**Problem-2**

**Aim:**

Given the head of a singly linked list, return true if it is a palindrome or false otherwise.

Constraints:

The number of nodes in the list is in the range [1, 105].

0 <= Node.val <= 9

1. **Problem Description :**

A singly linked list is given and its known head and we need to check whether the provided linked list is a palindrome linked list or not. Now, palindrome means any word or sequence that reads the same forwards as backward. For example mom is a palindrome as it reads the same forwards as backward.

We will use the two pointers slow and fast to find the middle of the linked list. Slow pointer is incremented by 1 and fast pointer is incremented by 2 in every step. Then we will reverse the linked list after the middle node and compare the both halves. If both halves are equal then linked list is palindrome and return true otherwise return false.

1. **Algorithm :**

Linked list is empty then

if(head -> next == NULL) {

return true;

}.

Middle of the linked list.

while(fast != NULL && fast-> next != NULL) {

fast = fast -> next -> next;

slow = slow -> next;

}

return slow;

Reverse the linked list after middle node.

Compare the both halves of linked list.

while(head2 != NULL) {

if(head2->val != head1->val) {

return 0;

}

head1 = head1 -> next;

head2 = head2 -> next;

}

After traversing whole linked list return true i.e. palindrome.

1. **Source Code for Experiment :**

class Solution {

public:

    bool isPalindrome(ListNode\* head) {

        ListNode \*slow = head, \*fast = head;

        while(fast!=NULL && fast->next!=NULL){

            slow = slow->next;

            fast = fast->next->next;

        }

        if(fast != NULL){

            //the length of list is odd

            //in this situation we should ignore the middle node

            slow = slow->next;

        }

        //slow: later part of original list

        //fast: former part of original list

        slow = reverse(slow);

        fast = head;

        while(slow != NULL){

            if(slow->val != fast->val) return false;

            slow = slow->next;

            fast = fast->next;

        }

        return true;

    }

    ListNode\* reverse(ListNode\* head){

        //reverse the later part of original list

        ListNode\* prev = NULL;

        ListNode\* tmp;

        while(head != NULL){

            tmp = head->next;

            head->next = prev;

            prev = head;

            head = tmp;

        }

        //when the loop finishes, head is NULL,

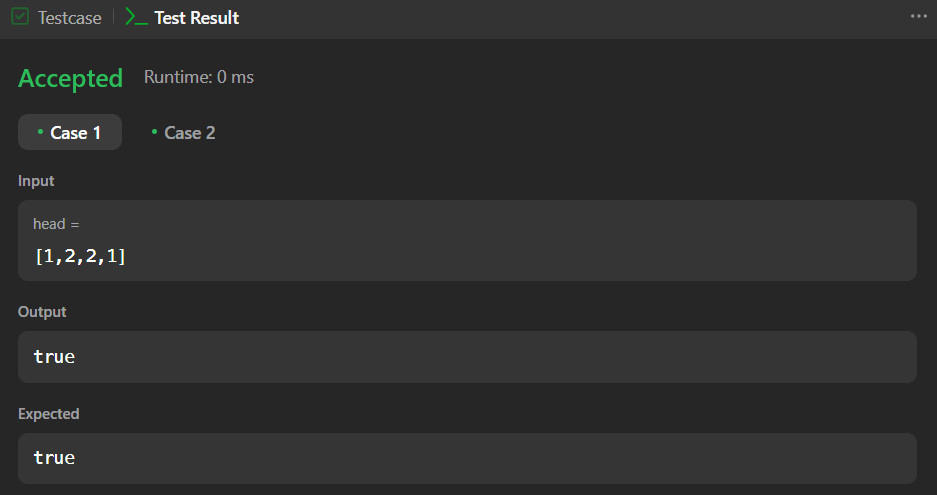
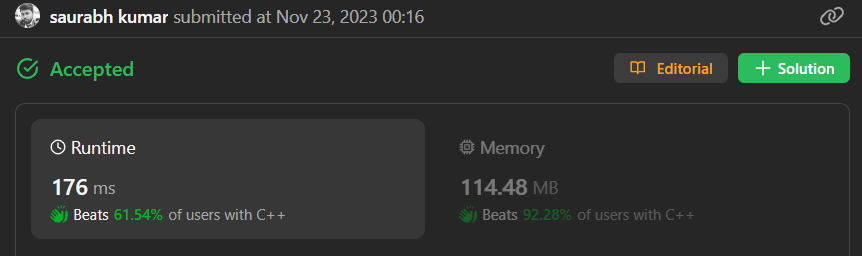
        //and prev is the last node in the original list

        return prev;

    }

};

1. **Result/Output :**

****

**Learning outcomes (What I have learnt):**

1. Learnt about the linked list data structure in C++.
2. Learnt about the fast and slow pointers approach.
3. Learnt about how to find the middle node in linked list.
4. Learnt about how to reverse the linked list.
5. Learnt about how to check palindrome linked list.