Problem G

Railroad

Source: `railroad.(c|cc|pas|java)`
Input: `railroad.in`

It is Friday evening and Jill hates two things which are common to all trains:

1. They are always late.
2. The posted schedule is always wrong.

Nevertheless, tomorrow in the early morning hours Jill will have to travel from Tuttlingen to Freiburg in order to get to the Regional Programming Contest. Since she is afraid of arriving too late and being excluded from the contest, she is looking for the train which gets her to Freiburg as early as possible. However, she dislikes getting to the station too early, so if there are several schedules with the same arrival time, she will choose the one with the latest departure time.

Jill asks you to help her with her problem, so that she can sleep a bit longer tomorrow. You are given a set of railroad schedules from which you have to compute the fastest connection among those with the earliest arrival time for going from one location to another. One good thing: Jill is very experienced in switching trains: she can do this instantaneously, i.e., in zero time!!!

Input

The input file contains several scenarios. Each of them consists of three parts.

Part one lists the names of all cities connected by the railroads. It starts with a line containing an integer C $(1 \leq C \leq 100)$ followed by C lines containing city names. These names consist of letters.

Part two describes all the trains running during the day. It starts with a number $T \leq 1000$ followed by T train descriptions. Each train description consists of one line with a number $t_i \leq 100$ and t_i more lines with a time and a city name, meaning that passengers can get on or off the train at that time at that city. The times are given in the 24-hour format `hhmm`.

Part three consists of three lines: Line one contains the earliest possible starting time for the journey, line two the name of the city where she starts, and line three the destination city. The two cities are always different.

The end of the input file is marked by a line containing only a zero (instead of C). Do not process this line.

Output

For each scenario print the line "`Scenario #`*n*" where *n* is the number of the scenario starting at 1.

If a connection exists then print the two lines containing zero padded timestamps and locations as shown in the sample output. Use blanks to achieve the indentation. If no connection exists on the same day (i.e., arrival before midnight), then print a line containing "`No connection`".

After each scenario print a blank line.

Sample Input

```
3
Tuttlingen
Constance
Freiburg
3
2
0949 Tuttlingen
1006 Constance
2
1325 Tuttlingen
1550 Freiburg
2
1205 Constance
1411 Freiburg
0800
Tuttlingen
Freiburg
2
Ulm
Vancouver
1
2
0100 Ulm
2300 Vancouver
0800
Ulm
Vancouver
0
```

Sample Output

```
Scenario #1
Departure 0949 Tuttlingen
Arrival   1411 Freiburg

Scenario #2
No connection
```