Gate Allocation

input file: `gates.in`

The large, “traditional” airlines operate their routes on a “hub and spoke” system, which requires most passengers to fly from their originating city to the airline’s hub, and from there to their destination city. Your problem is to write a computer program to assign incoming flights to free gates in airports in order to maximize the average distance passengers must cover between gates when they make their connections.[1]

The distance between gates in each city is known and can be represented as a matrix. Below is a sample matrix for an airport with 5 gates:

		To gate				
		1	2	3	4	5
From gate	1	50	60	95	120	300
	2	70	0	30	150	200
	3	95	35	0	190	190
	4	117	150	200	120	150
	5	315	215	205	165	30

Notice that because of one way passages and security restrictions, the distance from gate A to gate B is not always the same as the distance from gate B to gate A. Additionally, since passengers are sometimes forced to get off even if they are on a connecting flight, you cannot assume the distance from a gate to itself is 0.

You should assume the following rules, too, to simplify the problem:

1. Passengers will always be connecting to planes that are already at the airport. Your program doesn’t have to “remember” that a passenger who has already landed needs to connect to a plane that isn’t yet at a gate. All passengers will arrive long enough before their connecting flights to make their connections.

2. Your program can ignore passengers that aren’t making connections (that is, ignore anyone who originates or ends up at the hub city). In fact, the data won’t even include passengers that aren’t making connections.

3. You can’t predict what flights will arrive in the future, nor when they will arrive, so your program should just assign each incoming flight to a gate in order to maximize the average distance the passengers on *that* flight will have to cover to get to their connections.

[1] Shouldn’t that be to **minimize** the average distance? You’d think so, but any frequent flyer can tell you that the traditional airlines must be trying to **maximize** it, so that’s what your program should do, too. Airport shop owners are especially glad for this policy.

4. Just as you can't predict the future, you can't change the past. It may happen that when a new plane arrives, your program will realize that if it had assigned a previous flight to a different gate, it could really increase the average distance the new passengers have to travel. But you can't do anything about it.

5. The program will work throughout a single day. At the start of the day, there will be several planes already at gates they arrived the previous day. At the end of the day, several planes will be left at gates instead of departing again that day.

6. All flights depart on schedule. (Okay, it's a fantasy problem.)

7. There will never be more airplanes at the airport than there are gates.

8. Oh, and there can be at most a single plane at a gate at a time. But, if a plane leaves a gate at a given time, another plane can arrive at that gate at the same time.

Assumption 3 and 4 are very important, and help make the problem much more tractable to solve.

Input

The program will consist of one or more input sets representing different airports. Each input set will begin with a three letter airport code, one space, and a single integer, n, $2 \leq n \leq 30$, the number of gates at the airport. There will then be n lines, each with n non-negative integers (each less than 10000), giving the distance in feet between each pair of gates. There will be at least one blank between the pairs of number on a line.

There will then be zero or more lines with a description of the planes already at gates. Each plane's data will be given on a separate line. Each line will start with a gate number (an integer g, $1 \leq g \leq n$), one or more spaces, the outgoing flight number (an integer between 1 and 9999, inclusive), one or more spaces, and the departure time. All times in this problem will be given as four digits for HHMM, where HH is between 00 and 23, inclusive, and MM is between 00 and 59 inclusive.

After the description of planes, there will be a line with just a 0 (zero) in it.

The remaining lines in the input set will describe incoming planes. The first line of information about a plane will have 4 integers. Each line starts with the arrival time, then one or more spaces, then the flight number (which will be the same flight number used by the plane when it departs the airport), the scheduled departure time (or 9999 if the plane won't leave the airport that day), and the number of connecting flights, c, $0 \leq c \leq n$. No two planes will arrive at the same time and they will be listed in ascending order of time. There will then be c lines of information about connecting flights. These will contain two integers, the connecting flight number, f and the number of passengers taking that connecting flight, p, $1 \leq f \leq 9999$ and $1 \leq p \leq 500$. The last line of the plane description will be a flight with arrival time of -1. This line will have no other data.

Input ends with an airport with 0 gates.

Output

For each airport, first print a line with the airport code. Then, for each arriving flight (in the same order as listed in the input file, print the flight number, the gate that causes the longest average passenger connection distance, and the average number of feet each passenger will travel to get to their connecting gate. This average should be rounded to the nearest integer. If two gates both cause the same longest average passenger connection distance, use the lower numbered gate. Have a blank line after each airport's information.

Sample input

```
MCN 5
50   60  95 120 300
70    0  30 150 200
95   35   0 190 190
117 150 200 120 150
315 215 205 165  30
1 712 0620
5 222 0800
0
0600 811 0905 2
712 15
222 23
0610 4321 1206 2
222 40
811 17
0749 1608 1105 0
0750 2301 1037 2
1608 29
4321 32
-1
BAG 0
```

Sample Output (corresponding to sample input)

```
Airport MCN
Flight 811: gate 3 (153 feet)
Flight 4321: gate 4 (165 feet)
Flight 1608: gate 1 (0 feet)
Flight 2301: gate 2 (112 feet)
```