# Arbitrage

Time Limit: 3 sec

**Description**

The use of computers in the finance industry has been marked with controversy lately as programmed trading -- designed to take advantage of extremely small fluctuations in prices -- has been outlawed at many Wall Street firms. The ethics of computer programming is a fledgling field with many thorny issues. Arbitrage is the trading of one currency for another with the hopes of taking advantage of small differences in conversion rates among several currencies in order to achieve a profit. For example, if $1.00 in U.S. currency buys 0.7 British pounds currency, ?1 in British currency buys 9.5 French francs, and 1 French franc buys 0.16 in U.S. dollars, then an arbitrage trader can start with $1.00 and earn dollars thus earning a profit of 6.4 percent. You will write a program that determines whether a sequence of currency exchanges can yield a profit as described above. To result in successful arbitrage, a sequence of exchanges must begin and end with the same currency, but any starting currency may be considered.

**The Input**

The input file consists of one or more conversion tables. You must solve the arbitrage problem for each of the tables in the input file. Each table is preceded by an integer n on a line by itself giving the dimensions of the table. The maximum dimension is 20; the minimum dimension is 2. The table then follows in row major order but with the diagonal elements of the table missing (these are assumed to have value 1.0). Thus the first row of the table represents the conversion rates between country 1 and n-1 other countries, i.e., the amount of currency of country i that can be purchased with one unit of the currency of country 1. Thus each table consists of n+1 lines in the input file: 1 line containing n and n lines representing the conversion table.

**The Output**

For each table in the input file you must determine whether a sequence of exchanges exists that results in a profit of more than 1 percent (0.01). If a sequence exists you must print the sequence of exchanges that results in a profit. If there is more than one sequence that results in a profit of more than 1 percent you must print a sequence of minimal length, i.e., one of the sequences that uses the fewest exchanges of currencies to yield a profit. Because the IRS (United States Internal Revenue Service) notices lengthy transaction sequences, all profiting sequences must consist of n or fewer transactions where n is the dimension of the table giving conversion rates. The sequence 1 2 1 represents two conversions. If a profiting sequence exists you must print the sequence of exchanges that results in a profit. The sequence is printed as a sequence of integers with the integer i representing the line of the conversion table (country i). The first integer in the sequence is the country from which the profiting sequence starts. This integer also ends the sequence. If no profiting sequence of n or fewer transactions exists, then the line no arbitrage sequence exists

**Sample Input**

3

1.2 .89

.88 5.1

1.1 0.15

4

3.1 0.0023 0.35

0.21 0.00353 8.13

200 180.559 10.339

2.11 0.089 0.06111

2

2.0

0.45

**Sample Output**

1 2 1

1 2 4 1

no arbitrage sequence exists

*Author: WilliamsDing*

**The Postal Worker Rings Once**

Time Limit: 3 sec

**Description**

Graph algorithms form a very important part of computer science and have a lineage that goes back at least to Euler and the famous Seven Bridges of K?nigsberg problem. Many optimization problems involve determining efficient methods for reasoning about graphs.  
This problem involves determining a route for a postal worker so that all mail is delivered while the postal worker walks a minimal distance, so as to rest weary legs.  
Given a sequence of streets (connecting given intersections) you are to write a program that determines the minimal cost tour that traverses every street at least once. The tour must begin and end at the same intersection.  
The ``real-life'' analogy concerns a postal worker who parks a truck at an intersection and then walks all streets on the postal delivery route (delivering mail) and returns to the truck to continue with the next route.  
The cost of traversing a street is a function of the length of the street (there is a cost associated with delivering mail to houses and with walking even if no delivery occurs).  
In this problem the number of streets that meet at a given intersection is called the degree of the intersection. There will be at most two intersections with odd degree. All other intersections will have even degree, i.e., an even number of streets meeting at that intersection.

**The Input**

The input consists of a sequence of one or more postal routes. A route is composed of a sequence of street names (strings), one per line, and is terminated by the string ``deadend'' which is NOT part of the route. The first and last letters of each street name specify the two intersections for that street, the length of the street name indicates the cost of traversing the street. All street names will consist of lowercase alphabetic characters.  
For example, the name foo indicates a street with intersections f and o of length 3, and the name computer indicates a street with intersections c and r of length 8. No street name will have the same first and last letter and there will be at most one street directly connecting any two intersections. As specified, the number of intersections with odd degree in a postal route will be at most two. In each postal route there will be a path between all intersections, i.e., the intersections are connected.

**The Output**

For each postal route the output should consist of the cost of the minimal tour that visits all streets at least once. The minimal tour costs should be output in the order corresponding to the input postal routes.

**Sample Input**

one

two

three

deadend

mit

dartmouth

linkoping

tasmania

york

emory

cornell

duke

kaunas

hildesheim

concord

arkansas

williams

glasgow

deadend

**Sample Output**

11

114

*Author: WilliamsDing*

**Frogger**

Time Limit: 3 sec

**Description**

Freddy Frog is sitting on a stone in the middle of a lake. Suddenly he notices Fiona Frog who is sitting on another stone. He plans to visit her, but since the water is dirty and full of tourists' sunscreen, he wants to avoid swimming and instead reach her by jumping.  
Unfortunately Fiona's stone is out of his jump range. Therefore Freddy considers to use other stones as intermediate stops and reach her by a sequence of several small jumps.  
To execute a given sequence of jumps, a frog's jump range obviously must be at least as long as the longest jump occuring in the sequence.  
The frog distance (humans also call it minimax distance) between two stones therefore is defined as the minimum necessary jump range over all possible paths between the two stones.  
You are given the coordinates of Freddy's stone, Fiona's stone and all other stones in the lake. Your job is to compute the frog distance between Freddy's and Fiona's stone.

**The Input**

The input file will contain one or more test cases. The first line of each test case will contain the number of stones n . The next n lines each contain two integers xi, yi representing the coordinates of stone #i. Stone #1 is Freddy's stone, stone #2 is Fiona's stone, the other n-2 stones are unoccupied. There's a blank line following each test case. Input is terminated by a value of zero (0) for n.

**The Output**

For each test case, print a line saying ``Scenario #x" and a line saying ``Frog Distance = y" where x is replaced by the test case number (they are numbered from 1) and y is replaced by the appropriate real number, printed to three decimals. Put a blank line after each test case, even after the last one.

**Sample Input**

2

0 0

3 4

3

17 4

19 4

18 5

0

**Sample Output**

Scenario #1

Frog Distance = 5.000

Scenario #2

Frog Distance = 1.414

*Author: WilliamsDing*

**Longest Paths**

Time Limit: 3 sec

**Description**

It is a well known fact that some people do not have their social abilities completely enabled. One example is the lack of talent for calculating distances and intervals of time. This causes some people to always choose the longest way to go from one place to another, with the consequence that they are late to whatever appointments they have, including weddings and programming contests. This can be highly annoying for their friends.  
César has this kind of problem. When he has to go from one point to another he realizes that he has to visit many people, and thus always chooses the longest path. One of César's friends, Felipe, has understood the nature of the problem. Felipe thinks that with the help of a computer he might be able to calculate the time that César is going to need to arrive to his destination. That way he could spend his time in something more enjoyable than waiting for César.  
Your goal is to help Felipe developing a program that computes the length of the longest path that can be constructed in a given graph from a given starting point (César's residence). You can assume that the graph has no cycles (there is no path from any node to itself), so César will reach his destination in a finite time. In the same line of reasoning, nodes are not considered directly connected to themselves.

**The Input**

The input consists of a number of cases. The first line on each case contains a positive number n that specifies the number of points that César might visit (i.e., the number of nodes in the graph). A value of n = 0 indicates the end of the input. After this, a second number s is provided, indicating the starting point in César's journey . Then, you are given a list of pairs of places p and q, one pair per line, with the places on each line separated by white-space. The pair ``" indicates that César can visit q after p. A pair of zeros (``0 0") indicates the end of the case. As mentioned before, you can assume that the graphs provided will not be cyclic.

**The Output**

For each test case you have to find the length of the longest path that begins at the starting place. You also have to print the number of the final place of such longest path. If there are several paths of maximum length, print the final place with smallest number.  
Print a new line after each test case.

**Sample Input**

2

1

1 2

0 0

5

3

1 2

3 5

3 1

2 4

4 5

0 0

5

5

5 1

5 2

5 3

5 4

4 1

4 2

0 0

0

**Sample Output**

Case 1: The longest path from 1 has length 1, finishing at 2.

Case 2: The longest path from 3 has length 4, finishing at 5.

Case 3: The longest path from 5 has length 2, finishing at 1.

*Author: WilliamsDing*

**The Tourist Guide**

Time Limit: 3 sec

**Description**

Mr. G. works as a tourist guide. His current assignment is to take some tourists from one city to another. Some two-way roads connect the cities. For each pair of neighboring cities there is a bus service that runs only between those two cities and uses the road that directly connects them. Each bus service has a limit on the maximum number of passengers it can carry. Mr. G. has a map showing the cities and the roads connecting them. He also has the information regarding each bus service. He understands that it may not always be possible for him to take all the tourists to the destination city in a single trip. For example, consider the following road map of 7 cities. The edges connecting the cities represent the roads and the number written on each edge indicates the passenger limit of the bus service that runs on that road.  
  
Now, if he wants to take 99 tourists from city 1 to city 7, he will require at least 5 trips, since he has to ride the bus with each group, and the route he should take is : 1 - 2 - 4 - 7.  
But, Mr. G. finds it difficult to find the best route all by himself so that he may be able to take all the tourists to the destination city in minimum number of trips. So, he seeks your help.

**The Input**

The input will contain one or more test cases. The first line of each test case will contain two integers: N (N= 100) and R representing respectively the number of cities and the number of road segments. Then R lines will follow each containing three integers: C1, C2 and P. C1 and C2 are the city numbers and P (P> 1) is the limit on the maximum number of passengers to be carried by the bus service between the two cities. City numbers are positive integers ranging from 1 to N. The (R + 1)-th line will contain three integers: S, D and T representing respectively the starting city, the destination city and the number of tourists to be guided.  
The input will end with two zeroes for N and R.

**The Output**

For each test case in the input first output the scenario number. Then output the minimum number of trips required for this case on a separate line. Print a blank line after the output of each test case.

**Sample Input**

7 10

1 2 30

1 3 15

1 4 10

2 4 25

2 5 60

3 4 40

3 6 20

4 7 35

5 7 20

6 7 30

1 7 99

0 0

**Sample Output**

Scenario #1

Minimum Number of Trips = 5

*Author: WilliamsDing*