Road net

A diskette was enclosed to a road map. The diskette contains the table of the shortest ways (distances) between each pair of towns on the map. All the roads are two-way. The location of towns on the map has the following interesting property: *if the length of the shortest way from town A to town B equals the sum of the lengths of the shortest ways from A to C and C to B then town C lies on (certain) shortest way from A to B*. We say that towns A and B are neighbouring towns if there is no town C such that the length of the shortest way from A to B equals the sum of the lengths of the shortest ways from A to C and C to B. Find all the pairs of neighbouring towns.

Example

For the table of distances:

	A	**B**	**C**
A	0	1	2
B	1	0	3
C	2	3	0

the neighbouring towns are A, B and A, C.

Task

Write a program that for each test case:

- reads the table of distances from standard input;
- finds all the pairs of neighbouring towns;
- writes the result to standard output.

Input

The number of test cases *t* is in the first line of input, then *t* test cases follow separated by an empty line.

In the first line of each test case there is an integer *n*, 1 <= *n* <= 200, which equals the number of towns on the map. Towns are numbered from 1 to *n*.

The table of distances is written in the following *n* lines. In the (*i*+1)-th line, 1 <= *i* <= *n*, there are *n* non-negative integers not greater than 200, separated by single spaces. The *j*-th integer is the distance between towns *i* and *j*.

Output

For each test case your program should write all the pairs of the neighbouring towns (i.e. their numbers). There should be one pair in each line. Each pair can appear only once. The numbers in each pair should be given in increasing order. Pairs should be ordered so that if the pair (*a*, *b*) precedes the pair (*c*, *d*) then *a* < *c* or (*a* = *c* and *b* < *d*).

Consequent test cases should by separated by an empty line.

Example

Sample input:

```
1
3
0 1 2
1 0 3
2 3 0
```

Sample output:

```
1 2
1 3
```