 22CSP-351

Problem 1

1. Aim: Given a linked list. Print all the elements of the linked list separated by space followed.

2. Implementation/Code:

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

3. Output:

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed

1112 / 1112

Attempts : Correct / Total

2 / 2

Accuracy : 100%

Time Taken

0.08

Suggest Feedback

```
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         // your code goes here
40         Node*current=head;
41         while(current!=nullptr){
42             cout<<current->data<<" ";
43             current = current->next;
44         }
45     }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Problem 2

1. Aim: Remove duplicates from a sorted list

2. Implementation/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;
            } else {
                current = current->next;
            }
        }
        return head;
    }
};
```

3. Output:

The screenshot displays a coding platform interface. The top navigation bar includes links for Description, Editorial, Solutions, Accepted (168 / 168 testcases passed), and Submissions. The user 'shreya_singh_' is noted as having submitted the solution on Mar 06, 2025 at 20:31. The solution is marked as 'Accepted'.

Runtime Performance:

- Runtime: 0 ms (Beats 100.00%)
- Memory: 16.18 MB (Beats 67.73%)

Code (C++):

```
9  * };
10 /*
11 class Solution {
12 public:
13     ListNode* deleteDuplicates(ListNode* head) {
14         ListNode* current = head;
15         while (current && current->next) {
16             if (current->val == current->next->val) {
17                 current->next = current->next->next;
18             } else {
19                 current = current->next;
20             }
21         }
22     }
23 }
```

Test Result:

- Accepted (Runtime: 0 ms)
- Case 1, Case 2

Input:

```
head =
[1,1,2]
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Problem 3

1. **Aim:** Reverse a linked list.

2. Implementation/Code:

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode *nextNode, *prevNode = NULL;
        while (head) {
            nextNode = head->next;
            head->next = prevNode;
            prevNode = head;
            head = nextNode;
        }
        return prevNode;
    }
};
```

3. Output:

Accepted 28 / 28 testcases passed
shreya_singh_ submitted at Mar 06, 2025 20:34

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

13.26 MB | Beats 90.63%



Code | C++

```
/**
 * Definition for singly-linked list.
 */
```

```
12 public:
13     ListNode* reverseList(ListNode* head) {
14         ListNode *nextNode, *prevNode = NULL;
15         while (head) {
16             nextNode = head->next;
17             head->next = prevNode;
18             prevNode = head;
19             head = nextNode;
20         }
21         return prevNode;
22     }
23 };
```

Saved

Ln:

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =
[1,2,3,4,5]

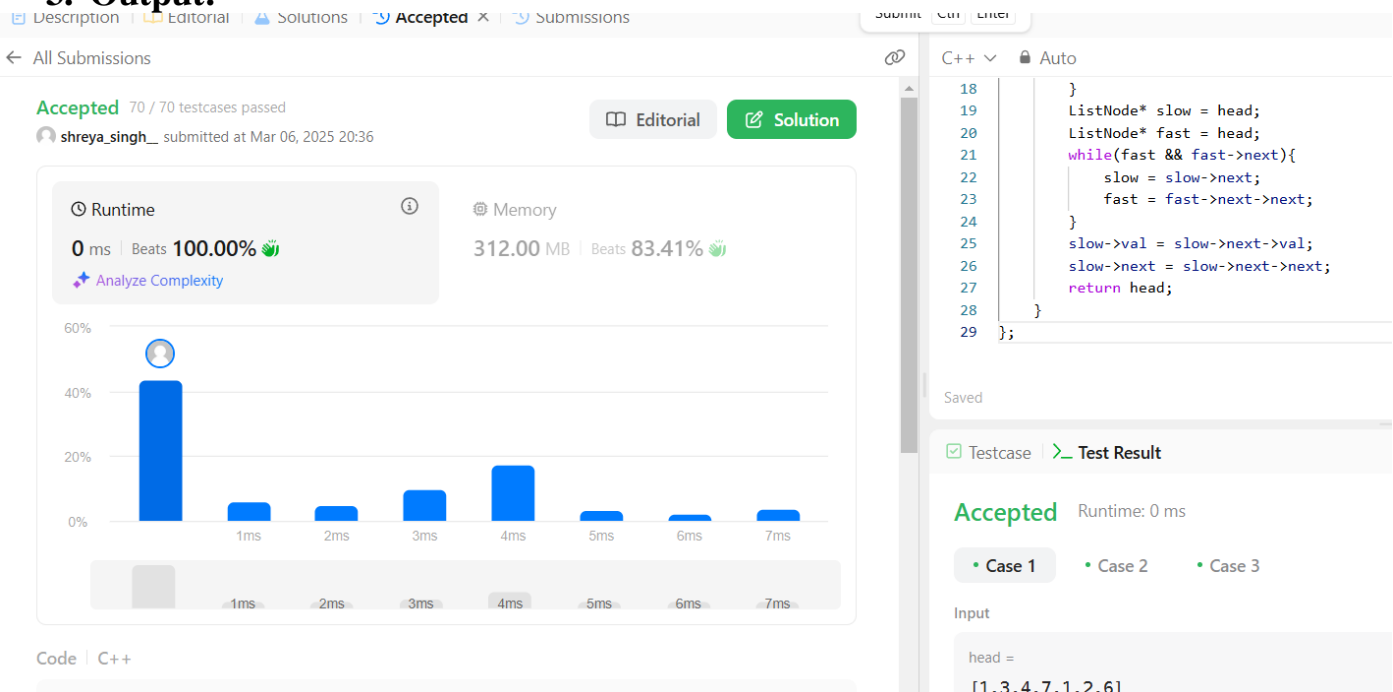
Problem 4

1. **Aim:** Delete middle node of a list

2. Implementation/Code:

```
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;
    }
};
```

3. Output:



Problem 5

1. Aim: Merge two sorted linked lists

2. Implementation/Code:

```
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;
        }
    }
};
```

3. Output:

← All Submissions

Accepted 208 / 208 testcases passed

shreya_singh_ submitted at Mar 06, 2025 20:39

Editorial

Solution

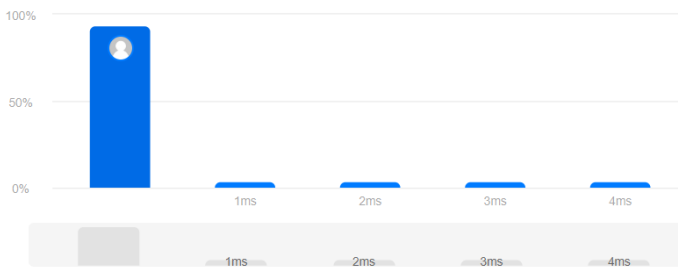
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

19.50 MB | Beats 28.39%



Code | C++

```
/**
 * Definition for singly-linked list.
 */
```

⌕

C++ Auto

```
17 }
18
19 if(list1->val <= list2->val){
20     list1->next = mergeTwoLists(list1->next, list2);
21     return list1;
22 }
23 else{
24     list2->next = mergeTwoLists(list1, list2->next);
25     return list2;
26 }
27 }
28 ;;
```

Saved

Testcase Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

```
list1 =
[1,2,4]
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

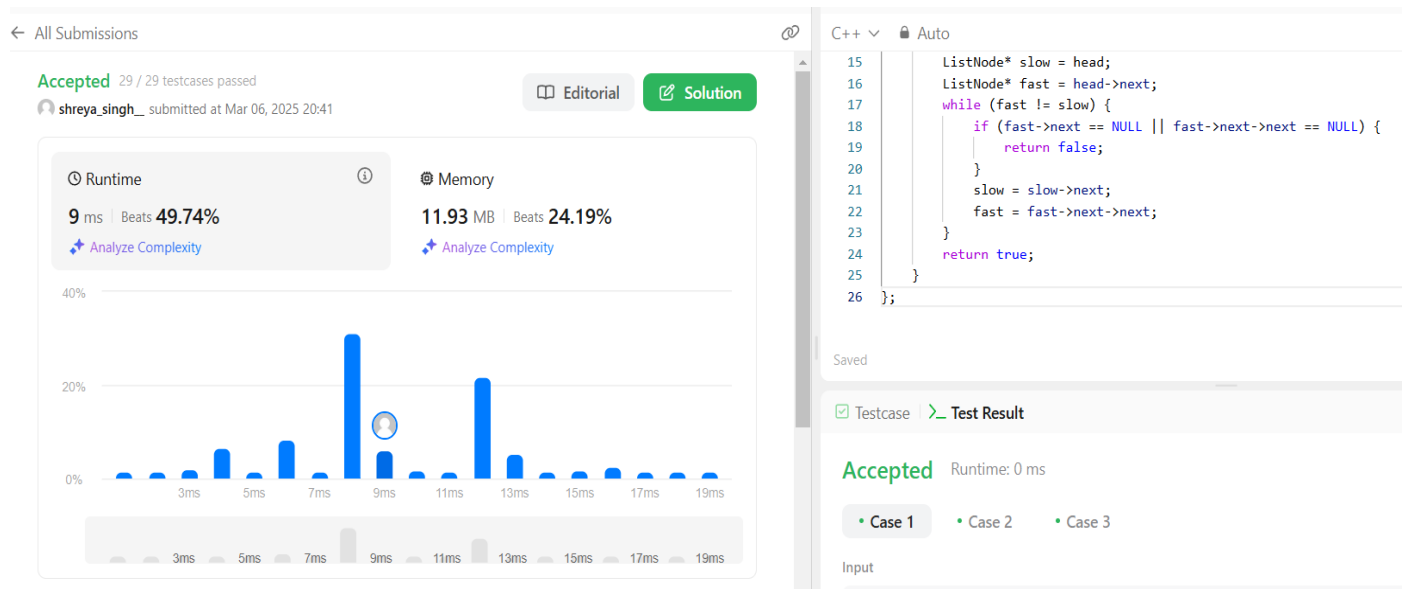
Problem 6

1. **Aim:** Detect a cycle in a linked list.

2. Implementation/Code:

```
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;
    }
};
```

3. Output:



Problem 7

1. Aim: Rotate a list.

2. Implementation/Code:

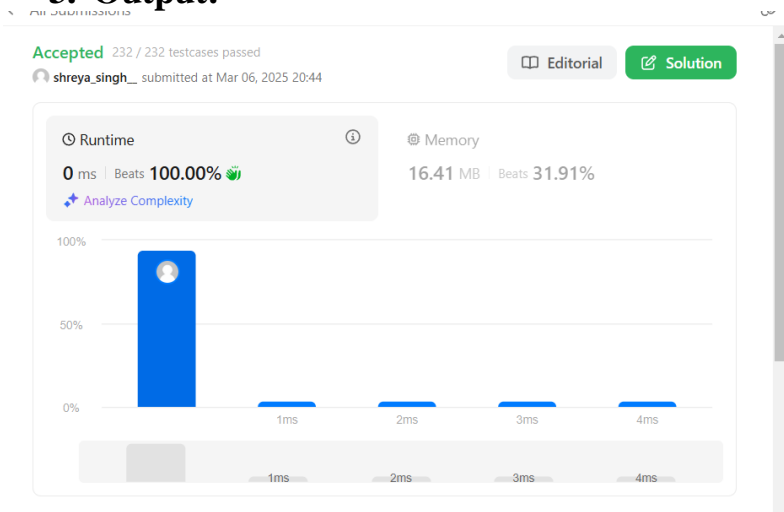
```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0) return head;

        ListNode* tail = head;
        int length = 1;

        while (tail->next) {
            tail = tail->next;
            length++;
        }
        k %= length;
        if (k == 0) return head;
        tail->next = head;
        int stepsToNewHead = length - k;
        ListNode* newTail = tail;

        while (stepsToNewHead--) {
            newTail = newTail->next;
        }
        ListNode* newHead = newTail->next;
        newTail->next = nullptr;
        return newHead;
    }
};
```

3. Output:





Problem 8

1. Aim: Sort List.

2. Implementation/Code:

```
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;
                temp = right;
                right = right->next;
            }
        }
        while (left != NULL) {
            temp->next = left;
            temp = left;
            left = left->next;
        }
        while (right != NULL) {
            temp->next = right;
            temp = right;
            right = right->next;
        }
        dummy = dummy->next;
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    return dummy;  
}
```

```
ListNode* sortList(ListNode* head) {  
    // using merge sort  
  
    // base case  
    if (head == NULL || head->next == NULL)  
        return head;  
  
    ListNode* mid = getmid(head);  
  
    ListNode* left = head;  
    ListNode* right = mid->next;  
    mid->next = NULL;  
  
    left = sortList(left);  
    right = sortList(right);  
  
    ListNode* result = merge(left, right);  
  
    return result;  
}  
};
```

3. Output:

← All Submissions

Accepted 174 / 174 testcases passed

shreya_singh_ submitted at Feb 20, 2025 18:24

Editorial

Solution

Runtime

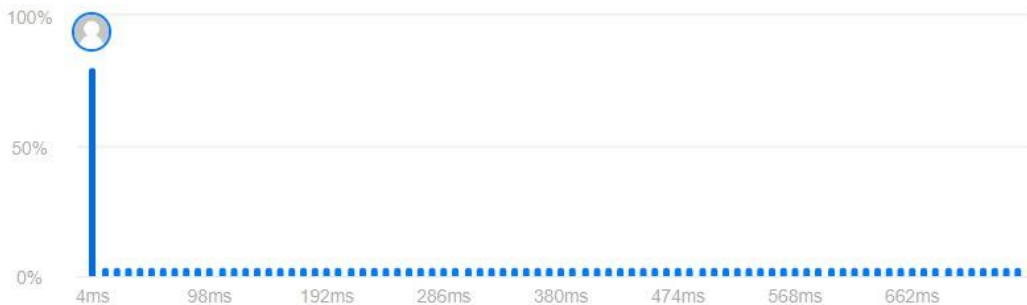


2 ms | Beats 30.81%

Analyze Complexity

Memory

52.20 MB | Beats 61.52%





Problem 9

1. **Aim:** Merge k sorted lists.

2. Implementation/Code:

```
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) {
            return nullptr;
        }
        return mergeKListsHelper(lists, 0, lists.size() - 1);
    }

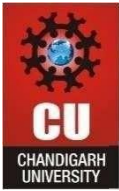
    ListNode* mergeKListsHelper(vector<ListNode*>& lists, int start, int end) {
        if (start == end) {
            return lists[start];
        }
        if (start + 1 == end) {
            return merge(lists[start], lists[end]);
        }
        int mid = start + (end - start) / 2;
        ListNode* 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;
                l1 = l1->next;
            } else {
                curr->next = l2;
                l2 = l2->next;
            }
            curr = curr->next;
        }

        curr->next = l1 ? l1 : l2;

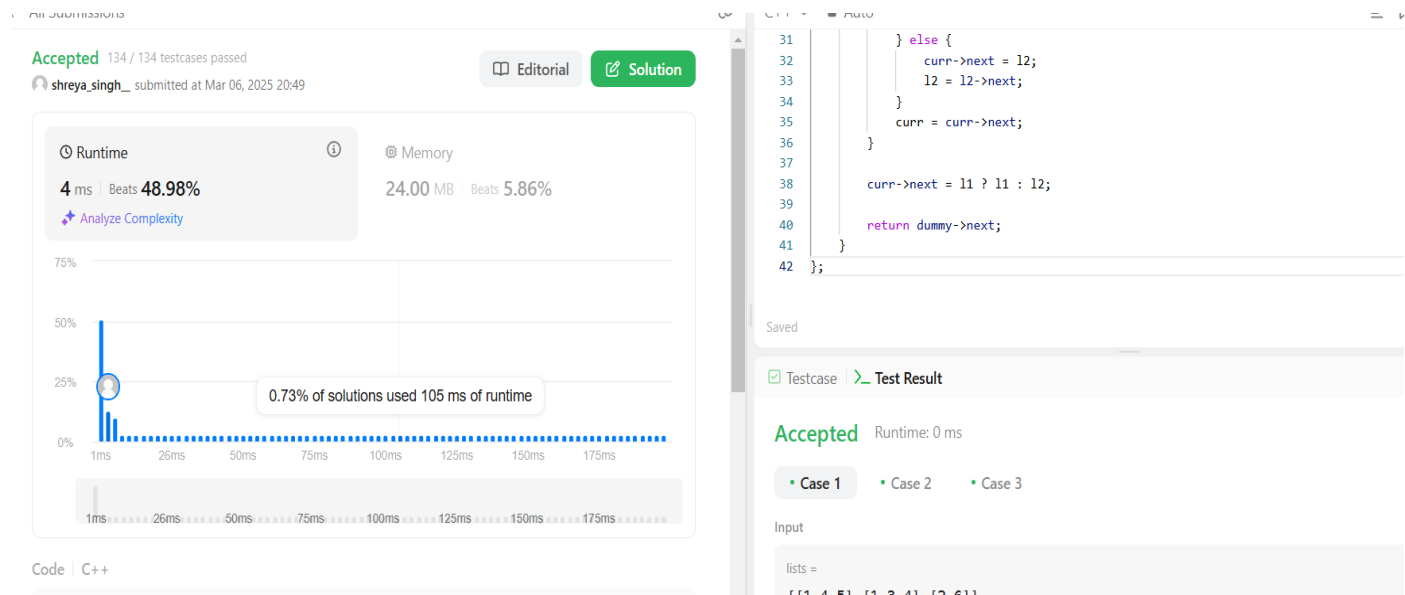
        return dummy->next;
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

3. Output:





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.