1.<u>Merge k Sorted Lists</u>

You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:

Input: lists = [[1,4,5],[1,3,4],[2,6]]

Output: [1,1,2,3,4,4,5,6]

Explanation: The linked-lists are:

```
[
  1->4->5,
  1->3->4,
  2->6
]
```

merging them into one sorted list:

1->1->2->3->4->4->5->6

Program:

```
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        List<Integer> values = new ArrayList<>();
        for (ListNode head : lists) {
            while (head != null) {
                values.add(head.val);
                head = head.next;
            }
        }
        Collections.sort(values);
        ListNode dummy = new ListNode(0);
        ListNode c = dummy;

        for (int val : values) {
            c.next = new ListNode(val);
            c = c.next;
        }
        return dummy.next;
    }
}
```

2.Delete Node in a Linked List

There is a singly-linked list head and we want to delete a node node in it.

You are given the node to be deleted node. You will **not be given access** to the first node of head.

All the values of the linked list are **unique**, and it is guaranteed that the given node node is not the last node in the linked list.

Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:

- The value of the given node should not exist in the linked list.
- The number of nodes in the linked list should decrease by one.
- All the values before node should be in the same order.
- All the values after node should be in the same order.

Custom testing:

- For the input, you should provide the entire linked list head and the node to be given node. node should not be the last node of the list and should be an actual node in the list.
- We will build the linked list and pass the node to your function.
- The output will be the entire list after calling your function.

  Input: head = [4,5,1,9], node = 5

Output: [4,1,9]

Explanation: You are given the second node with value 5, the linked list should become 4 -> 1 -> 9 after calling your function.

Program:

```
class Solution {
   public void deleteNode(ListNode node) {


       node.val=node.next.val;

       node.next=node.next.next;

      }

      }
```

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*.

Input: head = [1,1,2]

Output: [1,2]

Constraints:

- The number of nodes in the list is in the range [0, 300].
- -100 <= Node.val <= 100
- The list is guaranteed to be **sorted** in ascending order.

```
       return dummy.next;
```

Program:

```
class Solution {
   public ListNode deleteDuplicates(ListNode head) {
       ListNode temp=head;
       while(temp!=null&& temp.next!=null){
          while(temp.next!=null&& temp.val==temp.next.val){


             temp.next=temp.next.next;
          }
```

```
            temp=temp.next;
        }

        return head;
    }
}
```