```
// import java.util.Comparator;
import java.util.Collections;
import java.util.PriorityQueue;
import java.util.Scanner;

public class merge_KSortedList {

    public static Node reverse(Node head) {
        if (head == null || head.link == null) return head;
        Node prev = null, curr = head, next = null;
        while (curr != null) {
            next = curr.link;
            curr.link = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }


    public static Node merge_KLists(Node[] arr, int k) {
        Node final_head = new Node(-123);
        Node end_OfHead = final_head;
        PriorityQueue<Integer> min_heap = new PriorityQueue<>(Collections.
reverseOrder());

        for (int i=0; i<k; i++) {
            Node tmp = arr[i];
            while (tmp != null) {
                min_heap.add(tmp.item);
                tmp = tmp.link;
            }
        }
        while (!min_heap.isEmpty()) {
            end_OfHead = new Node(min_heap.poll());
            end_OfHead = end_OfHead.link;
        }
        Node tmp = final_head;
        final_head = final_head.link;
        tmp.link = null;
        return final_head;
    }


    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int k = input.nextInt();
        Node[] arr = new Node[k];
```

```
    for (var i=0; i<k; i++) {
        System.out.println("Enter size of " + i + " list: ");
        int size = input.nextInt();
        int[] arj = new int[size];
        for (int j=0; j<size; ++j) {
            arj[i] = input.nextInt();
        }
        for (var item:arj) arr[i] = add_item(arr[i], item);
    }
    // for (Node heads:arr) heads = sort_list(heads);
    for (Node heads:arr) print_list(heads);
    Node final_head = merge_KLists(arr, k);
    final_head = reverse(final_head);
    print_list(final_head);
    input.close();
}


// public static Node sort_list(Node head) {
//     if (head == null || head.link == null) return head;
//     PriorityQueue<Node> min_heap = new PriorityQueue<>(
//         new Comparator<Node>(){
//             public int compare(Node a, Node b) {
//                 return a.item-b.item;
//             }
//         });
//     Node tmp = head;
//     while (tmp != null) {
//         min_heap.add(tmp);
//         tmp = tmp.link;
//     }
//     tmp = head;
//     while (tmp != null) {
//         tmp.item = min_heap.poll().item;
//         tmp = tmp.link;
//     }
//     return head;
// }


public static void print_list(Node head) {
    if (head != null) {
        System.out.print( head.item + "  " );
        print_list(head.link);
    }
}


public static Node add_item(Node head, int key) {
```

```
        if (head == null) return new Node(key);
        Node tmp = head;
        while (tmp.link != null) tmp = tmp.link;
        tmp.link = new Node(key);
        return head;
    }


    static class Node {
        int item;
        Node link;
        public Node(int item) {
            this.item = item;
            link = null;
        }
    }
}
```