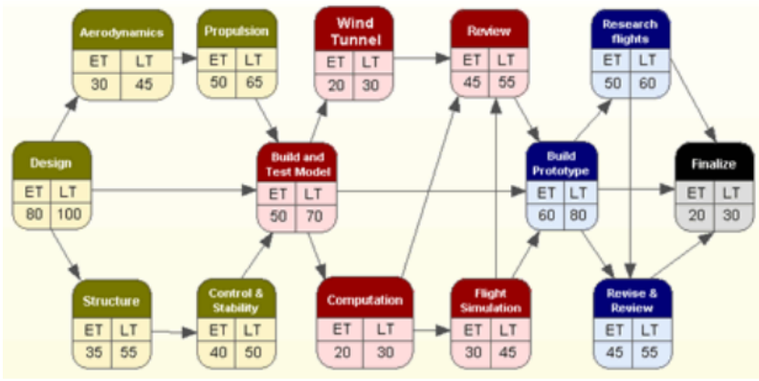


The job market is not holding a good prospect for Computer Science graduates lately. As a result a lot of CS graduates here are either going for MBA or looking for a job, which is not quite related to their field of study. Turza, a fresh CS graduate is one such example. He with his sound knowledge of CS could not find a single job where he could make use of what he knows. He ends up getting a job of project supervision, which is basically keeping track if the parts of the project are being completed according to the schedule. What a waste of four years' hard work!

A guy has to earn his living and we all understand that. But there should also be some sort of satisfaction in the job. That's what Turza is trying to find now. He is desperately searching for something interesting in his work. But the only interesting thing for him is mathematical problems. Fortunately for Turza, he has been able to create and solve one math problem from his not so interesting work. We would like to leave it for you to see if you can do it too. He says:

> "The project I supervise has many parts. Some of the parts are dependent on other parts. So if part X is dependent on part Y and Z, we'd have to make sure that both part Y and Z is finished before we start working on part X. **For some strange reason which I really don't have any clue to, here we must finish all the parts that are currently not dependent on any other parts. Once all these parts are completed, we proceed on to completing the other parts following the same rule**. It is fairly easy to calculate how many ways are there that this entire project can be completed. However, I am more interested in the solution of the inverse problem. Given the number of parts we have in a project and the number of ways that the project can be completed (if we follow the procedure I just explained) can you find the dependency list of the parts in the project? It is quite obvious that the answer need not be unique, so any valid answers would do."

If we want to calculate the number of ways to finish a project the number can get quite big, even if the number of parts in the project is not so large. So here we prefer to use the prime factorized form for the count. You can safely assume that for every count and number of parts we give you, there is at least one project that they correspond to. Your task is simply to find one such project and list the dependency among the parts.

Input

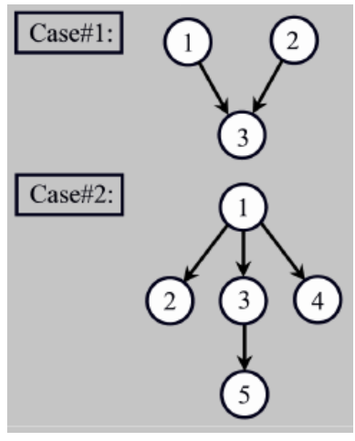
The first line of the input gives you the number of test cases, T ($T \leq 20$). Then you would have the description of T test cases. Every test case starts with two integers in one line. The first integer N ($1 \leq N \leq 50$) is the total number of parts in the project. The next integer is the number of primes, P ($P \leq N$) in the prime factorization of the count (the number of ways to complete the project). In the next P lines P pairs are listed. The first number in each pair is the prime number and the second number is the number of times it occurs in the count.

Please note that we only list a prime number if it occurs nonzero times in the count.

Output

For each of the test cases you have to print the serial number of the test case in the format '`Case#`x`:`' where x is the serial number starting from 1. In the next N lines you have to list the dependency list for each of the projects. The i-th dependency list starts with an integer which gives us the number of parts that depenedent on the i-th part. Then we would have the labels of that many parts on the same line. These labels are integers in the range 1 to N and should be separated by a single space. There is a special judge for this problem which will take your project as input and check if we can indeed reach that count. Please note that your project description must not be such as to mean that it is impossible to finish the project. And you must not mention a dependency more than once.

Illustration:



Sample Input

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

Sample Output

```
Case#1:
1 3
1 3
0
Case#2:
3 2 3 4
0
1 5
0
0
```