public class Practice3 {

    static class Node{

        int data;

        Node next;

        Node(int val){

            this.data=val;

        }

    }

    static class LinkedList{

        Node head=null;

        Node tail=null;

        void Display(){

            Node temp=head;

            while (temp!=null) {

                System.out.print(temp.data+"  ");

                temp=temp.next;

            }

            System.out.println();

        }

        int Size(){

            Node temp=head;

            int count=0;

            while (temp!=null) {

                count++;

            }

            return count;

        }

        void InsertAtBeg(int val){

            //creating new node

            Node temp=new Node(val);

            if(head==null){

                head=temp;

                tail=temp;

            }

            else{

                temp.next=head;

                head=temp;

            }

        }

        void InsertAtEnd(int val){

            Node temp=new Node(val);

            Node t=head;

            if(t==null){

                head=temp;

                tail=temp;

            }

            else{

                tail.next=temp;

                tail=temp;

            }

        }

        void InsertAtIndex(int val,int index){

            Node t=new Node(val);

            Node temp=head;

            if(index==0){

                t.next=head;

                head=t;

                return;

            }

            for(int i=0;i<index-1;i++){

                temp=temp.next;

            }

            t.next=temp.next;

            temp.next=t;

        }

        void GetElement(int idx){

            Node temp=head;

            for(int i=0;i<idx;i++){

                temp=temp.next;

            }

            System.out.println(temp.data);

        }

        void DeleteNode(int idx){

            Node temp=head;

            if(idx==0){

                head=head.next;

                return;

            }

            for(int i=1;i<=idx-1;i++){

                temp=temp.next;

            }

            temp.next=temp.next.next;

        }

        void DeleteNode(Node a){

            a.data=a.next.data;

            a.next=a.next.next;}

        void DeleteNodeLast(Node head,int pos){

            Node fast=head;

            Node slow=head;

            for(int i=1;i<=pos;i++){

                fast=fast.next;

            }

            while (fast.next!=null) {

                fast=fast.next;

                slow=slow.next;

            }

            slow.next=slow.next.next;

        }

        Node FindMiddleNode(Node head){

            Node fast=head;

            Node slow=head;

            while (fast!=null && fast.next!=null) {

                fast=fast.next.next;

                slow=slow.next;

            }

            return slow;

        }

        void DeleteMiddleNode(Node head){

            Node fast=head;

            Node slow=head;

            while (fast.next.next!=null && fast.next.next.next!=null) {

                slow=slow.next;

                fast=fast.next.next;

            }

            slow.next=slow.next.next;

        }

    }

    static Node InterSection(Node head1,Node head2){

            Node temp1=head1;

            Node temp2=head2;

            int count1=0;

            int count2=0;

            while (temp1!=null) {

                count1++;

                temp1=temp1.next;

            }

            while (temp2!=null) {

                count2++;

                temp2=temp2.next;

            }

            temp1=head1;

            temp2=head2;

            if(count1>count2){

                int pos=count1-count2;

                for(int i=1;i<=pos;i++){

                    temp1=temp1.next;

                }

                while (temp1!=temp2) {

                    temp1=temp1.next;

                    temp2=temp2.next;

                }

                return temp1;

            }

            else{

                int pos=count2-count1;

                for(int i=1;i<=pos;i++){

                    temp2=temp2.next;

                }

                while (temp1!=temp2) {

                    temp2=temp2.next;

                    temp1=temp1.next;

                }

                return temp2;

            }

        }

    static void Display(Node head){

            Node temp=head;

            while (temp!=null) {

                System.out.print(temp.data+"  ");

                temp=temp.next;

            }

            System.out.println();

        }

    static Node FindNthLastN(Node head,int pos){

            Node fast=head;

            Node slow=head;

            for(int i=1;i<=pos;i++){

                fast=fast.next;

            }

            while (fast.next!=null) {

                fast=fast.next;

                slow=slow.next;

            }

            return slow;

        }

    static boolean DetectCycle(Node head){

        Node fast=head;

        Node slow=head;

        while (fast!=null) {

            fast=fast.next.next;

            slow=slow.next;

            if(fast==slow){

                return true;

            }

        }

        return false;

    }

    static Node DetectCyclepoint(Node head){

        Node fast=head;

        Node slow=head;

        while(fast!=null){

            slow=slow.next;

            fast=fast.next.next;

            if(fast==slow){

                break;

            }

        }

        Node temp=head;

        while (temp!=slow) {

            slow=slow.next;

            temp=temp.next;

        }

        return slow;

    }

    static Node MergeSortedArray(Node list1,Node list2){

        Node ansList=new Node(12);

        Node head=ansList;

        Node temp1=list1;

        Node temp2=list2;

        while (temp1!=null && temp2!=null) {

            if(temp1.data<temp2.data){

                Node a=new Node(temp1.data);

                head.next=a;

                head=a;

                temp1=temp1.next;

            }else{

                Node a=new Node(temp2.data);

                head.next=a;

                head=a;

                temp2=temp2.next;

            }

            if(temp1==null){

                head.next=temp2;

            }

            if(temp2==null){

                head.next=temp1;

            }

        }

        return ansList.next;

    }

    public static void main(String[] args) {

        LinkedList ll=new LinkedList();

        ll.InsertAtBeg(100);

        ll.InsertAtBeg(90);

        ll.InsertAtBeg(80);

        ll.InsertAtBeg(70);

        ll.InsertAtBeg(60);

        ll.InsertAtBeg(50);

        ll.InsertAtBeg(40);

        Node a=new Node(10);

        Node b=new Node(20);

        Node c=new Node(30);

        Node e=new Node(40);

        Node f=new Node(50);

        Node g=new Node(60);

        Node d=new Node(35);

        a.next=b;

        b.next=c;

        c.next=d;

        d.next=e;

        e.next=f;

        f.next=g;

        Display(a);

        ll.Display();

        Node ans=MergeSortedArray(ll.head,a);

        Display(ans);

    }

}