**Problem – 2**

1. **Aim of the Experiment :**

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

1. **Objective of the Experiment :**

To determine the palindrome or not.

1. **Explanation:**

Case 1: If we read elements from left to right, we get [1,2,3,3,2,1]. When we read elements from right to left, we get [1,2,3,3,2,1]. Both ways list remains same and hence, the given linked list is palindrome.

Case 2: In the example of [1,2]. Reading from right to left, we get a list as [2,1]. Both are different and hence, the given linked list is not palindrome.

1. **Algorithm:**

Step 1: If linked list is empty then

if(head -> next == NULL) {

return true;

}.

Step 2: Find middle of the linked list.

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

fast = fast -> next -> next;

slow = slow -> next;

}

return slow;

Step 3: Reverse the linked list after middle node.

Step 4: Compare the both halves of linked list.

while(head2 != NULL) {

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

return 0;

}

head1 = head1 -> next;

head2 = head2 -> next;

}

Step 5: After traversing whole linked list return true i.e. palindrome.

1. **Code :**

class Solution {

public:

    bool isPalindrome(ListNode\* head) {

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

        while (fast && fast->next)

            slow = slow->next, fast = fast->next->next;

        prev = slow, slow = slow->next, prev->next = NULL;

        while (slow)

            temp = slow->next, slow->next = prev, prev = slow, slow = temp;

        fast = head, slow = prev;

        while (slow)

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

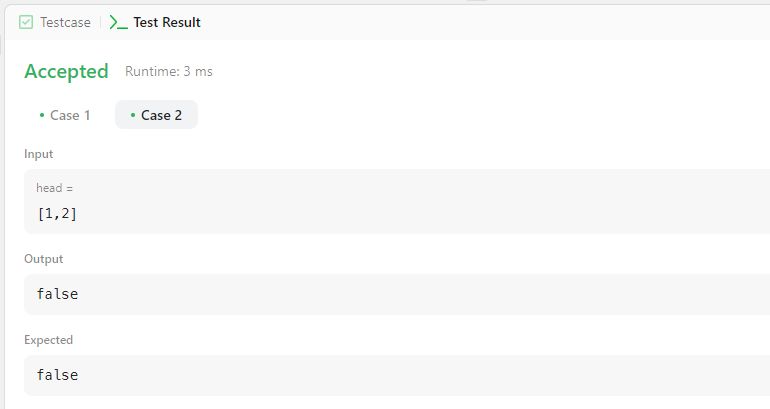
            else fast = fast->next, slow = slow->next;

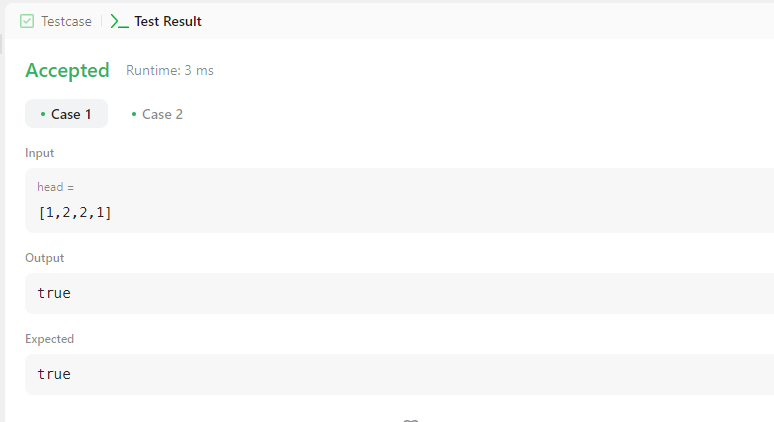
        return true;

    }

};

1. **Output:**

****

****

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

1. **Learned how to use the vectors in C++.**
2. **Learned how to found whether a list is palindrome or not using linked list.**