**Competitive Coding Lab 3(Linked List)**

**Student Name: Sahul Kr. Parida UID: 20BCS4919**

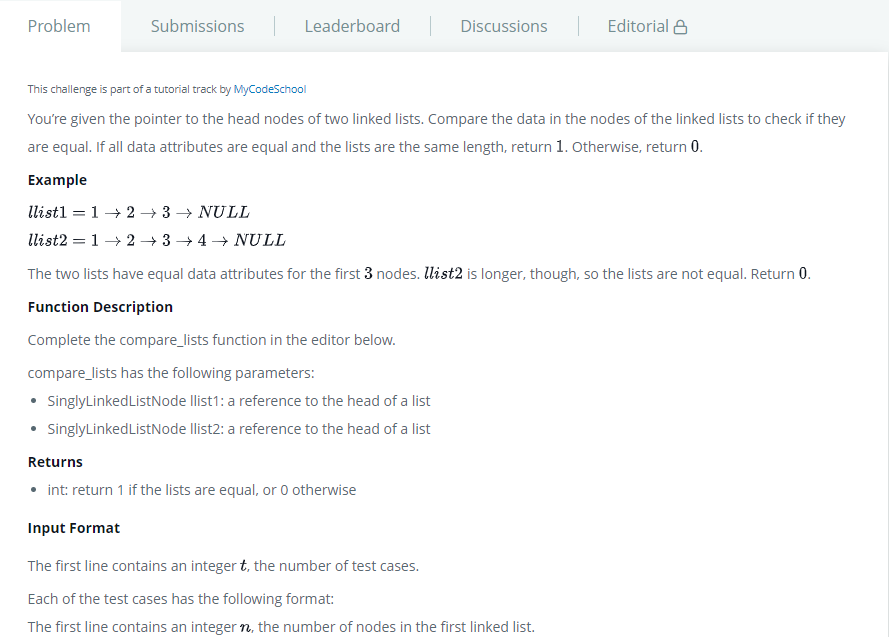
**Branch: CSE Section/Group: WM-904/B**

**Semester: 5th Date of Performance: 18/08/22**

**Subject Name: Competitive Coding(CC) Subject Code: 20CSP-314**

**PROBLEM STATEMENT 3.1: -**

[**https://www.hackerrank.com/challenges/compare-two-linked-lists/problem?isFullScreen=false**](https://www.hackerrank.com/challenges/compare-two-linked-lists/problem?isFullScreen=false)



**SOLUTION:**

import java.io.\*;

import java.math.\*;

import java.text.\*;

import java.util.\*;

import java.util.regex.\*;

/\* 20BCS4919\_Sahul Kumar Parida \*/

public class Solution {

    static class SinglyLinkedListNode {

        public int data;

        public SinglyLinkedListNode next;

        public SinglyLinkedListNode(int nodeData) {

            this.data = nodeData;

            this.next = null;

        }

    }

    static class SinglyLinkedList {

        public SinglyLinkedListNode head;

        public SinglyLinkedListNode tail;

        public SinglyLinkedList() {

            this.head = null;

            this.tail = null;

        }

        public void insertNode(int nodeData) {

            SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);

            if (this.head == null) {

                this.head = node;

            } else {

                this.tail.next = node;

            }

            this.tail = node;

        }

    }

    public static void printSinglyLinkedList(SinglyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

        while (node != null) {

            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {

                bufferedWriter.write(sep);

            }

        }

    }

    static boolean compareLists(SinglyLinkedListNode head1, SinglyLinkedListNode head2) {

        while((head1!=null && head2!=null) && head1.data == head2.data){

            head1 = head1.next;

            head2 = head2.next;

        }

        return (head1 == null && head2 == null);

    }

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) throws IOException {

        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

        int tests = scanner.nextInt();

        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int testsItr = 0; testsItr < tests; testsItr++) {

            SinglyLinkedList llist1 = new SinglyLinkedList();

            int llist1Count = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llist1Count; i++) {

                int llist1Item = scanner.nextInt();

                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist1.insertNode(llist1Item);

            }

            SinglyLinkedList llist2 = new SinglyLinkedList();

            int llist2Count = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llist2Count; i++) {

                int llist2Item = scanner.nextInt();

                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist2.insertNode(llist2Item);

            }

            boolean result = compareLists(llist1.head, llist2.head);

            bufferedWriter.write(String.valueOf(result ? 1 : 0));

            bufferedWriter.newLine();

        }

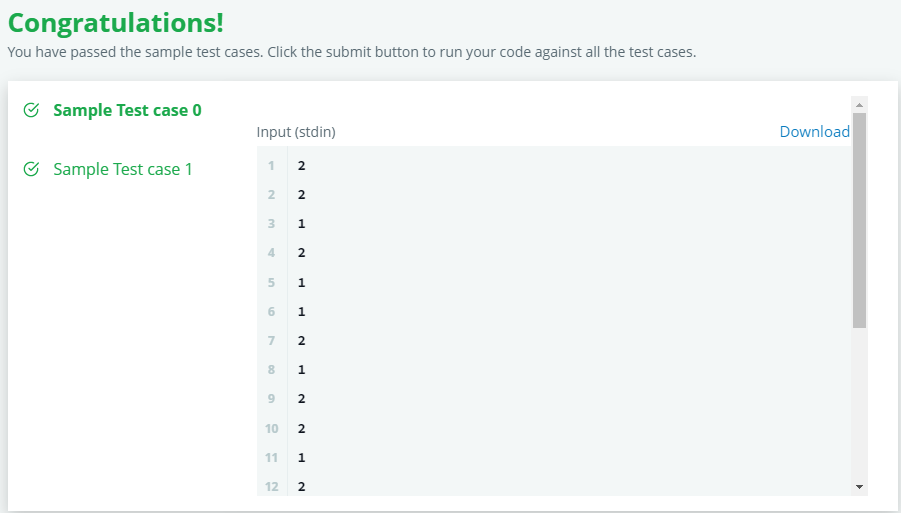
        bufferedWriter.close();

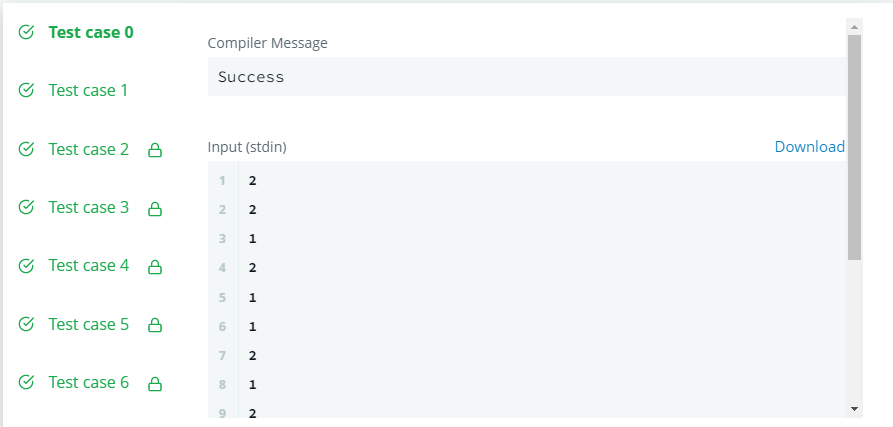
        scanner.close();

    }

}

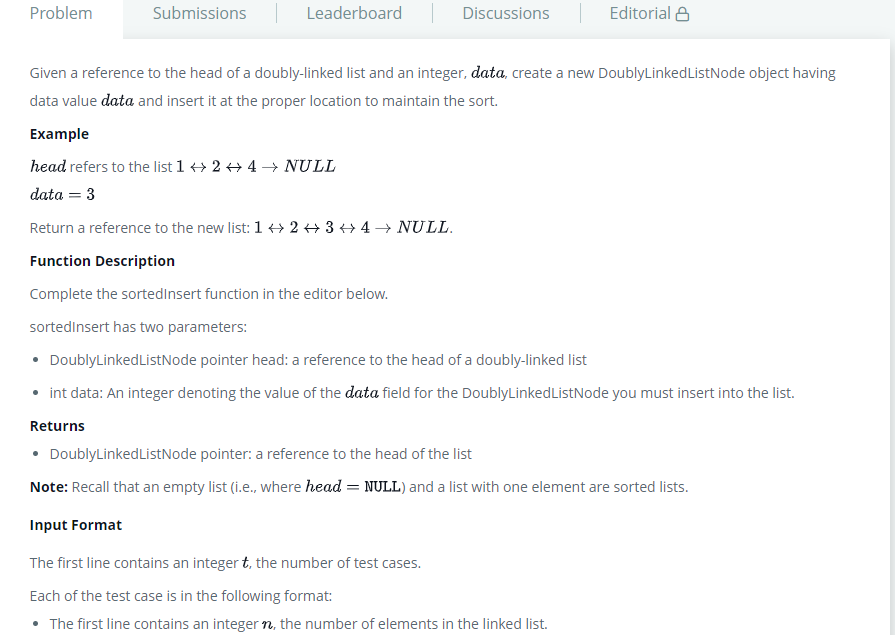
# TEST CASES:





**PROBLEM STATEMENT 3.2: -**

<https://www.hackerrank.com/challenges/insert-a-node-into-a-sorted-doubly-linked-list/problem?isFullScreen=true>



# SOLUTION:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

/\* 20BCS4919\_Sahul Kumar Parida \*/

public class Solution {

    static class DoublyLinkedListNode {

        public int data;

        public DoublyLinkedListNode next;

        public DoublyLinkedListNode prev;

        public DoublyLinkedListNode(int nodeData) {

            this.data = nodeData;

            this.next = null;

            this.prev = null;

        }

    }

    static class DoublyLinkedList {

        public DoublyLinkedListNode head;

        public DoublyLinkedListNode tail;

        public DoublyLinkedList() {

            this.head = null;

            this.tail = null;

        }

        public void insertNode(int nodeData) {

            DoublyLinkedListNode node = new DoublyLinkedListNode(nodeData);

            if (this.head == null) {

                this.head = node;

            } else {

                this.tail.next = node;

                node.prev = this.tail;

            }

            this.tail = node;

        }

    }

    public static void printDoublyLinkedList(DoublyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

        while (node != null) {

            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {

                bufferedWriter.write(sep);

            }

        }

    }

    static DoublyLinkedListNode sortedInsert(DoublyLinkedListNode head, int data) {

        DoublyLinkedListNode node = new DoublyLinkedListNode(data);

        if (head == null) {

            return node;

        } else if (data <= head.data) {

            node.next = head;

            head.prev = node;

            return node;

        } else {

            DoublyLinkedListNode ptr = sortedInsert(head.next, data);

            head.next = ptr;

            ptr.prev = head;

            return head;

        }

    }

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) throws IOException {

        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

        int t = scanner.nextInt();

        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int tItr = 0; tItr < t; tItr++) {

            DoublyLinkedList llist = new DoublyLinkedList();

            int llistCount = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llistCount; i++) {

                int llistItem = scanner.nextInt();

                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist.insertNode(llistItem);

            }

            int data = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            DoublyLinkedListNode llist1 = sortedInsert(llist.head, data);

            printDoublyLinkedList(llist1, " ", bufferedWriter);

            bufferedWriter.newLine();

        }

        bufferedWriter.close();

        scanner.close();

    }

}

# TEST CASES:

# 

# 