There is an ancient saying that “All Roads Lead to Rome”. If this were true, then there is a simple algorithm for finding a path between any two cities. To go from city A to city B, a traveller could take a road from A to Rome, then from Rome to B. Of course, a shorter route may exist.

The network of roads in the Roman Empire had a simple structure: beginning at Rome, a number of roads extended to the nearby cities. From these cities, more roads extended to the next further cities, and so on. Thus, the cities could be thought of as existing in *levels* around Rome, with cities in the ith level only connected to cities in the $i-1$th and $i+1$th levels (Rome was considered to be at level 0). No loops existed in the road network. Any city in level i was connected to a single city in level $i-1$, but was connected to zero or more cities in level $i+1$. Thus, to get to Rome from a given city in level i, a traveller could simply walk along the single road leading to the connected $i-1$ level city, and repeat this process, with each step getting closer to Rome.

Given a network of roads and cities, your task is to find the shortest route between any two given cities, where distance is measured in the number of intervening cities.

Input

The first line is the number of test cases, followed by a blank line.

The first line of each test case of the input contains two numbers in decimal notation separated by a single space. The first number (m) is the number of roads in the road network to be considered. The second number (n) represents the number of queries to follow later in the file.

For each test case, in the next m lines in the input each contain the names of a pair of cities separated by a single space. A city name consists of one or more letters, the first of which is in uppercase. No two cities begin with the same letter. The name `Rome` always appears at least once in this section of input, for each test case; this city is considered to be at level 0, the lowest-numbered level. The pairs of names indicate that a road connects the two named cities. The first city named on a line exists in a lower level than the second named city. The road structure obeys the rules described above. For each test case, no two lines of input in this section are repeated.

The next n lines, for each test case in the input each contain the names of a pair of cities separated by a single space. City names are as described above. These pairs of cities are the *query pairs*. Your task for each query pair is to find the shortest route from the first named city to the second. Each of the cities in a query pair is guaranteed to have appeared somewhere in the previous input section, for each test case, describing the road structure.

Each test case will be separated by a single line.

Output

In each test case, for each of the n query pairs, output a sequence of uppercase letters indicating the shortest route between the two query pair cities. The sequence must be output as consecutive letters, without intervening whitespace, on a single line. For each test case, the first output line corresponds to the first query pair, the second output line corresponds to the second query pair, and so on. The letters in each sequence indicate the first letter of the cities on the desired route between the query pair cities, including the query pair cities themselves. A city will never be paired with itself in a query.

Print a blank line between the outputs for two consecutive test cases.

Sample Input

```
1

7 3
Rome Turin
Turin Venice
Turin Genoa
Rome Pisa
Pisa Florence
Venice Athens
Turin Milan
Turin Pisa
Milan Florence
Athens Genoa
```

Sample Output

```
TRP
MTRPF
AVTG
```