– The Moon of Valencia –

It is well known that the Moon of Valencia is magical. Everyone talks about a mystery that happens at night. People remember what time they entered the first bar, what time arrived to the hotel and how happy they arrived, but nobody remembers the bars and pubs they visited.

The Valencia hotels have hired you for developing an application that helps customers to remember. The application will inform customers with one of all the possible sequences of bars and pubs. Customers have to provide the following information: departure time and place, arrival time and place, and degree of satisfaction on arrival.

The application uses a map with the location of each bar or pub. Each bar or pub produces a different degree of satisfaction when visited. But people gets angry when walks from one place to another, that's the reason why walking between different places reduces the degree of satisfaction. The reduction considered depends on the amount of minutes people needs to get one place from another one. If the amount of minutes is not an integer the remaining seconds should be considered a portion of a minute, i.e., 30 seconds imply 0.5 minutes. People walk at a speed of 4 km/h, can stay in a bar or pub the time they like, but for getting the satisfaction must remain at least 15 minutes. People can decide not to visit a bar or pub, i.e., they can use a path from the origin to the target and to enter in a subset of all the bars or pubs reachable with the path. Entering to the departure place is optional, like entering others places. The goal grade of satisfaction is computed up to the door of the target place, without entering it. So the grade of satisfaction of the target place (bar, pub or hotel) should not be computed.

INPUT

Input consists of several test cases. Each case begins with the map description, which is followed by the list of arrivals your application have to check. The description of a map begins with the word `MAP` in capital letters followed by two integer numbers, P and M, where P is the number of places and M is the number of paths connecting two places. ($1 \leq P \leq 64$). Places and paths are described one per line. Each place is described with two coordinates (real numbers represent kilometers), its grade of satisfaction (a real number), the ID and the name. The paths connecting two places are identified by the their identifiers. Each pair of places can only be connected by one path.

Figure 1 shows the map corresponding to the first map in the example input case. It is guaranteed that no crossing paths exist.

The list of arrivals to be processed begins after a line with the word `ARRIVALS` in uppercase. Each arrival is described in a single line, including departure time, departure place, arrival time, arrival place and the grade of satisfaction on arrival, a real number.

OUTPUT

The output for each case must begin with the word `MAP` in capital letters followed by a number indicating the number of test case. The first test case is `MAP 1`, second is `MAP 2`, and so on. For each arrival proposed in the input it must appear a line in the output specifying a valid path found or the string `Impossible!` indicating that it is not possible to find a path from origin to target with the required grade of satisfaction. A path found will be valid if the absolute difference between the required grade of satisfaction and the obtained one is less than 0.1, and contains no loops, i.e. a place can't appear in the path found more than once.

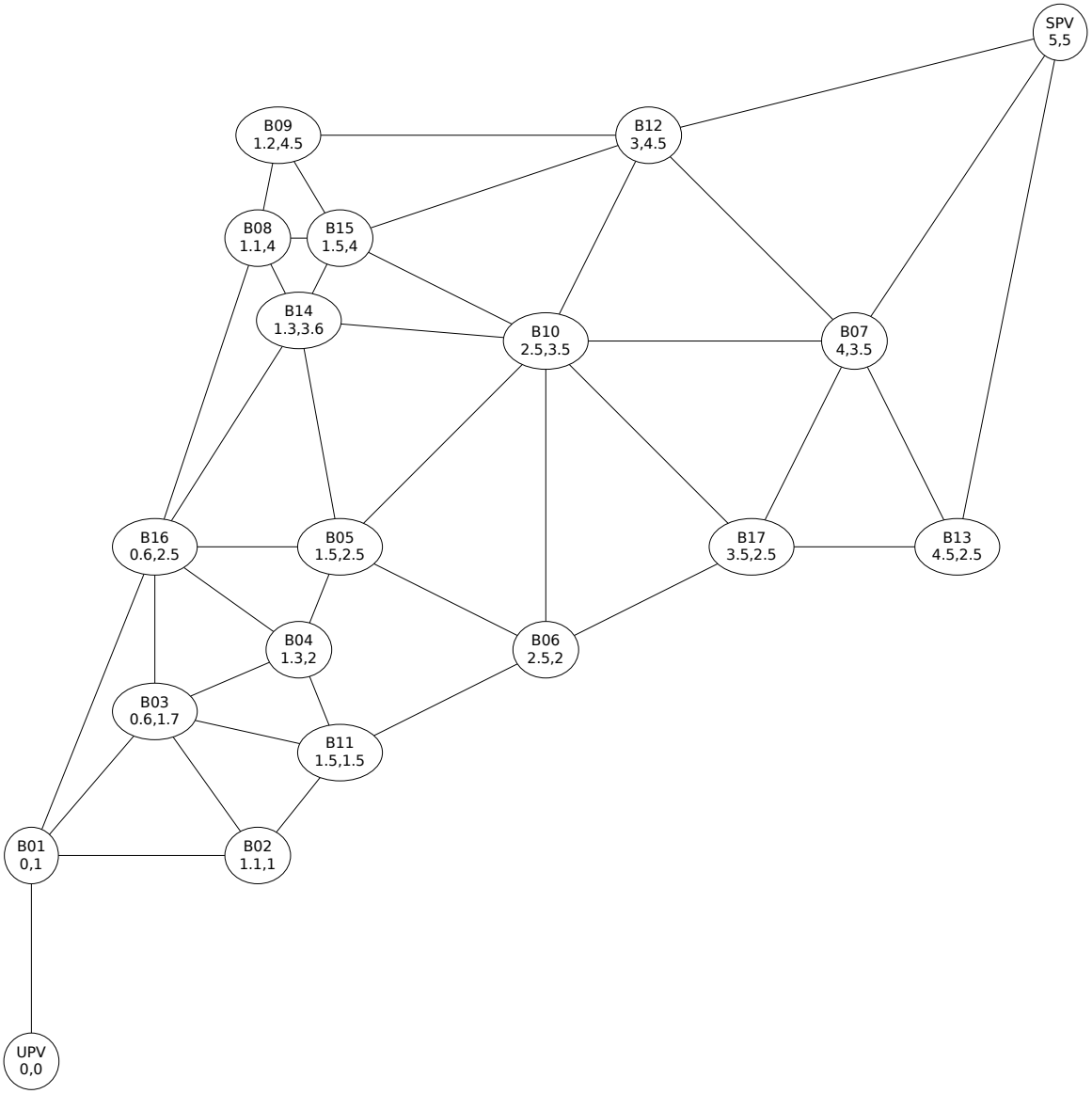


Figure 1: Representation of the first map in the input example.

Each path found must begin with the string `PATH FOUND:` in capital letters, followed by the obtained grade of satisfaction with three decimal digits and then the sequence of places from origin up to target. The ID of the unvisited places must appear preceded by the `!` sign. Notice as the ID of the target place never is preceded by this sign.

Hint: set up your solution for running as fast as possible by using this example, it should be enough for all test cases.

INPUT EXAMPLE

```
MAP 19 40
  0     0     0  UPV  Universitat Politecnica de Valencia
  5     5     0  SPV  Contest hotel
  0     1    35  B01  The Object
  1.1   1    42  B02  Opera
  0.6   1.7  33  B03  New York
  1.3   2    55  B04  Blue Note
  1.5   2.5  23  B05  The Popes
  2.5   2    13  B06  Petrol
  4     3.5  12  B07  King of Kings
  1.1   4    14  B08  O Salati
  1.2   4.5  13  B09  The Snails
  2.5   3.5  34  B10  The Earth
  1.5   1.5  55  B11  Cafe Coffee
  3     4.5  31  B12  Vermouth house
  4.5   2.5  45  B13  Jamon Session
  1.3   3.6  24  B14  Let's go to eat
  1.5   4    34  B15  I'm hungry
  0.6   2.5  53  B16  The Gecko
  3.5   2.5  43  B17  The Black Sheep
UPV B01
B01 B02
B01 B03
B01 B16
B02 B03
B02 B11
B16 B08
B16 B14
B16 B03
B03 B04
B03 B11
B04 B11
B04 B16
B04 B05
B05 B14
B08 B09
B08 B15
B08 B14
B11 B06
B14 B15
B05 B06
B05 B16
B05 B10
B15 B09
B15 B10
B09 B12
B06 B10
B06 B17
B10 B07
B10 B17
B10 B12
B10 B14
B12 B15
B12 B07
B12 SPV
B17 B07
B17 B13
B07 B13
B07 SPV
```

```
B13 SPV
ARRIVALS
23:00  UPV 03:00  SPV   9.0
23:00  UPV 03:00  SPV   8.0
23:00  UPV 03:00  SPV   7.0
23:00  UPV 03:00  SPV   6.0
23:00  UPV 03:00  SPV   5.0
23:00  UPV 03:00  SPV   4.0
23:00  UPV 03:00  SPV   3.0
23:00  UPV 03:00  SPV   2.0
23:00  UPV 03:00  SPV   1.0
23:00  UPV 03:00  SPV   0.0
23:00  UPV 03:00  SPV  -1.0
23:00  UPV 03:00  SPV  -2.0
23:00  UPV 03:00  SPV  -30.0
23:00  UPV 03:00  SPV  -40.0
23:00  B05 03:00  B10   40.0
23:00  B05 03:00  B10   30.0
23:00  B05 03:00  B10   20.0
23:00  B05 03:00  B10   10.0
23:00  B05 03:00  B10    0.0
23:00  B05 03:00  B10  -10.0
23:00  B05 03:00  B10  -20.0
23:00  B05 03:00  B10  -30.0
23:00  B05 03:00  B10  -40.0
MAP 2 1
 0  0 0 UPV Universitat Politecnica de Valencia
10 10 0 SPV Hotel Silken Puerta de Valencia
UPV SPV
ARRIVALS
23:00  UPV  1:00  SPV   9.0
23:00  UPV  1:00  SPV   8.0
```

OUTPUT EXAMPLE

```
MAP 1
PATH FOUND:    9.002   UPV  B01  B16  B04 !B11  B06 !B17 !B13 SPV
PATH FOUND:    7.929   UPV  B01  B16  B14  B08  B09 !B12 SPV
PATH FOUND:    6.973   UPV  B01  B16  B04 !B11  B06 !B10 !B07 SPV
PATH FOUND:    6.028   UPV  B01  B16  B05  B10 !B17 !B07 SPV
PATH FOUND:    4.995   UPV  B01  B16  B04 !B05 !B14 !B15 !B10  B07 SPV
PATH FOUND:    4.078   UPV  B01  B16  B08 !B15  B12  B07 SPV
PATH FOUND:    3.028   UPV  B01  B16  B05 !B10  B12 !B07 SPV
PATH FOUND:    1.929   UPV  B01  B16  B14 !B15  B08  B09 !B12 SPV
PATH FOUND:    0.912   UPV  B01  B16  B03 !B04 !B05 !B06  B17 !B13 !B07 SPV
PATH FOUND:    0.028   UPV  B01  B16  B05 !B10  B17 !B13 !B07 SPV
PATH FOUND:   -0.986   UPV  B01  B16  B08 !B09 !B15  B12 SPV
PATH FOUND:   -1.973   UPV  B01  B16  B03 !B04 !B05  B10 !B17 !B07 SPV
PATH FOUND:  -29.953   UPV  B01  B03 !B16  B05 !B10 !B14  B15 !B12 SPV
PATH FOUND:  -39.913   UPV  B01  B03 !B16  B08 !B09 !B15  B12 !B07 SPV
PATH FOUND:   40.069   B05  B16  B14 !B08  B09 !B15 B10
PATH FOUND:   30.012   B05  B16  B08 !B15 B10
PATH FOUND:   19.979  !B05  B04 !B03  B02 !B01  B16 !B08 !B09 !B15 B10
PATH FOUND:   10.004   B05  B14  B08 !B15 !B09  B12 B10
PATH FOUND:    0.004  !B05  B14  B08 !B15  B09  B12 B10
PATH FOUND:   -9.966   B05 !B14 !B15  B12 B10
PATH FOUND:  -20.012   B05 !B06 !B17  B07  B12  B15 !B14 B10
PATH FOUND:  -30.012  !B05  B06 !B17  B07  B12  B15 !B14 B10
PATH FOUND:  -40.018  !B05 !B04 !B16  B14 !B15 B10
MAP 2
Impossible!
Impossible!
```