A Link to the Past Randomizer

One of Florian's favourite games is *The Legend of Zelda: A Link to the Past.* Lately he discovered that some hackers programmed a randomizer for the game. This is a program that randomly mixes up all the items in the game. For example, so that you don't start with the sword at the beginning, but with the bow. Finally he can re-explore the fantastic world of Hyrule and relive this adventure. In hindsight, he begins to think about how this is technically done. The website claims that the generated ROMs are always solveable. It therefore cannot happen that you get stuck at one point in the game because of an item that you absolutely need, but that can only be found in a chest later in the game. We want to deal with this problem in this task:

In our simplified Legend of Zelda version, there are n different items that can be found in n chests scattered around the game. They are numbered from 1 to n. At the beginning you've got no items and you win as soon as you obtain all of them. For each item you know what other items are needed to obtain it. There can be several ways to get an item.

As an example, assume there are two ways to obtain the *bomb* item:

- With the grappling hook.
- With the boomerang and the pegasus boots.

This information is given to you in the form of m conditions. If there is no condition given for a specific item, then you can obtain it without having to obtain other items beforehand.

Can you determine if a given configuration is winnable? If so, find an order to unlock the items.

Input

The first line contains n and m, the number of items and the number of conditions. The following m lines are the conditions. Every condition has the form $i\ k\ a_1 \ldots a_k$.
i is the number of the item that you want to obtain.
$a_1 \ldots a_k$ are the k items that are needed before you are able to obtain the item i.

Output

If there is no valid configuration, print -1 to the console. Otherwise print a permutation to the console, that describes the order in which you have to unlock the items. If there are multiple solutions, you can print any of them.

Examples

Input	Output	Comments

`2 2`
`1 1 2`
`2 1 1`
	`-1`	Two items, but each can only be unlocked with the other.

Input	Output	Comments

`3 3`
`1 1 2`
`1 1 3`
`2 1 1`
	`3 1 2`	Like the previous input, but with a third item. Alternatively, item 1 can also be unlocked with item 3. Since there are no conditions for item 3, it can be picked up right at the beginning.

Input	Output	Comments

`5 4`
`2 1 1`
`3 1 2`
`4 1 3`
`5 4 1 2 3 4`
	`1 2 3 4 5`	Item 1 is available from the start. To obtain the items 2 to 4 you need the item with the number that is one smaller than the item you want to unlock. To unlock item 5 all other items are necessary.

Scoring

In general:

- $1 \leq n, m \leq 10^6$
- For each condition it holds that:
  - $1 \leq i \leq n$
  - $1 \leq k \leq n$
  - All a_j are different
- The sum of the k's of all conditions is less or equal to 10^6

Subtask 1 (25 Points): $n \leq 1000$ and for every item there is at most one condition.

Subtask 2 (25 Points): For every item there is at most one condition.

Subtask 3 (10 Points): $n \leq 10$

Subtask 4 (15 Points): $n \leq 1000$

Subtask 5 (25 Points): No additional constraints

Constraints

Time limit: 1 s **Memory limit:** 256 MB