```
import java.util.PriorityQueue;
import java.util.List;
import java.util.LinkedList;
import java.util.Iterator;
import java.util.Comparator;

/**
 * mergeTwoLinekedList
 */
public class mergeTwoLinekedList {


    public static Node mNode(Node node1, Node node2) {
        Node final_node = null;
        PriorityQueue<Node> min_heap = new PriorityQueue<>(
                new Comparator<Node>() {
                    public int compare(Node a, Node b) {
                        return a.item - b.item;
                    }
                }
        );
        while (node1 != null) {
            min_heap.add(node1);
            node1 = node1.link;
        }
        //min_heap.add(node2);
        while (node2 != null) {
            min_heap.add(node2);
            node2 = node2.link;
        }
        List<Integer> storage = new LinkedList<>();
        Iterator<Node> _it_ = min_heap.iterator();
        while (_it_.hasNext()) {
            storage.add(_it_.next().item);
        }
        Iterator<Integer> it = storage.iterator();
        Node tail = null;
        while (it.hasNext()) {
            if (final_node == null) {
                tail = final_node = new Node(it.next());
            } else {
                tail.link = new Node(it.next());
                tail = tail.link;
            }
        }
        return final_node;
    }
```

```
    public static void main(String[] args) {
        Node head = null;
        for (int i = 0; i < 5; i++) {
            head = add_item(head, i+1);
        }

        print_Llist(head);
        Node head2 = null;
        for (int i = 0; i < 5; i++) {
            head2 = add_item(head2, i + 5);
        }
        print_Llist(head2);
        System.out.println();
        print_Llist(mNode(head, head2));
    }


    static class Node {
        int item;
        Node link;

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


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


    public static Node add_item(Node root, int val) {
        if (root == null) {
            return new Node(val);
        }
        Node tmp = root;
        while (tmp.link != null)
            tmp = tmp.link;
        tmp.link = new Node(val);
        return root;
    }


}
```