Problem C. Cactus Construction

Input file: `cactus.in`
Output file: `cactus.out`

Let us consider the following way of constructing graphs. Pick the number of colors $\hat{c}$. Let n be the number of vertices in a graph. To build a graph, we use a workspace with several graphs in it. Each vertex of each graph has a color. Colors are denoted by integers from 1 to $\hat{c}$. Initially, we have n graphs in a workspace with one vertex in each graph, all colored with color 1, and no edges. The only vertex of i-th graph has number i.

The following operations are permitted:

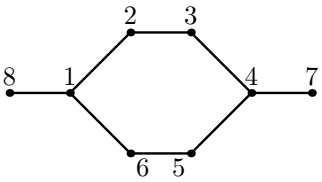
- *join* a b: join graphs containing vertices a and b into one graph. No edges are added. Vertices a and b must be in different graphs.
- *recolor* a c_1 c_2: in graph containing vertex a recolor all vertices of color c_1 with color c_2.
- *connect* a c_1 c_2: in graph containing vertex a create edges between all pairs of vertices where one vertex has color c_1 and the other has color c_2. If $c_1 = c_2$ loops are not created. If such an edge already exists, then the second parallel edge *is* created. Multi-edges are not allowed in this problem, so this case must not occur.

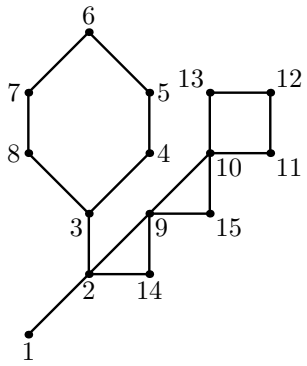
At the end we should have a single graph and colors of its vertices do not matter.

The minimal number of colors $\hat{c}$, that can be used to construct a graph, is called a *clique width* of a graph. Clique width is one of the characteristics of graph complexity. Many NP-hard problems can be solved in polynomial time on graphs with bounded clique width, using dynamic programming on this process of building a graph. For a general graph, calculating the exact value of a clique width is known to be NP-hard. However, for some graph classes bounds on a clique width are known.

Cactus is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively cactus is a generalization of a tree where some cycles are allowed. Multi-edges (multiple edges between a pair of vertices) and loops (edges that connect a vertex to itself) are not allowed in a cactus. It is known that a clique width of a cactus does not exceed 4.

You are given a cactus. Find out how to build it in the described way using at most $\hat{c} = 4$ colors.





Input

The first line of the input file contains two integers n and m ($1 \le n \le 50\,000$; $0 \le m \le 50\,000$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths.

Each of the following m lines contains a path in the graph. A path starts with an integer k_i ($2 \le k_i \le 1000$) followed by k_i integers from 1 to n. These k_i integers represent vertices of a path. Adjacent vertices in the path are distinct. The path can go to the same vertex multiple times, but every edge is traversed exactly once in the whole input file.

The graph in the input file is a cactus.

Output

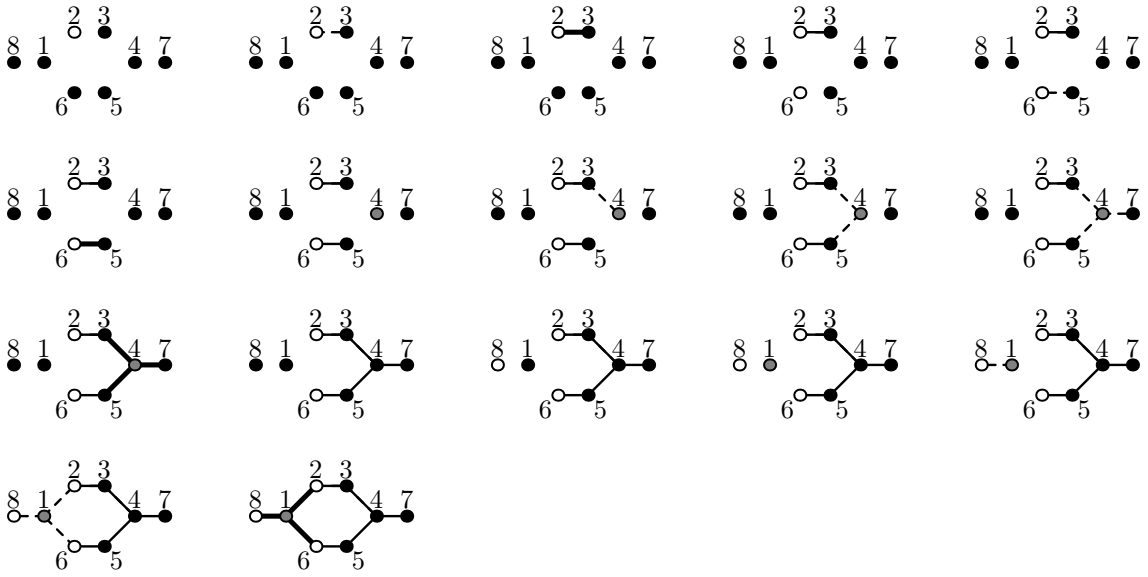
In the first line print one integer q — the number of operations you need. This number should not be greater than 10^6. In the next q lines print operations. Each operation is denoted by its first letter ("`j`" for *join*, "`r`" for *recolor* and "`c`" for *connect*) and its arguments in the order they are described in the problem statement.

At the end, after applying all these operations, one should have one graph, which is equal to the cactus in the input. This means that there should be exactly one edge between each pair of vertices connected in the input graph, and no edges between vertices not connected in the input graph.

Examples

cactus.in	cactus.out
8 2 5 1 2 3 4 7 5 4 5 6 1 8	17 r 2 1 2 j 2 3 c 2 1 2 r 6 1 2 j 5 6 c 6 1 2 r 4 1 3 j 4 3 j 4 6 j 4 7 c 4 3 1 r 4 3 1 r 8 1 2 r 1 1 3 j 1 8 j 1 4 c 1 3 2

The following picture visualizes the sequence 17 operations from the first sample output. If an edge is not created yet, but its vertices are already in one graph, then this edge is drawn as dashed.



cactus.in	cactus.out
15 3 9 1 2 3 4 5 6 7 8 3 7 2 9 10 11 12 13 10 5 2 14 9 15 10	39 r 7 1 2 r 5 1 2 j 7 8 c 7 1 2 j 5 4 c 5 1 2 r 6 1 3 j 6 7 j 6 5 c 6 3 2 r 3 1 4 j 6 3 c 6 4 1 r 11 1 2 r 13 1 2 j 12 11 j 12 13 c 11 1 2 r 10 1 3 j 12 10 c 10 2 3 r 10 1 2 r 10 4 2 r 15 1 3 j 15 10 c 15 3 3 j 9 10 c 9 1 3 r 9 3 2 r 9 1 4 r 14 1 4 j 9 14 c 9 4 4 r 1 1 4 r 3 1 2 j 2 1 j 2 14 j 2 3 c 2 1 4