2432 - Project File Dependencies

Europe - Southwestern - 2001/2002

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 and outputs the order in which the tasks should be performed.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

For simplicity we represent each task by an integer number from $1,2,...,N$ (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 $N \le 100$, $M \le 100$.

The rest of the input consists of M *rules*, one in each line, specifying dependencies using the following syntax:

$T_0 \quad k \quad T_1 \quad T_2 \quad ... \quad T_k$

This rule means that task number T_0 depends on k tasks $T_1, T_2, ..., T_k$ (we say that task T_0 is the *target* and $T_1,...,T_k$ 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

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output should be a single line with the permutation of the tasks $1 ... N$ 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).

Sample Input

```
1

5 4
```

```
3 2 1 5
2 2 5 3
4 1 3
5 1 1
```

Sample Output

```
1 5 3 2 4
```

Southwestern 2001-2002