Problem-2

Aim of the Experiment :

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 head is known to user. We have to check whether the linked list is palindrome or not. If linked list is palindrome then return true otherwise return false.

We will use the two pointers which are 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.

2. 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;
}
```

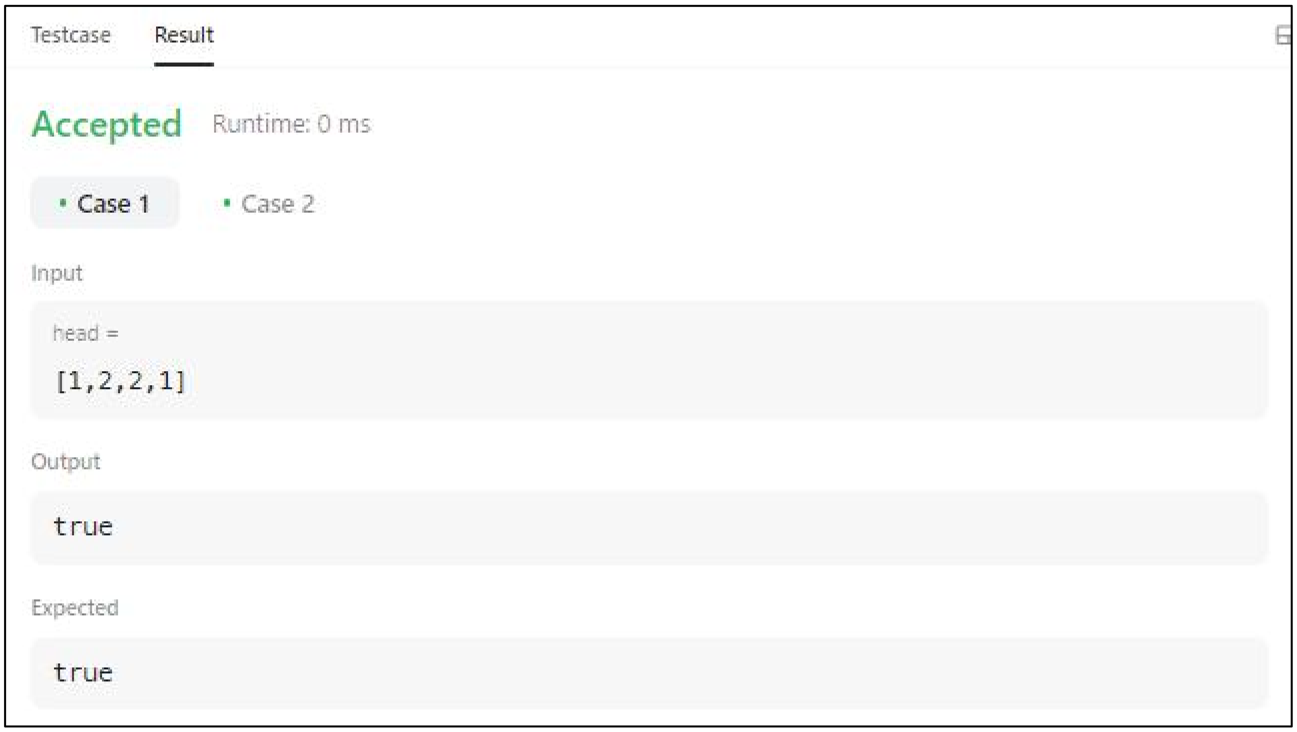
Step 5: After traversing whole linked list return true i.e. palindrome.

3. Source Code for Experiment :

```
class Solution {
private:
   ListNode* getMid(ListNode* head ) {
      ListNode* slow = head;
      ListNode* fast = head -> next;

      while(fast != NULL && fast-> next != NULL) {
         fast = fast -> next -> next;
         slow = slow -> next;
      }
      return slow;
   }
   ListNode* reverse(ListNode* head) {
      ListNode* curr = head;
      ListNode* prev = NULL;
      ListNode* next = NULL;

      while(curr != NULL) {
         next = curr -> next;
         curr -> next = prev;
         prev = curr;
         curr = next;
```

```
        }
        return prev;
    }
public:
    bool isPalindrome(ListNode* head) {
        if(head -> next == NULL) {
            return true;
        }

        //step 1 -> find Middle
        ListNode* middle = getMid(head);
        //cout << "Middle " << middle->data << endl;

        //step2 -> reverse List after Middle
        ListNode* temp = middle -> next;
        middle -> next = reverse(temp);

        //step3 - Compare both halves
        ListNode* head1 = head;
        ListNode* head2 = middle -> next;

        while(head2 != NULL) {
            if(head2->val != head1->val) {
                return 0;
            }
            head1 = head1 -> next;
            head2 = head2 -> next;
        }

        //step4 - repeat step 2
        temp = middle -> next;
        middle -> next = reverse(temp);

        return true;
    }
};
```

4. Result/Output :


Testcase Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

head =

```
[1,2,2,1]
```

Output

```
true
```

Expected

```
true
```

Learning outcomes (What I have learnt):

1. I learnt about the linked list data structure in C++.

2. I learnt about the fast and slow pointers in linked list.

3. I learnt about how to find the middle node in linked list.

4. I learnt about how to reverse the linked list.

5. I learnt about how to check palindrome linked list.