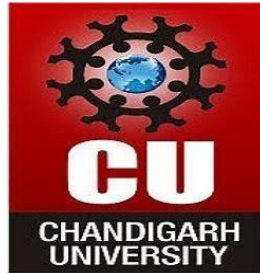


UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE/IT-5th Sem)



Subject Name: AP LAB-II

Subject Code: 22CSP-351

Submitted to:

Faculty name

Bhumika-E18199

Submitted by:

Name: Ayush Kumar Mayank

UID: 22BCS11193

Section: 22BCS-IOT-614

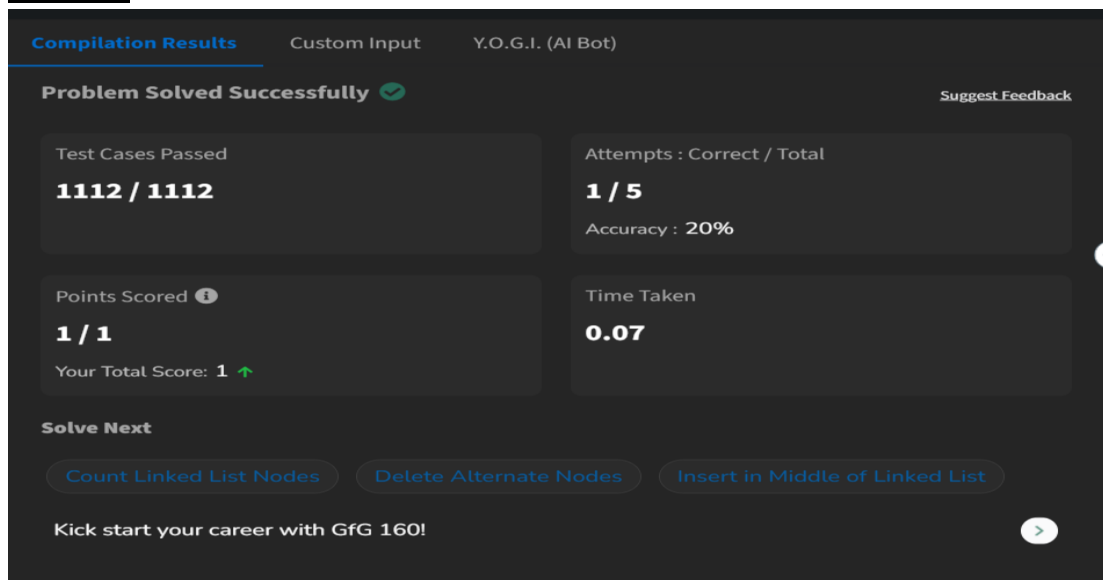
Group: B

1. Problem - Print linked list:

Code:

```
/* Solution class */
class Solution {
public:
    void printList(Node* head) {
        while (head) {
            cout << head->data << " ";
            head = head->next;
        }
    }
};
```

Output:



The screenshot displays the 'Compilation Results' tab of a coding platform. At the top, it shows 'Custom Input' and 'Y.O.G.I. (AI Bot)'. A green checkmark indicates the 'Problem Solved Successfully'. Below this, four statistics are shown in a grid: 'Test Cases Passed' (1112 / 1112), 'Attempts : Correct / Total' (1 / 5), 'Points Scored' (1 / 1), and 'Time Taken' (0.07). The 'Accuracy' is listed as 20%. A 'Suggest Feedback' link is present. Under the 'Solve Next' section, three buttons are visible: 'Count Linked List Nodes', 'Delete Alternate Nodes', and 'Insert in Middle of Linked List'. At the bottom, a message says 'Kick start your career with GfG 160!' with a right arrow button.

Test Cases Passed	Attempts : Correct / Total
1112 / 1112	1 / 5

Points Scored	Time Taken
1 / 1	0.07

Accuracy : 20%

Your Total Score: 1 ↑

Solve Next

- Count Linked List Nodes
- Delete Alternate Nodes
- Insert in Middle of Linked List

Kick start your career with GfG 160!

83.Problem - Remove duplicates from a 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 next node
            }
        }
    }
};
```

```

        return head;
    }
};

```

Output:

Case-1:

☒ Testcase
 |
 [Test Result](#)

Input

head =
 [1,1,2]

Output

[1,2]

Expected

[1,2]

[Contribute a testcase](#)

Case-2:

Input

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

Output

[1,2,3]

Expected

[1,2,3]

[Contribute a testcase](#)

206.Problem - [Reverse a linked list]:

Code:

```

class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* current = head;

        while (current) {
            ListNode* nextNode = current->next; // Store next node
            current->next = prev; // Reverse the link
            prev = current; // Move prev forward
            current = nextNode; // Move current forward
        }

        return prev; // New head of reversed list
    }
};

```

Output:

Case-1:

Testcase

Test Result

Input

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

Output

[5,4,3,2,1]

Expected

[5,4,3,2,1]

Case-2:

Testcase

Test Result

Input

head =
[1,2]

Output

[2,1]

Expected

[2,1]

Contribute a testcase

2095.Problem - Delete middle node of a list:

Code:

```
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
        if (!head || !head->next) return nullptr; // If only one node, 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; // Delete middle node
        delete slow;

        return head;
    }
};
```

Output:

Case-1:

Input

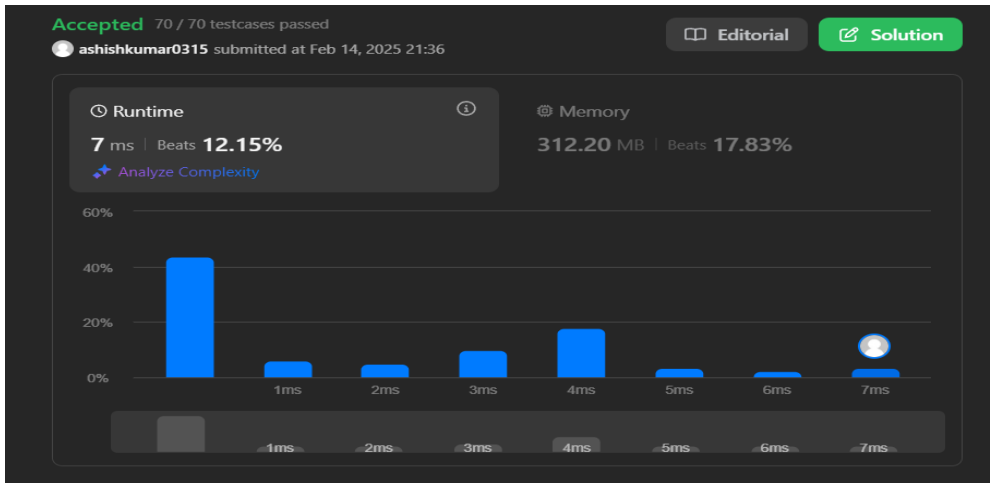
head =
[1,3,4,7,1,2,6]

Output

[1,3,4,1,2,6]

Expected

[1,3,4,1,2,6]



21. Problem - Merge two sorted linked lists:

Code:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if (!list1) return list2;
        if (!list2) return list1;

        ListNode* dummy = new ListNode(-1);
        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;
        }

        current->next = list1 ? list1 : list2;

        return dummy->next;
    }
}
```

};

Output:

Case-1:

Testcase

Test Result

Input

list1 =

[1,2,4]

list2 =

[1,3,4]

Output

[1,1,2,3,4,4]

Expected

[1,1,2,3,4,4]

Case-2:

Input

list1 =

[]

list2 =

[]

Output

[]

Expected

[]

Case-3:

list1 =

[]

list2 =

[0]

Output

[0]

Expected

[0]

Accepted:

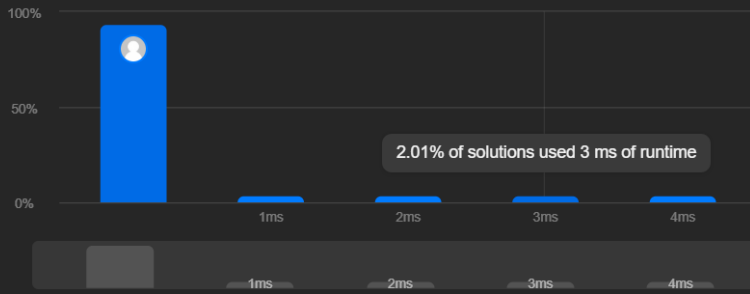
Runtime

0 ms | Beats 100.00%

[Analyze Complexity](#)

Memory

19.46 MB | Beats 62.56%



82. Problem - Remove duplicates from sorted lists 2:

Code:

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* dummy = new ListNode(0, head);
        ListNode* prev = dummy;

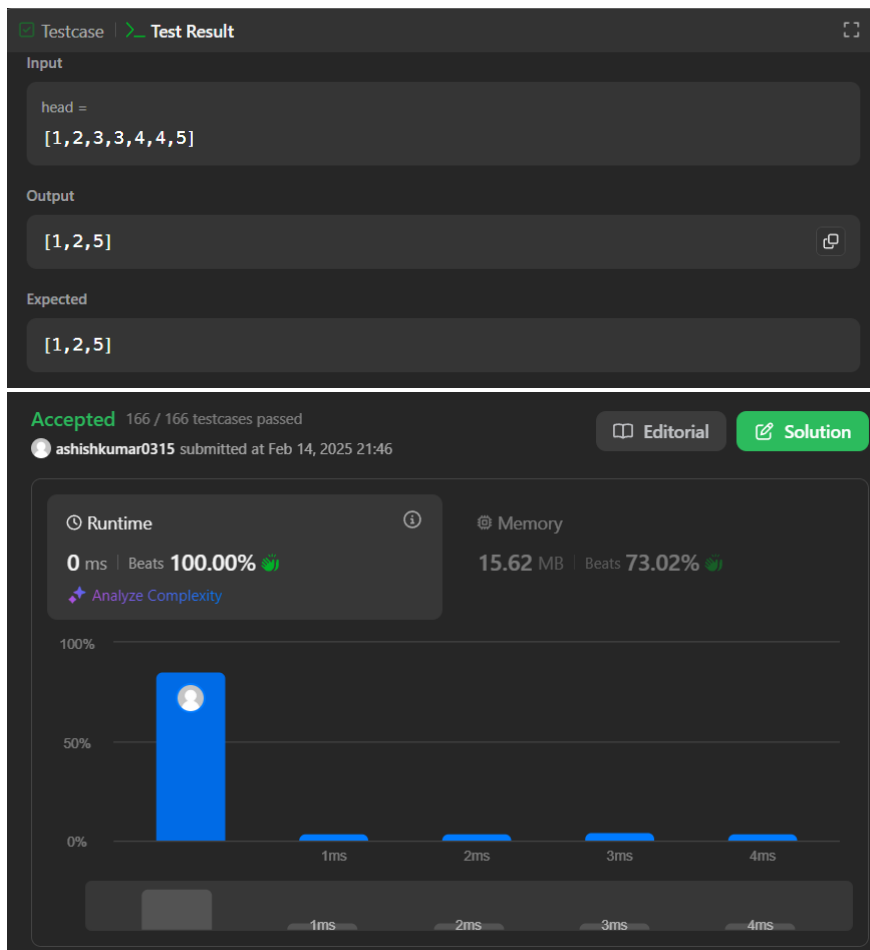
        while (head) {
            if (head->next && head->val == head->next->val) {

                while (head->next && head->val == head->next->val) {
                    head = head->next;
                }
                prev->next = head->next;
            } else {
                prev = prev->next;
            }
            head = head->next;
        }

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

Output:

Case-1:



141.Problem - Detect a cycle in a linked list:

Code:

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next) return false;

        ListNode* slow = head;
        ListNode* fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast) return true;
        }

        return false;
    }
};
```

Output:

Case-1:

Input

head =
[3,2,0,-4]

pos =
1

Output

true

Expected

Case-2:

Input

head =
[1,2]

pos =
0

Output

true

Expected

Case-3:

Input

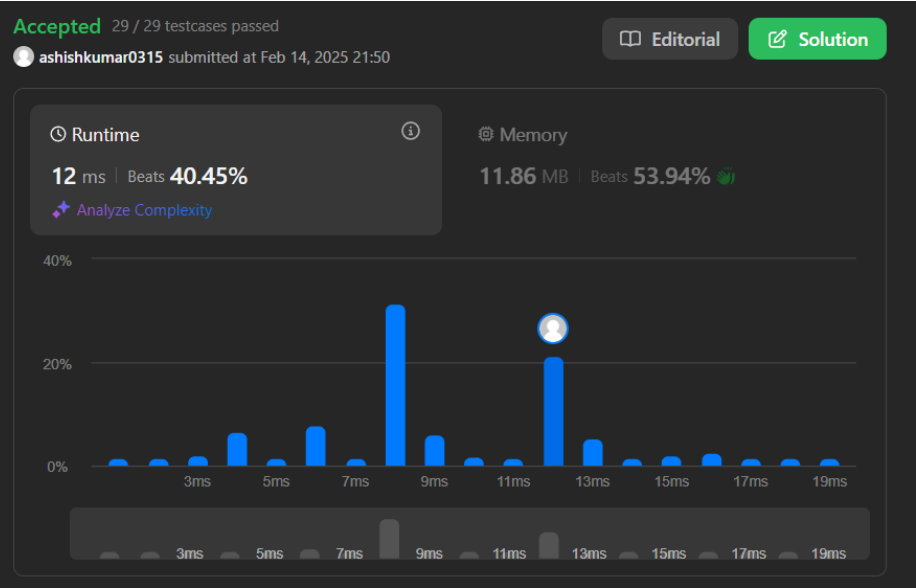
head =
[1]

pos =
-1

Output

false

Expected



92. Problem - Reverse linked list 2:

Code:

```

class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        if (!head || left == right) return head;

        ListNode* dummy = new ListNode(0, head);
        ListNode* prev = dummy;

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

        ListNode* curr = prev->next;
        ListNode* nextNode = nullptr;
        ListNode* prevNode = nullptr;

        for (int i = left; i <= right; ++i) {
            nextNode = curr->next;
            curr->next = prevNode;
            prevNode = curr;
            curr = nextNode;
        }

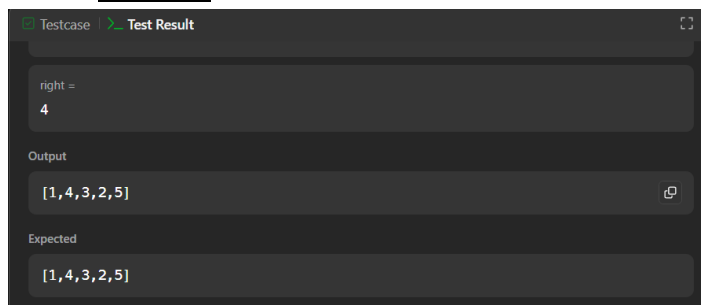
        prev->next->next = curr;
        prev->next = prevNode;

        return dummy->next;
    }
};

```

Output:

Case-1:



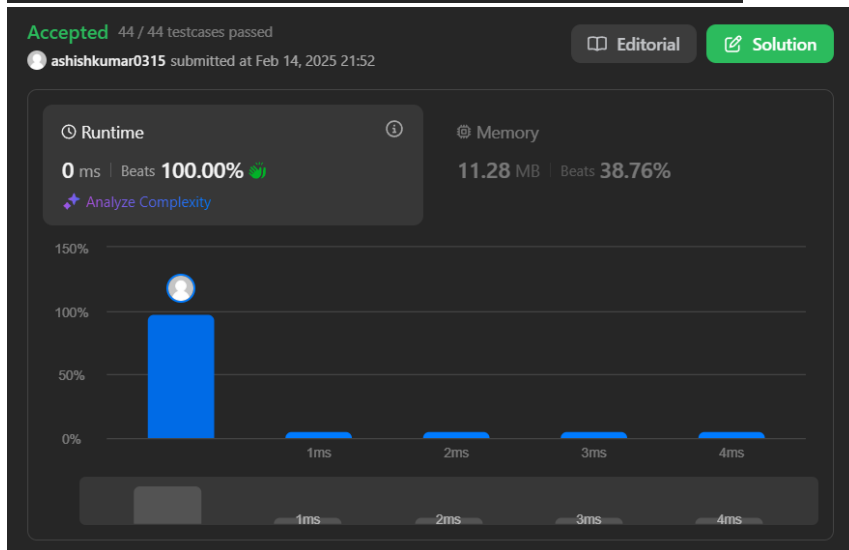
Case-2:

Testcase > Test Result

right =
1

Output
[5]

Expected
[5]



61.Problem - rotate a list:

Code:

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

        int length = 1;
        ListNode* tail = head;
        while (tail->next) {
            tail = tail->next;
            length++;
        }

        tail->next = head;

        k = k % length;
        int stepsToNewHead = length - k;
        ListNode* newTail = head;

        for (int i = 1; i < stepsToNewHead; i++) {
            newTail = newTail->next;
        }
    }
};
```

```

head = newTail->next;
newTail->next = nullptr;

return head;
}
};

```

Output:

Case-1:

Testcase | Test Result

k =
2

Output
[4, 5, 1, 2, 3]

Expected
[4, 5, 1, 2, 3]

Case-2:

Testcase | Test Result

k =
4

Output
[2, 0, 1]

Expected
[2, 0, 1]

Accepted 232 / 232 testcases passed

ashishkumar0315 submitted at Feb 14, 2025 21:58

Editorial Solution

Runtime 0 ms | Beats 100.00%

Memory 16.36 MB | Beats 65.01%

Analyze Complexity

100%
50%
0%

1ms 2ms 3ms 4ms

148. Problem - Sort List:

Code:

```

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

        ListNode* mid = findMiddle(head);
        ListNode* left = head;
        ListNode* right = mid->next;
        mid->next = nullptr;

        left = sortList(left);
        right = sortList(right);

        return merge(left, right);
    }

private:
    ListNode* findMiddle(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;
        ListNode* prev = nullptr;

        while (fast && fast->next) {
            prev = slow;
            slow = slow->next;
            fast = fast->next->next;
        }

        return prev;
    }

    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 = 11 ? 11 : 12;
```

```
return dummy.next;
```

```
}  
};
```

Output:

Case-1:

Testcase Test Result

Input

head =
[4, 2, 1, 3]

Output

[1, 2, 3, 4]

Expected

[1, 2, 3, 4]

Case-2:

Testcase Test Result

Input

head =
[-1, 5, 3, 4, 0]

Output

[-1, 0, 3, 4, 5]

Expected

[-1, 0, 3, 4, 5]

Case-3:

Testcase Test Result

Input

head =
[]

Output

[]

Expected

[]

Accepted 30 / 30 testcases passed

ashishkumar0315 submitted at Feb 14, 2025 22:02

Editorial Solution

Runtime 10 ms | Beats 87.70%

Memory 57.06 MB | Beats 86.19%

Analyze Complexity

Bar chart showing runtime distribution across various time intervals (11ms to 75ms).

142. Problem - Detect a cycle in a linked list 2:

Code:

```
class Solution {
```

public:

```
ListNode* detectCycle(ListNode* head) {  
    if (!head || !head->next) return nullptr;  
  
    ListNode* slow = head;  
    ListNode* fast = head;  
  
    while (fast && fast->next) {  
        slow = slow->next;  
        fast = fast->next->next;  
        if (slow == fast) {  
            break;  
        }  
    }  
  
    if (!fast || !fast->next) return nullptr;  
  
    slow = head;  
    while (slow != fast) {  
        slow = slow->next;  
        fast = fast->next;  
    }  
  
    return slow;  
}  
};
```

Output:

Case-1:



Case-2:



Case-3:

Testcase

Test Result

pos =

-1

Output

no cycle

Expected

no cycle

Accepted

18 / 18 testcases passed

ashishkumar0315

submitted at Feb 14, 2025 22:06

Editorial

Solution

Runtime

8 ms | Beats 49.08%

Analyze Complexity

Memory

11.32 MB | Beats 53.90%

Analyze Complexity

Runtime (ms)	Percentage (%)
3ms	1%
4ms	1%
5ms	10%
6ms	5%
7ms	19%
8ms	2%
9ms	1%
10ms	27%
11ms	9%
12ms	1%
13ms	1%