Task 1: Comparing two Linked List

Problem

You're given the pointer to the head nodes of two linked lists. Compare the data in the nodes of the linked lists to check if they are equal. The lists are equal only if they have the same number of nodes and corresponding nodes contain the same data. Either head pointer given may be null meaning that the corresponding list is empty. You have to complete the `int CompareLists(Node* headA, Node* headB)` method which takes two arguments - the heads of the two linked lists to compare.

Input

The first line contains **t**, the number of test cases. The format for each test case is as follows:
The first line contains an integer **n**, denoting the number of elements in the first linked list.
The next **n** lines contain an integer each, denoting the elements of the first linked list.
The next line contains an integer **m**, denoting the number of elements in the second linked list.
The next **m** lines contain an integer each, denoting the elements of the second linked list.

Output

Compare the two linked lists and return 1 if the lists are equal. Otherwise, return 0.

Sample Test

Input	Output
2 2 1 2 1 1 2 1 2 2 1 2	0 1

Task 2: Merge two sorted linked lists

Problem

You're given the pointer to the head nodes of two sorted linked lists. The data in both lists will be sorted in ascending order. Change the next pointers to obtain a single, merged linked list which also has data in ascending order. Either head pointer given may be null meaning that the corresponding list is empty.
You have to complete the Node MergeLists(Node headA, Node headB) method which takes two arguments - the heads of the two sorted linked lists to merge.

Input

The first line contains **t**, the number of test cases. The format for each test case is as follows:
The first line contains an integer **n**, denoting the number of elements in the first linked list. The next **n** lines contain an integer each, denoting the elements of the first linked list. The next line contains an integer **m**, denoting the number of elements in the second linked list. The next **m** lines contain an integer each, denoting the elements of the second linked list.

Output

Change the next pointer of individual nodes so that nodes from both lists are merged into a single list. Then return the head of this merged list.

Sample Test

Input	**Output**
1 3 1 2 3 2 3 4	1 2 3 3 4

<u>Task 3: Union of Doubly Linked Lists</u>

Problem:

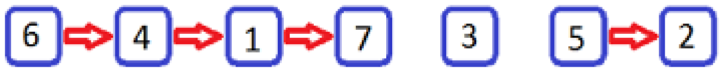
Doubly linked list is one of the fundamental data structures. A doubly linked list is a sequence of elements, each containing information about the previous and the next elements of the list. In this problem all lists have linear structure. I.e. each element except the first has exactly one previous element, each element except the last has exactly one next element. The list is not closed in a cycle.

In this problem you are given *n* memory cells forming one or more doubly linked lists. Each cell contains information about element from some list. Memory cells are numbered from 1 to *n*.

For each cell *i* you are given two values:

- l_i — cell containing previous element for the element in the cell *i*;
- r_i — cell containing next element for the element in the cell *i*.

If cell *i* contains information about the element which has no previous element then $l_i = 0$. Similarly, if cell *i* contains information about the element which has no next element then $r_i = 0$.



Three lists are shown on the picture.

For example, for the picture above the values of *l* and *r* are the following: $l_1 = 4$, $r_1 = 7$; $l_2 = 5$, $r_2 = 0$; $l_3 = 0$, $r_3 = 0$; $l_4 = 6$, $r_4 = 1$; $l_5 = 0$, $r_5 = 2$; $l_6 = 0$, $r_6 = 4$; $l_7 = 1$, $r_7 = 0$.

Your task is to unite all given lists in a single list, joining them to each other in any order. In particular, if the input data already contains a single list, then there is no need to perform any actions. Print the resulting list in the form of values l_i, r_i.

Any other action, other than joining the beginning of one list to the end of another, cannot be performed.

Input

The first line contains a single integer *n* ($1 \le n \le 100$) — the number of memory cells where the doubly linked lists are located.

Each of the following *n* lines contains two integers l_i, r_i ($0 \le l_i, r_i \le n$) — the cells of the previous and the next element of list for cell *i*. Value $l_i = 0$ if element in cell *i* has no previous element in its list. Value $r_i = 0$ if element in cell *i* has no next element in its list.

It is guaranteed that the input contains the correct description of a single or more doubly linked lists. All lists have linear structure: each element of list except the first has exactly one previous element; each element of list except the last has exactly one next element. Each memory cell contains information about one element from some list, each element of each list written in one of *n* given cells.

Output

Print *n* lines, the *i*-th line must contain two integers l_i and r_i — the cells of the previous and the next element of list for cell *i* after all lists from the input are united in a single list. If there are many solutions print any of them.

Sample Test

Input	**Output**
`7`	`4 7`
`4 7`	`5 6`
`5 0`	`0 5`
`0 0`	`6 1`
`6 1`	`3 2`
`0 2`	`2 4`
`0 4`	`1 0`
`1 0`	