Advanced Pragramming

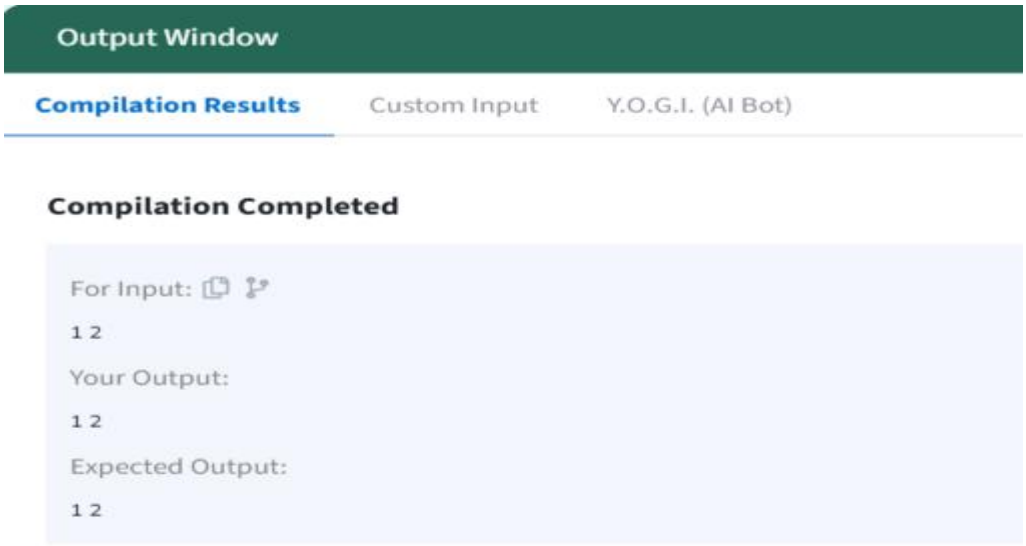
ASSIGNMENT 01

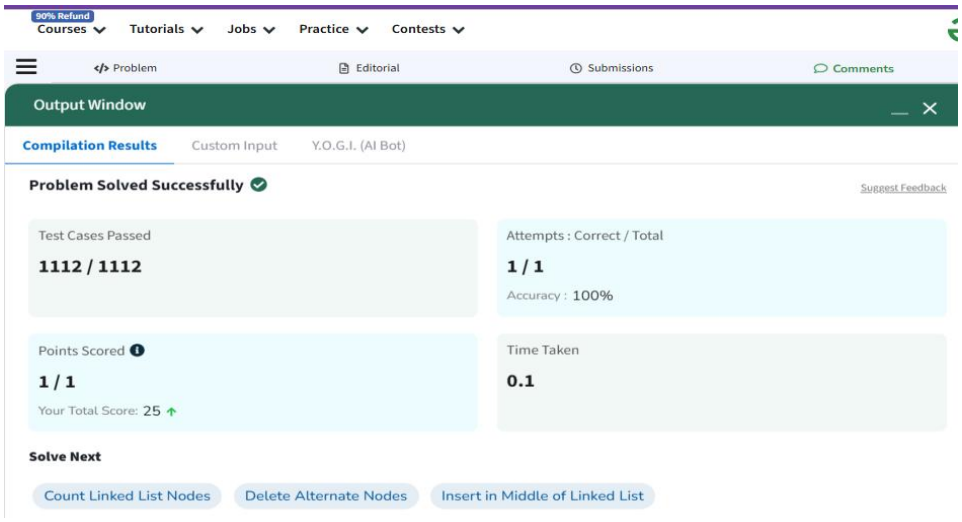
Q1. Print Linked list.

Code:

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

    }
};
// } Driver Code Ends
```

Output:



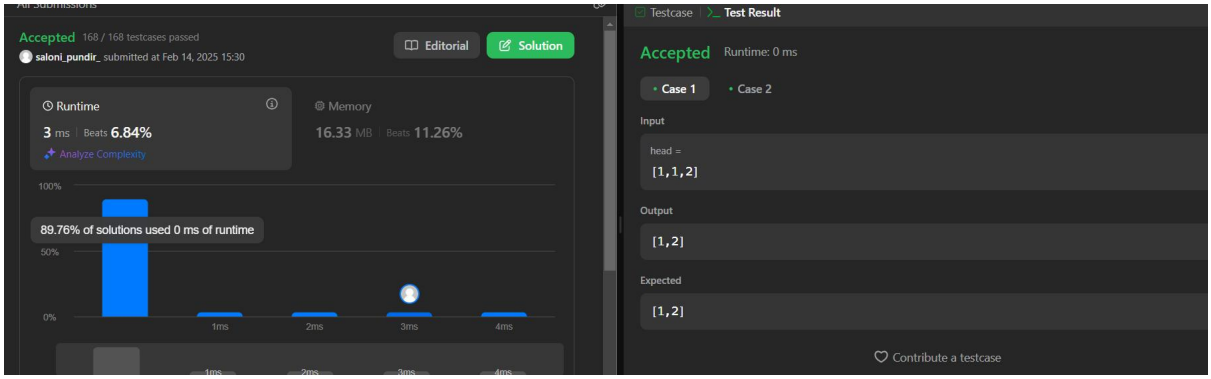


Q2. Remove duplicates from a sorted list.

Code:

```
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;
        while(current && current-> next){
            if(current->val == current->next->val){
                ListNode* temp = current->next;
                current->next = current->next->next;
                delete temp;
            }
            else
            current = current->next;
        }
        return head;

    }
};
```

Output:

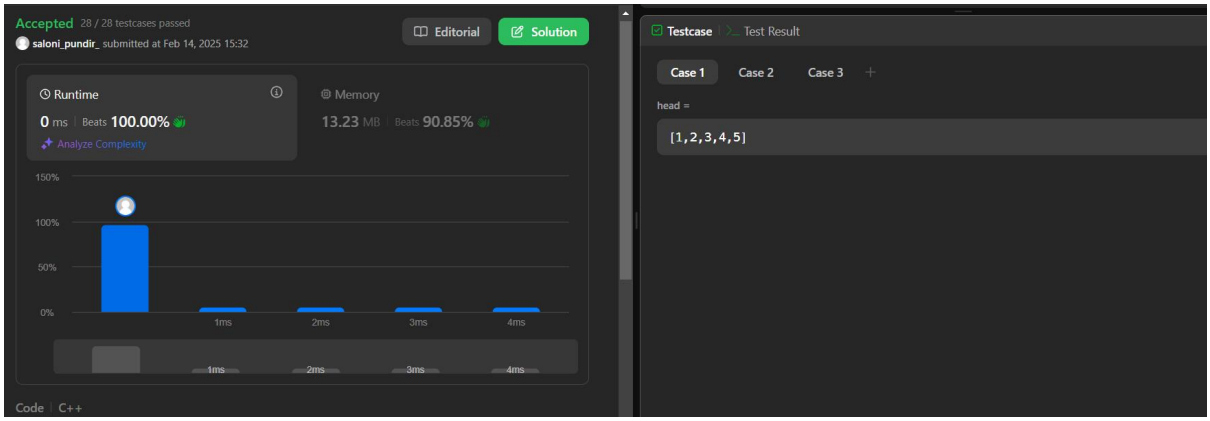


Q3. Reverse a linked list.

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;
    }
};
```

Output:

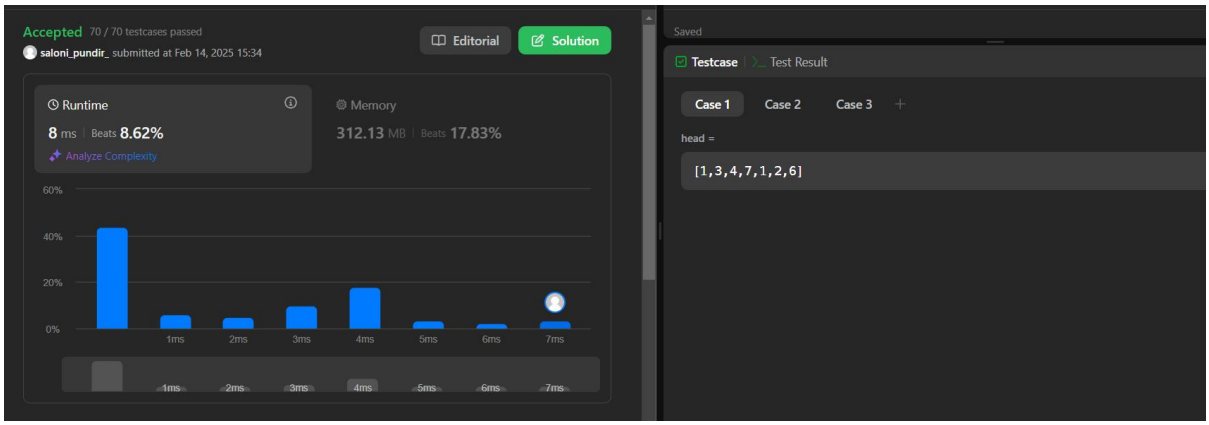


Q4. Delete middle node of a list.

Code:

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

    }
};
```

Output:

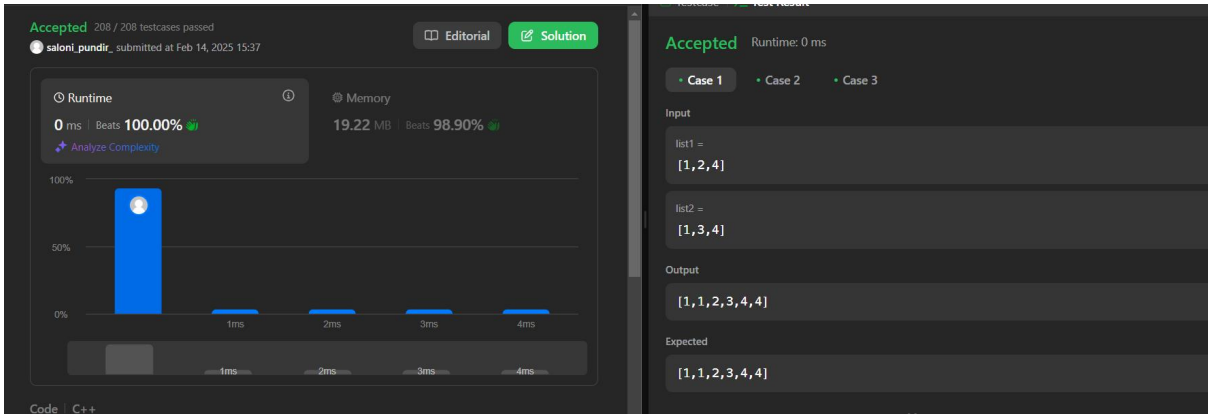


Q5. Merge two sorted linked list.

Code:

```
Code
C++ Auto
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if (!list1) return list2;
        if (!list2) return list1;

        if (list1->val < list2->val) {
            list1->next = mergeTwoLists(list1->next, list2);
            return list1;
        } else {
            list2->next = mergeTwoLists(list1, list2->next);
            return list2;
        }
    }
};
```

Output:

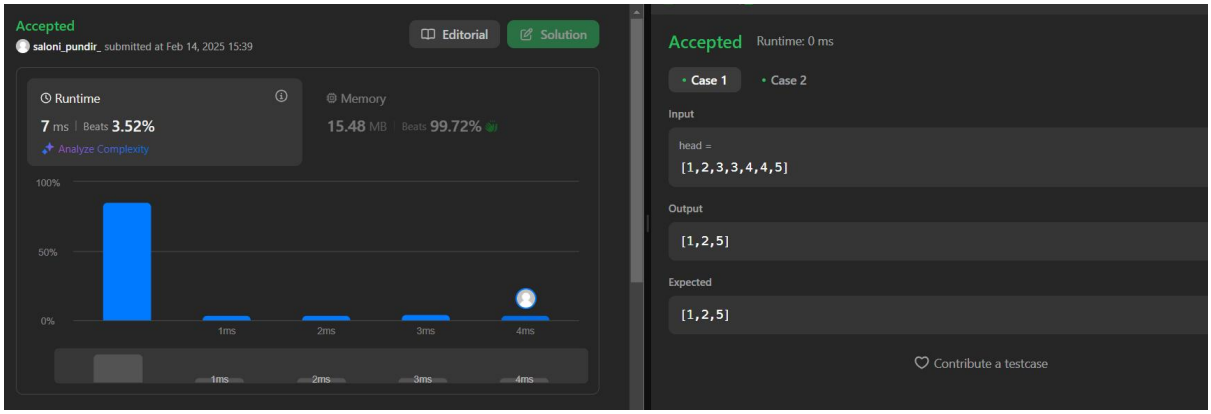


Q6. Remove duplicates from sorted list 2.

Code:

```cpp
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (head == nullptr || head->next == nullptr) {
            return head;
        }

        ListNode* dummy = new ListNode(0, head);
        ListNode* prev = dummy;
        ListNode* curr = head;
        ListNode* temp = head->next;
        bool flag = false;

        while (temp != nullptr) {
            if (curr->val != temp->val) {
                if (flag) {
                    prev->next = temp;
                    flag = false;
                } else {
                    prev = prev->next;
                }
            } else {
                flag = true;
            }
            temp = temp->next;
            curr = curr->next;
        }

        if (flag) {
            prev->next = temp;
        }

        return dummy->next;

    }
};
```

Output:

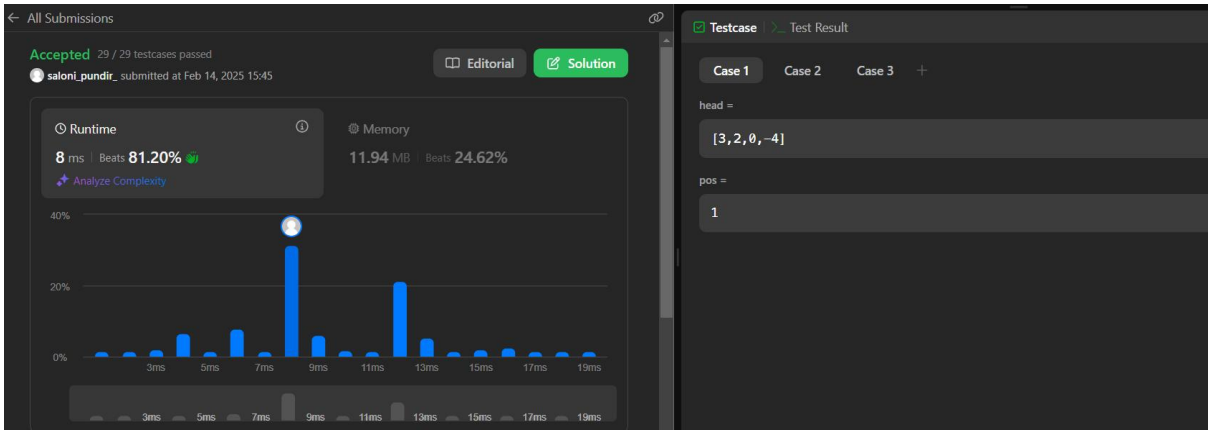


Q7. Detect a cycle in a linked list.

Code:

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

};
```

Output:

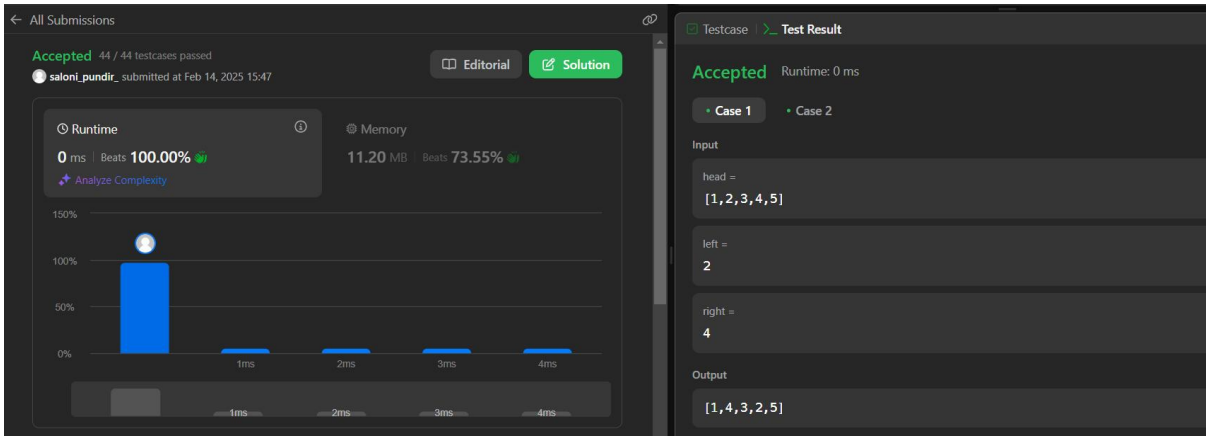


Q8. Reverse linked list 2.

Code:

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

        ListNode* dummy = new ListNode(0);
        dummy->next = head;
        ListNode* prev = dummy;

        for (int i = 0; i < left - 1; i++) {
            prev = prev->next;
        }

        ListNode* cur = prev->next;

        for (int i = 0; i < right - left; i++) {
            ListNode* temp = cur->next;
            cur->next = temp->next;
            temp->next = prev->next;
            prev->next = temp;
        }

        return dummy->next;
    }
```

Output:

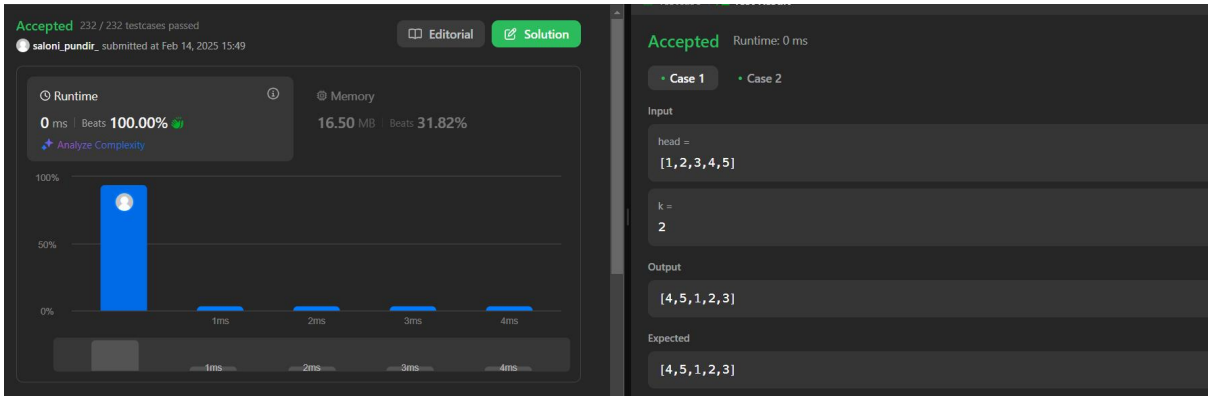


Q9. Rotate a list.

Code:

```cpp
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {

        if (!head) return nullptr;
        int n = 1;
        ListNode* temp = head;
        while (temp->next) {
            n++;
            temp = temp->next;
        }
        k %= n;
        temp->next = head;
        for (int i = 0; i < n - k; i++) temp = temp->next;
        ListNode* new_head = temp->next;
        temp->next = nullptr;
        return new_head;
    }
};
```

Output:

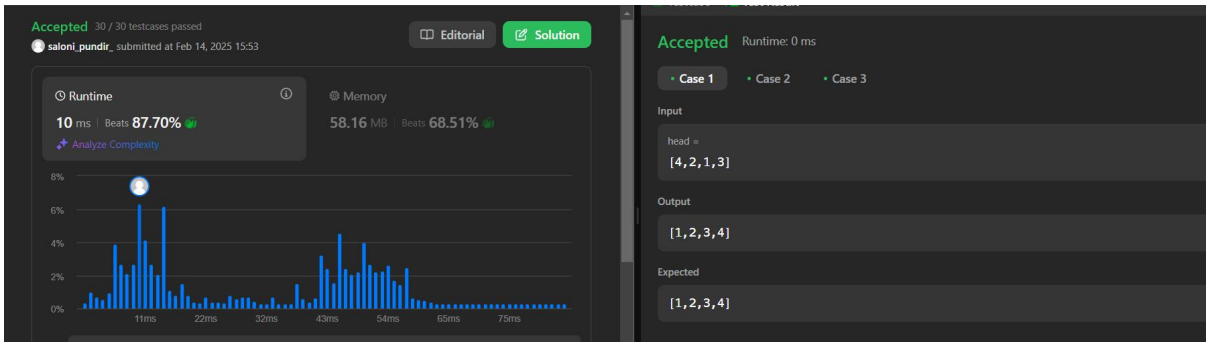


Q10. Sort list

Code:

```
C++ Auto
20  */
21  class Solution {
22  public:
23      ListNode* sortList(ListNode* head) {
24          vector<int> arr;
25          ListNode* temp = head;
26          while(temp != nullptr){
27              arr.push_back(temp->val);
28              temp = temp->next;
29          }
30          sort(arr.begin(), arr.end());
31          temp = head;
32          for(int i = 0; temp != nullptr; i++){
33              temp->val = arr[i];
34              temp = temp->next;
35          }
36          return head;
37
38      }
39  };
```

Output:

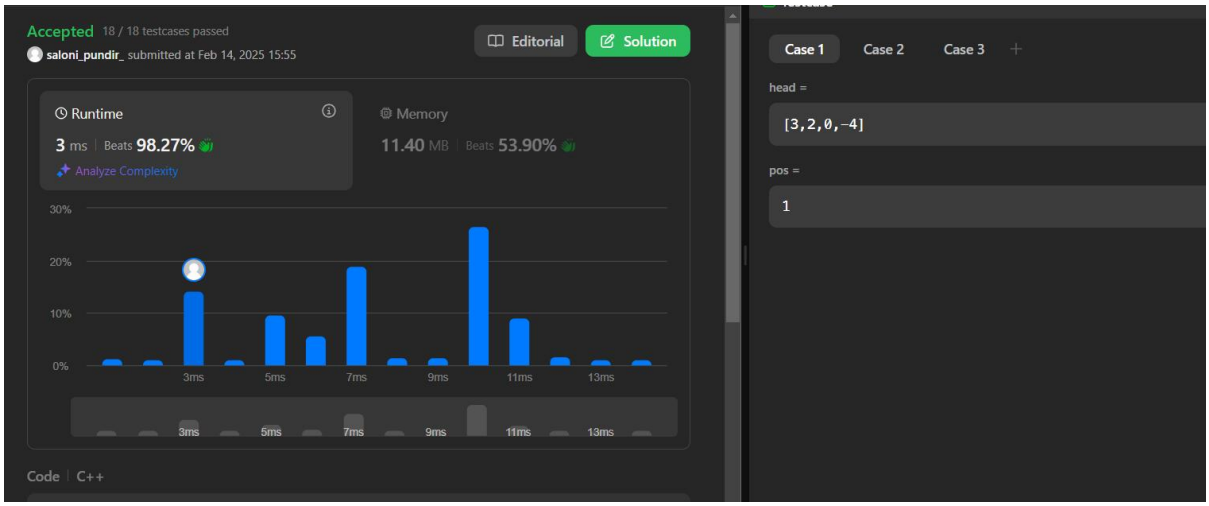


Q11. Detect a cycle in linked list 2.

Code:

```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {

        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;

        fast = head;
        while (fast != slow) {
            fast = fast->next;
            slow = slow->next;
        }

        return slow;
    }
};
```

Output:


Accepted 18 / 18 testcases passed
saloni_pundir_ submitted at Feb 14, 2025 15:55
Editorial
Solution
Runtime
3 ms | Beats 98.27%
Analyze Complexity
Memory
11.40 MB | Beats 53.90%
30%
20%
10%
0%
3ms
5ms
7ms
9ms
11ms
13ms
Code | C++
Case 1
Case 2
Case 3
head =
[3,2,0,-4]
pos =
1