Challenge 9 - Bendito Caos

« Prev Next »

You are organizing a huge party in your city, AwesomeVille. It's been a great success and everyone wants to come. People from other cities don't want to miss out on all the fun, so they are planning to drive to AwesomeVille for the super party.

There is going to be a major traffic jam, so you need to calculate how many people can come to AwesomeVille from each city to calculate the food and beverages.

There are two type of roads, normal roads and dirt roads. Depending on the area, the maximum speed of each road varies. Each road may have one or two lanes. All roads are unidirectional, independently of their number of lanes.

You have a map of the roads that connect each city with AwesomeVille.

Everyone wants to get there really fast, but they don’t want to break the law.

Input

A line with the number of cities **C**

The map of each city will contain:

- The name of the city
- The maximum speed for roads **S** (km/h) and the maximum speed for dirt roads **D** (km/h)
- The **I** intersection and **R** road numbers
- Each road will also be defined by a line:
    - from to type lanes
- from and to can be intersections (numbers from 0 to I-1), the name of the departure city or AwesomeVille. The type will be normal or dirt.

Observations: When you begin the calculations, the roads are already full. The size of each car is 4 m and there is only a space of 1 m between cars. The two lanes of a road are in the same direction. The roads and intersections from each city to AwesomeVille are not shared between cities. You may treat each case separately. All the numbers are integers.

Output

For each city, you must calculate the number of cars that can travel from the city in question to AwesomeVille in one hour.

Sample input

```
1
BoringVille
80 60
2 5
BoringVille 0 normal 1
BoringVille 1 normal 2
0 1 dirt 2
0 AwesomeVille dirt 1
1 AwesomeVille normal 1
```

Sample output

```
BoringVille 28000
```

Submit & test your code

To test and submit code we provide a set of tools to help you. Download contest tools if you haven't already done that. You will then be able to test your solution to this challenge with the challenge tokens.

```
Challenge tokens: CHALLENGE_9, CHALLENGE_SUBMIT_9
```

To test your program

```
./test_challenge CHALLENGE_9 path/program
```

A nice output will tell you if your program got the right solution or not. You can try as many times as you need.

To test your program against the input provided in the submit phase

```
./test_challenge CHALLENGE_SUBMIT_9 path/program
```

During the submit phase, in some problems, we might give your program harder inputs. As with the test token, a nice output will tell you if your program got the right solution or not. You can try as many times as you need.

In the actual contest you first need to solve the test phase before submitting the code, you must provide the source code used to solve the challenge and you can only submit once (once your solution is submitted you won't be able to amend it to fix issues or make it faster).

If you have any doubts, please check the info section.

« Prev Next »