### Sorted insert for circular linked list

Given a sorted circular linked list, the task is to insert a new node in this circular list so that it remains a sorted circular linked list.

**Example 1:**

**Input:**

LinkedList = 1->2->4

(the first and last node is connected,

i.e. 4 --> 1)

data = 2

**Output:** 1 2 2 4

**Example 2:**

**Input:**

LinkedList = 1->4->7->9

(the first and last node is connected,

i.e. 9 --> 1)

data = 5

**Output:** 1 4 5 7 9

**Expected Time Complexity:** O(N)  
**Expected Auxiliary Space:** O(1)

**Constraints:**  
0 <= N <= 105

**Company Tags**

[**Zoho**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Zoho) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft)

//{ Driver Code Starts

import java.util.\*;

class Node{

int data;

Node next;

Node(int d){

data=d;

next=null;

}

}

class LL{

Node head;

Node tail;

public void addToTheLast(Node node) {

if (head == null) {

head = node;

tail = head;

} else {

tail.next = node;

tail = tail.next;

}

}

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while (t--> 0) {

int n = sc.nextInt();

LL llist = new LL();

llist.head=null;

int a1 = sc.nextInt();

Node head = new Node(a1);

llist.addToTheLast(head);

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

int a = sc.nextInt();

llist.addToTheLast(new Node(a));

}

int value=sc.nextInt();

Node phead=head;

while(phead.next!=null){

phead=phead.next;

}

phead.next=head;

Solution ob=new Solution();

head = ob.sortedInsert(head,value);

Node current = head;

if(head != null){

do{

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

current = current.next;

}while(current != head);

}

System.out.println();

}

}

}

// } Driver Code Ends

class Solution

{

public static Node sortedInsert(Node head,int data)

{

if(head.data>=data){

Node temp=new Node(data);

temp.next=head;

Node curr=head;

while(curr.next!=head){

curr=curr.next;

}

curr.next=temp;

return temp;

}

Node root=head;

Node ptr=null;

while(head.data<data ){

ptr=head;

head=head.next;

}

ptr.next=new Node(data);

ptr=ptr.next;

ptr.next=head;

return root;

}

}