Programación Imperativa y Modular
PIMO 2013-1

Maratón -- Mayo 3

Grafos

Problems

Este cuadernillo contiene 3 problemas

(Prestados de muchas fuentes en línea; la plantilla fue tomada de `http://www.programmingleague.org`.)

A - Dueling Philosophers

Source file name: `dusk.c, dusk.cpp` *or* `dusk.java`

Following a sad and strange incident involving a room full of philosophers, several plates of spaghetti, and one too few forks, the faculty of the Department of Philosophy at ACM University have been going through the papers of a recently deceased colleague. The faculty members were amazed to find numerous unpublished essays. They believe that the essays, collected into one volume, may constitute a major work of scholarship that will give their department some much-needed positive publicity. Naturally, all of the faculty members began to vie for the honor (to say nothing of the fame) of serving as editor of the collection.
After much debate, the faculty members have narrowed the list to two candidates. Both applicants were asked to explain how they would arrange the essays within the final book. Both have noted that many of the essays define terminology and concepts that are explored in other essays. Both have agreed to the basic principle that an essay that uses a term must appear after the essay that defines that term. One of the candidates has presented what he claims is the only possible arrangement of the essays, under those constraints, and is arguing that he should be given the job simply because he has already done this major part of the work. The second candidate scoffs at this claim, insisting that there are many possible arrangements of the essays, and that an editor of true skill (himself) is needed to choose the optimal arrangement. Write a program to determine if zero, one, or more than one arrangement of the essays is possible.

Input

There will be multiple test cases in the input. Each test case will begin with a line with two integers, n ($1 \leq n \leq 1,000$) and m ($1 \leq m \leq 500,000$), where n is the number of essays, and m is the number of relationships between essays caused by sharing terms. They will be separated by a single space. On each of the next m lines will be two integers, d followed by u ($1 \leq d, u \leq n, d \neq u$) which indicate that some term is defined in essay d and used in essay u. Integers d and u will be separated by a single space. The input will end with two 0s on their own line.

The input must be read from standard input.

Output

For each test case, output a 0 if no arrangement is possible, a 1 if exactly one arrangement is possible, or a 2 if multiple arrangements are possible (output 2 no matter how many arrangements there are). Output no extra spaces, and do not separate answers with blank lines.

The output must be written to standard output.

Sample input	**Sample output**
5 4 1 5 5 2 3 2 4 3 5 4 3 1 4 2 1 5 5 4 2 2 1 2 2 1 0 0	2 1 0

B - Tsunami

Source file name: `tsunami.c`, `tsunami.cpp` *or* `tsunami.java`

The country of Cartesia can be described simply by a Cartesian plane. The x-axis is a shoreline. The positive y half-plane is land, and the negative y half-plane is ocean. Several large cities dot the mainland. Their positions can be described by coordinates (x, y), with $y > 0$. Unfortunately, there are sometimes tsunamis in the ocean near Cartesia. When this happens, the entire country can flood. The waters will start at $y = 0$ and advance uniformly in the positive y direction.
Cartesia is trying to develop a tsunami warning system. The warning system consists of two components: a single meteorological center which can detect a tsunami miles out, and wired connections which can carry the warning from city to city in straight lines (No wireless communication!!). Despite the lack of wireless communications, the wired connections can carry the warning virtually instantaneously.
A city is considered safe if it either has the meteorological center, or if it has a direct wire connection to another safe city (i.e. if it has a multi-hop cable path to the single meteorological center).
Unfortunately, a simple engineering problem is made more complicated by politics! If a city A receives the warning from city B, and city B is further away from the shore than city A, then city A's leaders will complain! ‘‘‘We're closer to the ocean than city B, so the warning should have gone through us!’’ With a sigh, you agree to find a solution where no city will get the warning from a city that's further from the shore. (This means that the meteorological center must be in a city that's closest to the shore.)
Given a description of Cartesia, find the least amount of cable necessary to build a tsunami warning system where every city is *safe*, and no city will receive the warning from another city that is further from the shore.

Input

There will be several test cases in the input. Each test case will begin an integer n ($1 \leq n \leq 1,000$) on its own line, indicating the number of cities. On each of the next n lines will be a pair of integers x and y ($-1,000 \leq x \leq 1,000, 0 < y \leq 1,000$), each of which is the (x, y) location of a city. These integers will be separated by a single space. Within a test case, all (x, y) pairs will be unique. The input will end with a line containing a single 0.

The input must be read from standard input.

Output

For each test case, output a single real number on its own line, which is the minimum amount of cable which must be used to build the tsunami warning system. Output this number with exactly two decimal places. Output no extra spaces, and do not separate answers with blank lines.

The output must be written to standard output.

Sample input	**Sample output**
3 100 10 300 10 200 110 4 100 10 300 10 200 110 200 60 0	341.42 361.80

C - From Dusk Till Down

Source file name: `dusk.c, dusk.cpp` *or* `dusk.java`

Vladimir has white skin, very long teeth and is 600 years old, but this is no problem because Vladimir is a vampire. Vladimir has never had any problems with being a vampire. In fact, he is a very successful doctor who always takes the night shift and so has made many friends among his colleagues. He has a very impressive trick which he shows at dinner partys: He can tell tell blood group by taste.
Vladimir loves to travel, but being a vampire he has to overcome three problems.

- First, he can only travel by train because he has to take his coffin with him. (On the up side he can always travel first class because he has invested a lot of money in long term stocks.)
- Second, he can only travel from dusk till dawn, namely from 6 pm to 6 am. During the day he has to stay inside a train station.
- Third, he has to take something to eat with him. He needs one litre of blood per day, which he drinks at noon (12:00) inside his coffin.

You should help Vladimir to find the shortest route between two given cities, so that he can travel with the minimum amount of blood. (If he takes too much with him, people will ask funny questions like ‘‘What do you do with all that blood?’’)

Input

The first line of the input will contain a single number telling you the number of test cases. Each test case specification begins with a single number telling you how many route specifications follow. Each route specification consists of the names of two cities, the departure time from city one and the total travelling time. The times are in hours.
Note that Vladimir can't use routes departing earlier than 18:00 or arriving later than 6:00. There will be at most 100 cities and less than 1000 connections. No route takes less than one hour and more than 24 hours. (Note that Vladimir can use only routes with a maximum of 12 hours travel time (from dusk till dawn).) All city names are shorter than 32 characters.
The last line contains two city names. The first is Vladimir's start city, the second is Vladimir's destination.

The input must be read from standard input.

Output

For each test case you should output the number of the test case followed by

```
Vladimir needs # litre(s) of blood.
```

or

```
There is no route Vladimir can take.
```

The output must be written to standard output.

Sample input	**Sample output**
2 3 Ulm Muenchen 17 2 Ulm Muenchen 19 12 Ulm Muenchen 5 2 Ulm Muenchen 10 Lugoj Sibiu 12 6 Lugoj Sibiu 18 6 Lugoj Sibiu 24 5 Lugoj Medias 22 8 Lugoj Medias 18 8 Lugoj Reghin 17 4 Sibiu Reghin 19 9 Sibiu Medias 20 3 Reghin Medias 20 4 Reghin Bacau 24 6 Lugoj Bacau	Test Case 1. There is no route Vladimir can take. Test Case 2. Vladimir needs 2 litre(s) of blood.