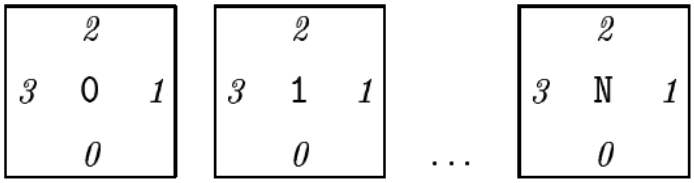
2390 Robotic Jigsaw

Your task here is to write a program to aid in the assembly of a jigsaw puzzle by a robot.

The puzzle is a rectangular array of flat four-sided pieces. Except for the outer borders, each side of each piece is shaped in such a way that it matches one side of exactly one other piece.

When the puzzle is presented to the robot, all the pieces are scrambled and spread out on a table, face up, in front ofthe robot's camera eye. Let's assume that someone already wrote software that does the image processing part of the problem. tind the puzzle pieces in the image captured by the camera, break the outline of each piece into its four sides, and tind pairs of sides that have the same shape. Your task is to program the next step — namely, decide where in the grid each piece should be placed, and how it should be rotated.

The image-processing software assigns, to each piece that the robot sees on the table, a unique *piece number* between 0 and $N-1$, in some arbitrary order. It also labels the sides of the piece with integers 0, 1, 2, 3, in counterclockwise order, starting from an arbitrary side.



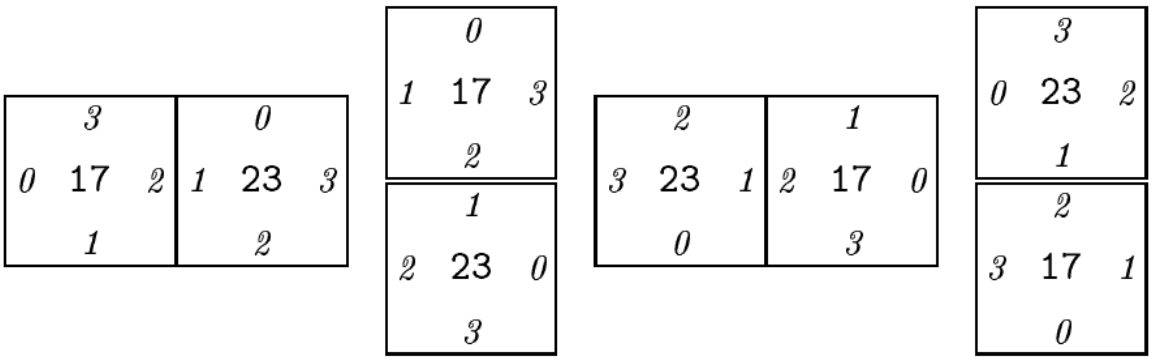
Input

The input file contains several consecutive instances of the problem. Each instance consists of: one line contaning two integers N, $0 < N < 500$ (the number of pieces) and K; then come K lines, one for each pair of similar sides that were found among all pieces seen by the robot, in arbitrary order.

Each of these lines contains four integers like this:

```
17 2   23 1
```

This line says that side number 2 of piece number 17 fits side 1 of piece 23. Therefore, in the assembled puzzle, pieces 17 and 23 should be neighbors, in one of these four positions.



You can assume that the input data includes enough information to assemble the puzzle completely. Note, however, that it may not include all pairs of matching sides, and it may include some redundant pairs that could be deduced from the previous ones. The end of the input is marked by a line containing a single '0'.

Output

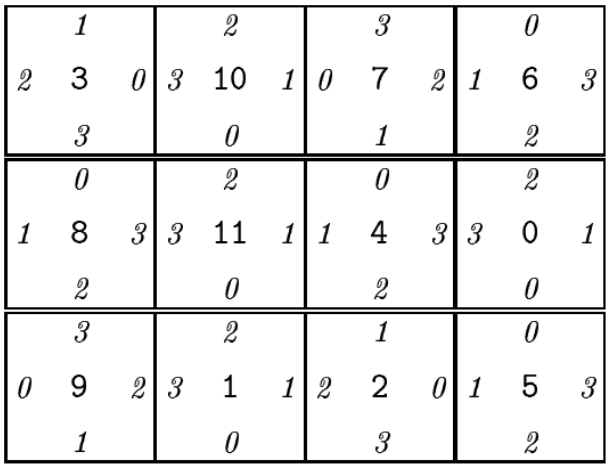
For the i-th instance of the input file you should write the line '`Instance` i`:`', followed by a sequence of N lines, one for each piece, containing four integers such as '`6 9 17 2`'. This line says that piece number 17 should be placed in row 6 and column 9 of the grid, turned so that its side number 2 lies at the bottom. (By convention, rows are numbered from top to bottom, and columns from left to right, starting with 0.)

Note also that any solution can be turned by multiples of 90 degrees to give three other valid solutions. In order to make the output unique, you must turn your solution around so that piece number 0 has its side number 0 at the bottom (turned towards the robot). In any case, the upper left corner of the assembled puzzle should be at position [0, 0].

The lines of the output should be presented in lexicographic order of the rows and columns, as shown in the sample output and using the same format.

Assume that the puzzle is completely filled by the pieces.

Note: The figure on right represents the solution below.



Sample Input

```
12   13
 0 0   5 0
 0 2   6 2
 1 2  11 0
 1 3   9 2
 2 0   5 1
 2 1   4 2
 3 0  10 3
 3 3   8 0
 4 0   7 1
 4 1  11 1
 6 1   7 2
 8 2   9 3
10 0  11 2
0
```

Sample Output

```
Instance 1:
0   0    3 3
0   1   10 0
0   2    7 1
0   3    6 2
1   0    8 2
1   1   11 0
1   2    4 2
1   3    0 0
2   0    9 1
2   1    1 0
2   2    2 3
2   3    5 2
```