Problem Instructions:

1. This is a timed exercise; however, it is on the honor system. We expect you to spend no more than two hours total over the next 48 hours to complete it. No need to rush. We don't consider how long it takes you to complete the problem so please take the time to check your work thoroughly before submission.

2. If you do not finish within the time allotted, please submit what you have completed and provide an explanation in your email of what work remains to be done. If you have problems with your development environment, please notify us immediately via email by responding to this message.

.

3. We will accept solutions in Java, Python, C++ or C. Please use only standard libraries and write your code to be portable. You are free to use your favorite IDE, editor, and reference documentation.

4. Your solution will be judged primarily for its correctness. However, we also place significant emphasis on code clarity, design and efficiency.

5. If you have questions about the problem, please use your best judgment and carefully document your assumptions in the code.

6. When you have your final solution, please reply to this email with your solution as a .zip, .tar, or .tar.gz attachment. Do not include any binaries, only source code.

Problem Title: Project File Dependencies

Project managers, such as the UNIX utility make, are used to maintain large software projects made up from many components. Users write a *project file* specifying which components (called *tasks*) depend on others and the project manager can automatically update the components in the correct order.

Write a program that reads a project file from stdin and writes to stdout the order in which the tasks should be performed.

Input

For simplicity we represent each task by an integer number from $1,2,3\ldots N$ where N is the total number of tasks. The first line of input specifies the number N of tasks and the number M of rules, such that $N \leq 100$ and $M \leq 100$.
The rest of the input consists of M *rules*, one in each line, specifying dependencies using the following syntax:

$$T_0\, k\, T_1 T_2 T_3 \ldots T_k$$

This rule means that task number T_0 depends on k tasks $T_1 \ldots T_k$ (we say that task T_0 is the *target* and $T_1 \ldots T_k$ are *dependents*).

Note that tasks numbers are separated by single spaces and that rules end with a newline. Rules can appear in any order, but each task can appear as target only once.
Your program can assume that there are no circular dependencies in the rules, i.e. no task depends

directly or indirectly on itself.

Output

The output should be a single line with the permutation of the tasks $1,2...N$ to be performed, ordered by dependencies (i.e. no task should appear before others that it depends on).

To avoid ambiguity in the output, tasks that do not depend on each other should be ordered by their number (lower numbers first).

Example

Input:
5 4
3 2 1 5
2 2 5 3
4 1 3
5 1 1

Output:
1 5 3 2 4