23 Answer

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */

class Solution {
    public ListNode Merge(ListNode Current, ListNode Prev) {
            ListNode dummy = new ListNode(0);
            ListNode NewHead = dummy;
            while(Current != null && Prev != null) {
                if(Current.val <= Prev.val) {
                    NewHead.next = Current;
                    Current = Current.next;
                }
                else {
                    NewHead.next = Prev;
                    Prev = Prev.next;
                }
                NewHead = NewHead.next;
            }
            if(Current != null) {
                NewHead.next = Current;
            }
            else {
                NewHead.next = Prev;
            }
            return dummy.next;
    }
    public ListNode mergeKLists(ListNode[] lists) {
        if(lists.length == 0) {
            return null;
        }
```

```
        for(int i = 1; i < lists.length; i++) {
            lists[i] = Merge(lists[i], lists[i - 1]);
        }
        return lists[lists.length - 1];
    }
}
```

237 Answer

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public void deleteNode(ListNode node) {
        if(node == null || node.next == null) {
            return;
        }
        node.val = node.next.val;
        node.next = node.next.next;
    }
}
```

83

```
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;
            } else {
                current = current.next;
            }
        }

        return head;
    }
}
```