DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Experiment 2

Student Name: Rhythm Tyagi
UID: 22BCS17203
Branch: CSE
Section: NTPP_602-A
Semester: 6th
DOP: 23/01/25
Subject: AP-LAB-2
Subject Code:22CSP-351

Aim:

Problem 2.1: **Reverse Linked List**

Given the head of a singly linked list, reverse the list, and return *the reversed list*

Problem 2.2: **Rotate List**

Given the head of a linked list, rotate the list to the right by k places

Problem 2.3: **Remove Duplicates from Sorted List**

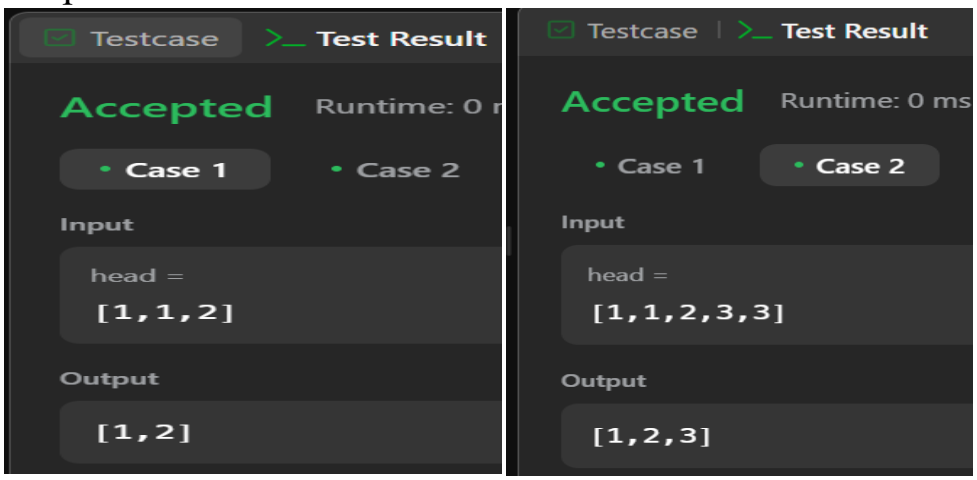
Given the head of a sorted linked list, *delete all duplicates such that each element appears only once*. Return *the linked list **sorted** as well*.

Code: 2.1

```
class Solution {
  public ListNode deleteDuplicates(ListNode head) {
    ListNode current = head;

    while (current != null && current.next != null) {
      if (current.val == current.next.val) {
        current.next = current.next.next; // Skip duplicate
      } else {
        current = current.next;
      }
    }
    return head;
  }
}
```

Output:

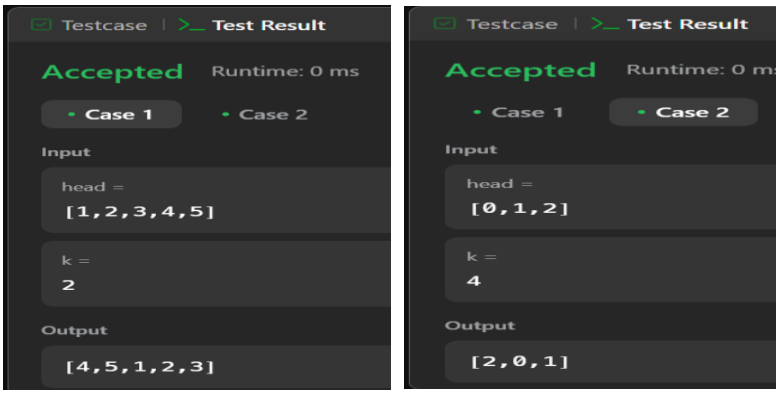


CODE: 6.2

```
class Solution {
  public ListNode rotateRight(ListNode head, int k) {
    if (head == null || head.next == null || k == 0) return head; // Edge cases
    int length = 1; // Start at 1 because we will traverse the list
    ListNode tail = head;
    while (tail.next != null) {
      tail = tail.next;
      length++;
    }
    tail.next = head;
    k = k % length;
    int newTailPosition = length - k;
    ListNode newTail = head;
    for (int i = 1; i < newTailPosition; i++) {
      newTail = newTail.next;
    }
    head = newTail.next;
    newTail.next = null;
    return head;
  }
```

}

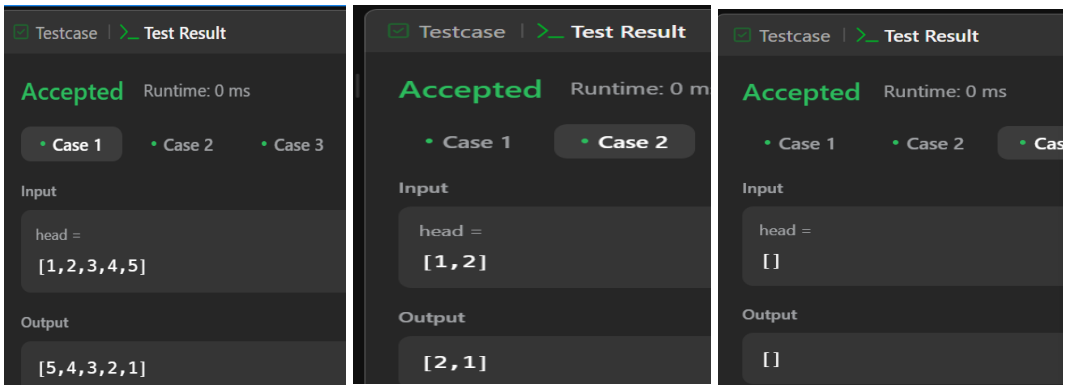
OUTPUT:



CODE: 6.3

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;
        while (current != null) {
            ListNode nextNode = current.next; // Store the next node
            current.next = prev; // Reverse the link
            prev = current; // Move prev to current node
            current = nextNode; // Move to the next node
        }
        return prev; // New head of the reversed list
    }
}
```

OUTPUT:



Learning Outcomes:

1. Linked List Manipulation: Learned to modify linked lists by merging, reversing, and removing duplicates.
2. Pointer Handling: Gained experience in handling next pointers efficiently to traverse and manipulate linked lists.
3. Iterative vs. Recursive Approaches: Understood both iterative (O(1) space) and recursive (O(n) space) methods for linked list operations.
4. Time & Space Complexity Analysis: Evaluated and optimized algorithms based on efficiency considerations.
5. LeetCode Submission Guidelines: Learned how to submit solutions correctly by avoiding redundant ListNode definitions and handling input/output as per platform requirements