Experiment 3(a)

Student Name: Neetu **UID:** 22BCS17083
Branch: BE-CSE **Section/Group:** 641- A
Semester: 6th **Date of Performance:** 02/02/2025
Subject Name: Advanced Programming-II **Subject Code:** 22CSH-312

1. **Aim:** You are given the heads of two sorted linked lists list1 and list2.Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists. Return the head of the merged linked list.

2. **Objective:** The goal is to merge two sorted linked lists into a single sorted list by efficiently splicing together their nodes. The merged list should maintain the original sorted order of the input lists. This helps in understanding linked list traversal, pointer manipulation, and efficient merging techniques.

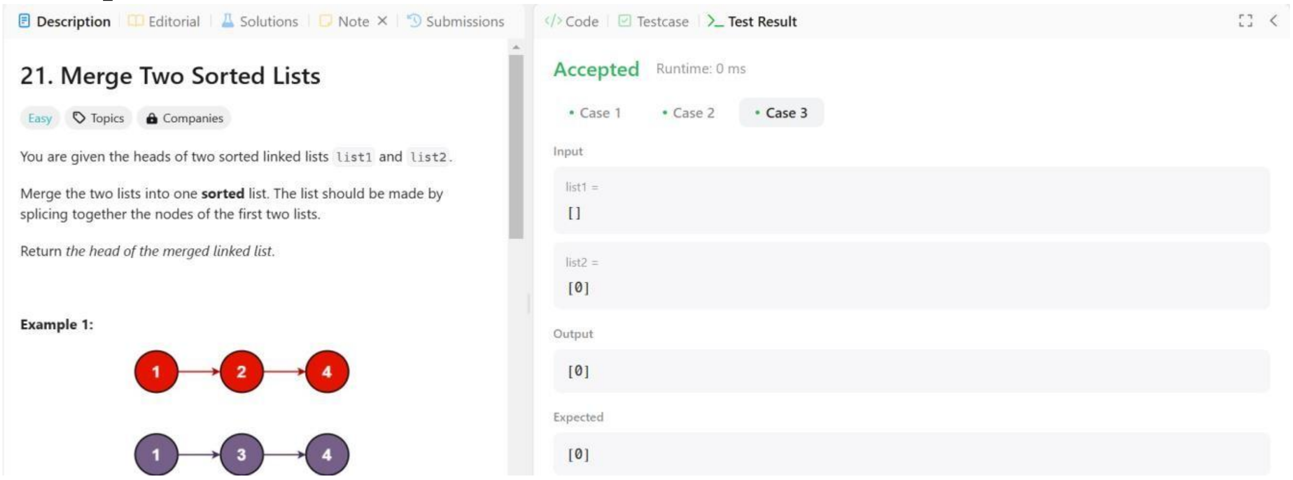
3. **Implementation/Code:**
#The code is written in Java language: class

```
Solution {
   public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
      ListNode dummy = new ListNode(-1);
      ListNode current = dummy;

      while (list1 != null && list2 != null) {
         if (list1.val <= list2.val) {
current.next = list1;              list1
= list1.next;
         } else {
current.next = list2;
list2 = list2.next;
         }
         current = current.next;
      }
      if (list1 != null) {
```

```
                current.next = list1;
            } else {
                current.next = list2;
            }
            return dummy.next;
        }
    }
```

4. Output:



21. Merge Two Sorted Lists

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:

Accepted Runtime: 0 ms

Input

```
list1 =
[]
list2 =
[0]
```

Output

```
[0]
```

Expected

```
[0]
```

5. Learning Outcomes:

- Understand how to traverse and merge two sorted linked lists.
- Implement an efficient approach to handle linked list merging without extra space.
- Learn to manage edge cases like empty lists or lists of different lengths.
- Improve problem-solving skills using iterative and recursive approaches.
- Strengthen understanding of linked list operations and pointer manipulation.

Experiment 3(b)

1. **Aim:** Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list sorted as well.

2. **Objective:** The goal is to remove all nodes with duplicate values from a sorted linked list, ensuring only distinct numbers remain. The modified list should maintain its sorted order. This helps in understanding linked list traversal, handling duplicates efficiently, and improving problem-solving skills in linked list operations.

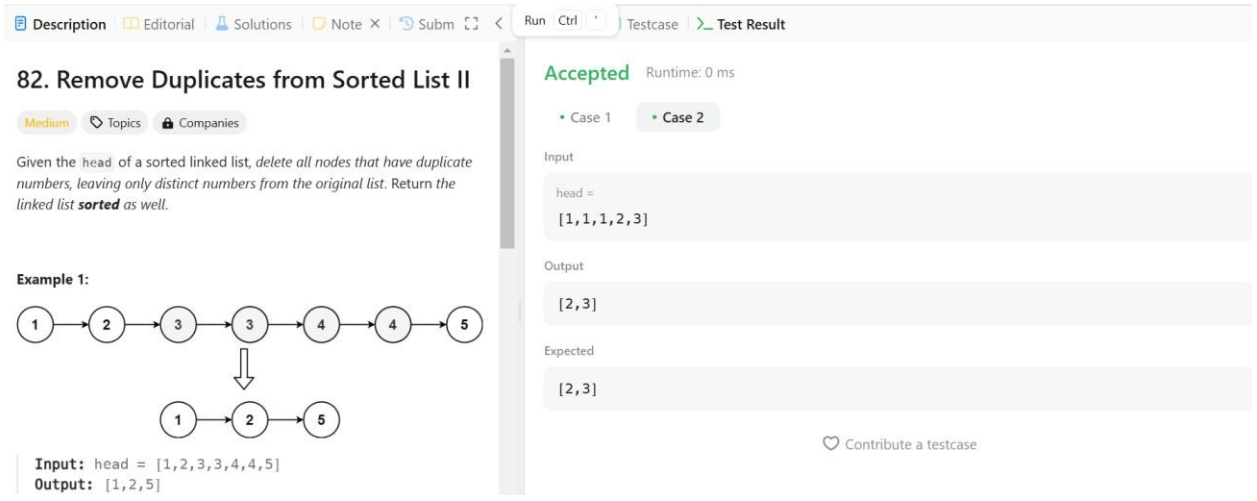
3. **Implementation/Code:**
   #The code is written in Java language:

```
    class    Solution    {                    public    ListNode
 deleteDuplicates(ListNode head) {
        ListNode dummy = new ListNode(0, head);
        ListNode prev = dummy;

        while (head != null) {            if (head.next != null &&
 head.val == head.next.val) {             while (head.next != null &&
 head.val == head.next.val) {                  head = head.next;
                }
                prev.next = head.next;
            } else {
                prev = prev.next;
            }
            head = head.next;
        }

        return dummy.next;
    }
}
```

4. Output:



82. Remove Duplicates from Sorted List II

Given the `head` of a sorted linked list, *delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list*. Return *the linked list* ***sorted*** *as well*.

Example 1:

```
Input: head = [1,2,3,3,4,4,5]
Output: [1,2,5]
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
head =
[1,1,1,2,3]
```

Output

```
[2,3]
```

Expected

```
[2,3]
```

5. Learning Outcomes:

- Understand how to traverse a sorted linked list and identify duplicates.
- Implement an efficient approach to remove duplicate nodes while maintaining order.
- Learn pointer manipulation techniques for modifying linked lists in place.
- Develop problem-solving skills by handling edge cases like consecutive duplicates.
- Improve understanding of time complexity in linked list operations.