Distance table

Submission deadline:	**2011-11-20 23:59:59**	298947.303 sec
Evaluation:	**0.0000**	
Max. assessment:	**3.0000** (Without bonus points)	
Submissions:	0 / 10 Free retries + 20 Penalized retries (-2 % penalty each retry)	
Advices:	0 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)	

The task is to develop a program which decides whether a distance table is consistent, or not.

Road maps usually provide distance tables for a set of important cities. It is quite easy to find the distance of any two cities in the table - just take the row corresponding to one of the cities and the column corresponding to the other city and you have the desired distance. The table is, of course, symmetric. The distance gives the travel distance between the two cities. It does not mean there is a direct road connecting the two cities.

The problem is to test validity of the table. Suppose the table gives: A B distance is 50km, B C distance is 80km and C A distance is 150km. Such combination is obviously a nonsense.

The input of the program is an integer `n` denoting the number of cities in the table. Then, there will be lines, one line describing one table cell. The lines will have general format city#1, city#2, and distance. The cities are identified by numbers 0, 1, 2, ..., the distance s are given in kilometers. Thus, input line `3 1 20` means that the distance from city 3 to city 1 is 50 km (an obviously the same applies to the opposite direction). The input is terminated when EOF is reached (EOF is active in standard input). There input does not have to describe all city pairs. If a pair of cities does not have the distance set, it means there is not any road connection of the two cities (e.g. the cities are on different islands, we do not consider ferries in our problem).

The output of the program is the validation of the table. The exact answers are shown in sample runs below.

The program must detect an invalid input. The following is considered invalid:

- non-numerical input values,
- zero or negative distances,
- invalid city identification,
- duplicate definitions of the distance (e.g. distance is set twice or more times for the same city pair).

The program is tested in a limited environment. Both time and memory is limited. The limits are set such that a correct implementation of a naive algorithm passes all tests.

Sample program run:

```
Enter number of cities:
3
Enter distances:
0 1 10
0 2 20
2 1 17
Distance table is consistent.
```

```
Enter number of cities:
3
Enter distances:
0 1 10
0 2 20
1 2 40
Distance table is inconsistent.
```

```
Enter number of cities:
4
Enter distances:
0 1 10
2 3 20
Distance table is consistent.
```

```
Enter number of cities:
4
Enter distances:
0 1 10
1 2 10
2 3 10
Distance table is inconsistent.
```

```
Enter number of cities:
2
Enter distances:
0 1 10
1 0 10
Invalid input.
```

```
Enter number of cities:
2
Enter distances:
0 1 -3
Invalid input.
```

```
Enter number of cities:
2
Enter distances:
0 1 abcd
Invalid input.
```

Help:

- If distance A -> B is greater than distance A -> C + C -> B, the table is wrong. Thus, naive solution may try to find all such possible "detours" and test the inequality.
- You have to store all input distances to test the detours. Use 2D array to store the distances.

Sample data: **Download**

Submit: [redacted] **Submit**

Reference