Problem F. Mark on a Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a graph with n vertices and m edges: the graph is undirected and has no self-loops and no multiple edges. You know for a fact that the graph was obtained in one of the two ways.

- The graph is randomly generated: the process starts with a graph with no edges, and then, m times, a random uniformly chosen non-existent edge is added to it.

  In this case, leave a mark on the graph. For that, you can do the following operation from 0 to 5 times: pick a pair of vertices and change the state of the edge between them, adding it if it was not present or removing it otherwise.

- The graph contains a mark, in other words, it is obtained from a random graph by the procedure described above. But after that procedure, the vertices are renumerated randomly, and the edges are given in random order as well. The two vertices of each edge can also be given in any ordder.

  In this case, nothing more has to be done.

Interaction Protocol

In this problem, your solution will be run twice on each test. In input and output, numbers on a single line are separated by spaces. Each line of input is terminated by an end-of-line character.

During each run, the solution gets a graph as input. The first line contains two integers n and m: the number of vertices and edges in the graph. Each of the following m lines contains two integers u and v denoting an edge between vertices u and v in the graph ($1 \le u, v \le n$, $u \ne v$, the bidirectional edges are all distinct).

First Run

During the first run, the given graph is randomly generated in advance according to the problem statement ($n = 1000$, $2000 \le m \le 5000$). On the first line, print the word "`mark`", and on the second line, print the number k of operations with edges ($0 \le k \le 5$). Each of the following k lines must contain two integers u and v denoting the change of state of the edge between vertices u and v ($1 \le u, v \le n$, $u \ne v$).

Second Run

During the second run, the given graph is the one obtained after the first run. However, the vertices are renumerated randomly, the edges are given in random order, and the vertices of each edge are also given in random order. All the shuffles are fixed in advance in each test, so, if solutions make the same choices in the first run, they will get the same inputs for the second run. In this case, print the word "`ok`" on the first line.

Example

For each test, the input during the second run depends on the solution's output during the first run.

Below we show two runs of a certain solution on the first test. The graphs are shown only partially for brevity. The full version of the example can be seen in `samples.zip`.

standard input	*standard output*
1000 3560 603 151 415 20 102 569 895 552 <...> 224 267 651 506	mark 3 763 968 572 286 453 139

standard input	*standard output*
1000 3561 192 768 693 994 786 238 351 329 <...> 100 66 54 819	ok