Treino multi-institucional: UDESC, UFPR, UTFPR e PUC-PR
Local: UFPR

BOCA: http://maratona.c3sl.ufpr.br/boca

Os basenames de cada programa são os seguintes

A - multiples
B - card
C - dig
D - salesman
E - street
F - hostel
G - air
H - letters
J - matrix
K - standing
L - thegod
M - connecting
N - adjacent

Se esta utilizando Java, utilize estes nomes como o nome da sua classe.

Os problemas estão ordenados em ordem de dificuldade.

PROBLEM A **FACTORS AND MULTIPLES** **3 POINTS**

As I am sure you are aware, 4 x 3 = 12.

This means that 3 is a factor of 12 and that 12 is a multiple of 3.

This also means that 4 is a factor of 12 and that 12 is a multiple of 4.

In this problem you will be given sets of two numbers and have to decide which of three relationships applies:

1. The first number is a factor of the second number
2. The first number is a multiple of the second number
3. The first number is neither a factor nor a multiple of the second number.

Input consists of a series of lines, each line containing two positive integers, both less than 10,000, separated by a space. Input is finished when the numbers are 0 0; do not process this line.

Output consists of one line for each line of input. The line will consist of the word `factor` if the first number is a factor of the second number, the word `multiple` if the first number is a multiple of the second number or the word `neither` if the first number is neither a factor nor a multiple of the second number.

Sample Input	**Output for Sample Input**
`8 16`	`factor`
`32 4`	`multiple`
`17 5`	`neither`
`0 0`	

PROBLEM B **CARD CUTTING** **3 POINTS**

Cheryl and Tania frequently play a simple card cutting game. They remove the picture cards then take it in turns to cut the pack. Every time an odd card turns up, Cheryl gets a point, every time an even card turns up, Tania scores one. When they get fed up, they add up the points.

Input consists of data for a number of games. Each game consists of a list of card values on a single line, each value separated by a single space. '`A`' represents an Ace which counts as a 1. The highest card value is 10. The last character on the line is a * to indicate the end of the game – do not process it as a card.

Input is terminated by a line containing just a # - that line should not be processed.

Output consists of one line for each game giving the name of the winner, `Cheryl` or `Tania`, or the word `Draw` if both girls score the same number of points.

Sample Input

```
A 2 8 3 10 A 3 A 5 *

2 4 9 3 6 3 7 8 *

#
```

Output for Sample Input

```
Cheryl

Draw
```

Explanation

Cheryl scores 6 points for A, 3, A, 3, A and 5, Tania scores 3 points for 2, 8 and 10.

Cheryl scores 4 points for 9, 3, 3 and 7, Tania also scores 4 points for 2, 4, 6 and 8.

PROBLEM C **ARCHAEOLOGICAL DIGS** **3 POINTS**

Archaeologists at a dig typically divide up the area they are examining into a grid, and will record in which grid cell each item is found. It is thus quite easy to tell how many items were found in a given cell.

In this problem you will be given a number of scenarios. Each scenario begins with a line containing two digits X and Y (separated by a space) representing the length and width of the grid (*0 < X, Y <= 100*). A scenario in which X and Y are both 0 marks the end of input.

The second line of the scenario is a single digit M (*0 < M <= 10000*) which gives the number of items located by the archaeologists. This is followed by M lines each containing the X and Y coordinates of the grid cell in which an item was found. Note that the grid coordinate system starts at 0, 0 and that several items may be found in a particular cell, so cell coordinates may be repeated.

Following the M lines of item locations there is a list of cell references for which the total number of found items is required. The first line of this section is a single integer, N, which gives the number of cells (*0 < N <= (X * Y)*). There follows N lines each containing the X and Y coordinates of a cell.

Output consists of a single line for each scenario. It contains the total number of items found in the N cells listed.

Sample Input

```
10 10
8
4 5
3 4
0 0
1 5
9 9
5 6
3 4
9 9
3
9 9
4 5
6 3
0 0
```

Output for Sample Input

```
3
```

Explanation

Cell 9,9 contains 2 items (it appears twice in the input list), cell 4,5 contains 1 and cell 6,3 contains none (it did not occur in the input list).

PROBLEM D **TRAVELLING SALESMAN** **3 POINTS**

Bob Smith has to tour New Zealand visiting his company's customers. His database churns out a list of the towns where each customer lives, but it has not been well programmed so may display a given town more than once. Your job is to help Bob by removing the duplicates and telling him how many towns he actually has to visit.

Input consists of a number of lists, each representing a week of visits. The first line of each week is a single integer, N (*1 < N <= 100*), which is the number of towns in the list. Input is terminated by N = 0 – this week should not be processed.

Each week contains a list of N towns, each on a line by itself. The name of a town may contain more than one word. The first letter of each word in a town's name begins with an upper case letter; all other letters are lower case. A town's name will contain no more than 20 characters.

Output consists of a single line for each week. It contains the word Week, followed by a space, followed by the week number, the first week being 1, followed by a space, followed by the actual number of towns to be visited, duplicates having been removed.

Sample Input

```
5
Wellsford
Ruakaka
Marsden Point
Wellsford
Warkworth
4
Rangiora
Oxford
Oxford
Rangiora
0
```

Output for Sample Input	**Explanation**
`Week 1 4`	Wellsford is repeated
`Week 2 2`	Rangiora and Oxford – both repeated

PROBLEM E **STREET NUMBERS** **10 POINTS**

The NZ Number Company (NZNC) produces metal numbers that can be placed at the front of a house to display its street number. When a new street of houses is built, NZNC often get orders from the builders to supply numbers for the entire street. To avoid waste, NZNC need to be able to work out how many of each digit they require to complete the order.

The builders supply NZNC with the range of house numbers required for the new street. Sometimes there is a gap in the houses (for example where there is a school or a sports ground) so a range of numbers will not be required.

Input consists of several scenarios, each starting with three integers, L, H and G. Input is terminated with all three equal to 0 – that line should not be processed. L represents the lowest number of a house on the street and H represents the highest number (*0 < L <= H <= 999*). G indicates the number of gaps in the housing that have to be accommodated (*0 <= G < 20*).

If G is zero, then the company has to supply numbers for all houses from lowest to highest inclusive. Otherwise (if there are missing houses which do not require numbers) this is indicated by G sections following L and H on the same line. A section consists of two integers L1 and H1, and one of the letters A, E or O, all separated by single spaces. L1 is the lowest number of the block of missing houses, H1 the highest. H1 may equal L1 to indicate a single house missing (*L <= L1, H1 <= H*). The letter A means all houses in the range are missing, E means only even numbered houses, O only odd numbered houses. You may assume that gaps do not overlap, so no house will be excluded more than once.

Output consists of 1 line per scenario. The line consists of 10 integers, separated by single spaces. The output numbers represent the number of each digit required to complete the order from 0 (leftmost) to 9 (rightmost). If a digit is not required, 0 must be displayed in the appropriate place.

Sample Input

```
10 20 0

1 50 1 12 18 E

0 0 0
```

Output for Sample Input	**Explanation**
`2 11 2 1 1 1 1 1 1 1`	`10 11 12 13 14 15 16 17 18 19 20` Count them!
`5 11 14 15 14 6 4 5 4 5`	12, 14, 16 and 18 are missing

PROBLEM F **HOSTEL NIGHTS** **10 POINTS**

Students staying in a hostel on the first floor have a reputation for being a bit too noisy. The warder decides to investigate reports of noisiness over the course of 5 nights of a week. Other students are not willing to dob in their floor-mates directly but will help him eliminate some rooms where the students were **not** noisy. They are eliminated through being either odd or even, because they are a multiple of some number n, or because the inhabitant's name starts with a particular letter.

Over the 5 nights the warden manages to form 3 variables from various other students' help. At the end of 5 nights he will be able to work out who the noisiest students are. There is at least one as this floor is notorious.

Input begins with a single integer, W, being the number of weeks worth of data (each week being 5 nights). The data for each week begins with 20 lines, each showing a room number and the name of a student, separated by a space. Rooms are numbered from 101 to 120. A student is represented by a single name.

This names list is followed by 5 lines each representing one night of the week. Each line consists of a letter, a number and another letter, each of which is separated by a space. The first letter is E or O to indicate whether even (E) or odd (O) numbered rooms may be eliminated. The number is used to eliminate all rooms that are a multiple of that number. The second letter may be used to eliminate any students whose name begins with that letter.

Output for each week consists of the week number, where the first week is 1, followed by a list of the noisiest students; that is those who have not been eliminated on the greatest number of nights. Names are output in room number order, one per line.

Turn over for sample input and output.

Sample Input

```
1
101 Fred
102 Gregory
103 Susan
104 Rewi
105 Albert
106 Georgina
107 Peter
108 Bethany
109 Sarah
110 Justine
111 Barry
112 Matthew
113 Justin
114 Chris
115 Devina
116 Yong
117 William
118 Edward
119 Ruth
120 Luckylast
O 5 G
E 3 F
E 5 S
E 1 A
O 4 C
```

113 Francis

Output for Sample Input

```
Week 1
Peter
Edward
Ruth
```

Explanation

Noisy students (those not eliminated) on each night are:

Night 1: Rewi, Bethany, Matthew, Chris, Yong, **Edward**

Night 2: Susan, **Peter**, Sarah, Devina, **Ruth**

Night 3: Fred, **Peter**, Barry, Francis, William, **Ruth**

Night 4: (none)

Night 5: Gregory, Georgina, Justine, **Edward**

Peter, Edward and Ruth appear twice each so are the noisiest students.

PROBLEM G **AIR OLD ZEELAND** **10 POINTS**

Air Old Zeeland, or as it is informally known, Air OZ has taken to allowing customers to redeem their loyalty points for products. As it is only a recent idea, Air OZ is testing the market by offering a small but select number of products. Unfortunately the suppliers are not very good at keeping up with the demand and Air OZ decided to record the number of days a customer would have to wait for each product along with its listed price. This will enable them to get a feel for how many discontented customers they have

Input

Input will consist of a number of scenarios. The first line of each scenario contains a number N (*0 < N <= 50*) which represents the number of products in this offering. End of input is marked by a scenario with 0 products – this line should not be processed.

The product count is followed by N lines with each of Air OZ's products listed on its own line. Each product name, which is limited to 20 characters and with no embedded spaces, is followed by the number of loyalty points needed to acquire the reward and the days to wait for it to be shipped to the purchaser. Each of these fields is separated by a single space. No product costs more than 1000 loyalty points and no product will be delayed by more than 100 days.

These N lines are followed by a line with the number of customers to be processed, C (*0 < C <= 500*). For each customer, there follows a line with the customer number, the number of products wishing to be purchased, P, and the maximum days they are prepared to wait for a product, M. (*0 < P <= N*, *0 < M <100*) Each of these 3 numbers is separated by a single space. There follows P lines listing the products they wish to order, each on a single line.

Output

For each customer in a scenario, output the customer number followed by the value of their purchases, separated by a space, on a single line. If a product is not available within the timeframe set by the customer, then it is deemed that the product has not been bought, and that the customer is therefore discontented. On the same line, following the value, output an asterisk (*) if that customer has not purchased something they wanted due to shipping delays. The value and asterisk, if applicable, should be separated by a single space. The final line of each scenario's output contains the text "`Number of discontented customers is: `", followed by the number of customers in that scenario who could not obtain all their products (ie the number of asterisks).

Turn over for sample input and output.

Sample Input

```
3
iPodNano 255 0
DucksFeetPerfume 120 15
SilverCharmPendant 180 3
3
1001 2 3
iPodNano
SilverCharmPendant
1860 1 5
DucksFeetPerfume
1025 2 6
iPodNano
DucksFeetPerfume
0
```

Output for Sample Input

```
1001 435
1860 0 *
1025 255 *
Number of discontented customers is: 2
```

Explanation :

The 2 products 1001 wants can both be shipped within the customer's time frame.

Customer 1860's purchase can't be shipped within his timeframe and so no purchase is made, and the customer is discontented.

With customer 1025, one product can be shipped in an acceptable time, the other cannot. The iPod is thus sold (cost 255) but the perfume is not, hence the *.

This makes 2 customers who could not complete their purchases.

PROBLEM H **LETTER REPLACEMENT** **10 POINTS**

Mr Sythe is teaching an ESL class about repeated letters in English words. As an exercise, he gets his students to replace all the repeated letters in a word with symbols.

The symbols used are as follows:

* is used to replace the first repeated letter (the first letter encountered which has occurred before)
? for the second repeated letter
/ for the third repeated letter
+ for the fourth repeated letter
! for the fifth repeated letter.

No word that Mr Sythe uses has more than 5 repeated letters.

So, for example, the word **Reindeer** would become **Reind**?** because e is repeated twice and r is repeated once. The repeated e comes before the repeated r, hence the allocation of * to e and ? to r. Note that the first letter in the word is an upper case R, but this is treated as the same letter as the lower case r.

In this problem, you will write a program to help Mr Sythe mark the exercise by giving him a list of correct answers. Input will consist of a list of words, one per line. Each word begins with an upper case letter and contains no more than 10 letters. The last line contains just a # - do not process this line.

Output will be one word for each line of input each on a separate line. The output word must be the input word with repeated letters replaced as indicated by Mr Sythe's rules.

Sample Input	**Output For Sample Input**
`Reindeer`	`Reind**?`
`Bubbles`	`Bu**les`
`Occurrence`	`Oc*ur?en*/`
`#`	

Problem J **Matrix Powers** **30 points**

Your task is to write a program to raise an integer matrix to a given power using modulo arithmetic. There is no back story here; at least not one that can be told. The application is too confidential (spying and military intelligence and all that) to be described in public.

For example, to raise the 2 by 2 matrix

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

to the power 2 using modulo 17 arithmetic

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} (1*1+2*3)\ mod\ 17 & (1*2+2*4)\ mod\ 17 \\ (3*1+4*3)\ mod\ 17 & (3*2+4*4)\ mod\ 17 \end{pmatrix} = \begin{pmatrix} 7 & 10 \\ 15 & 5 \end{pmatrix}$$

Input

The input consists of a number of problems. Each problem starts with a line holding three numbers (N, M, and P) separated by single spaces. 1 <= N <= 100 is the size (N by N) of the matrix to be processed. 2 <= M <= 32000 is the modulo base and 1 <= P <= 32000 is the power to which the matrix must be raised. Following this line are N lines, each holding the n integer values of successive rows of the matrix as a series of positive integers i: 0 <= i < M separated by single spaces. Input is terminated by a line with three zeros.

Output

Output for each problem consists of a blank line, followed by the N rows of the result matrix. Each row is output on a single line as a sequence of integer values separated by single spaces. It doesn't matter that lines may be quite long – no-one will be allowed to read them anyway.

Sample Input

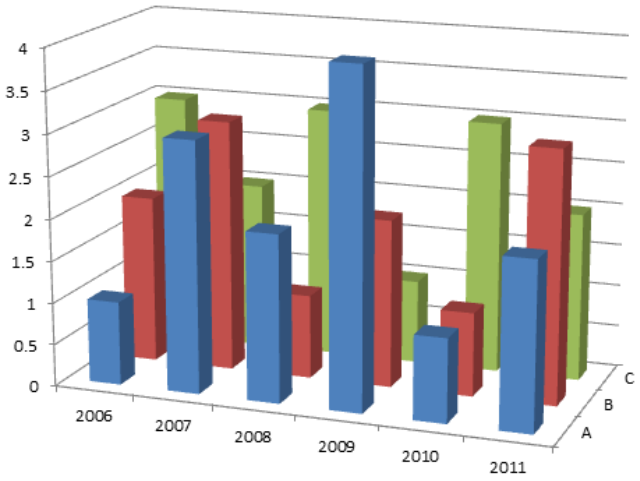
```
2 17 2
1 2
3 4
0 0 0
```

Sample Output

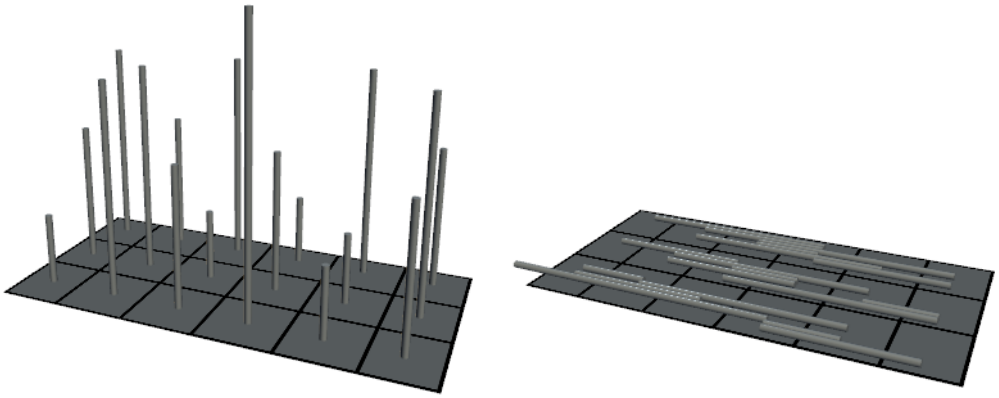
```

7 10
15 5
```

Problem K **Standing Pins** **30 points**

For a special presentation to the Computer Game Developers association Mario made up a two dimensional histogram of showing number of game titles published by year and game genre. As displayed by his spreadsheet software it was quite attractive.

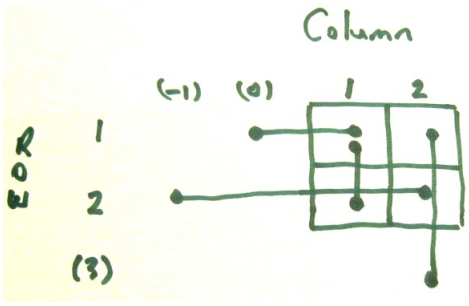


Mario decided that a physical model would be even better and built one from wire and cardboard – see left picture below. When packed for transport to the conference it looked like the right hand picture below. Each wire had been laid down, either to the left or right at random, while keeping its base in the correct square.

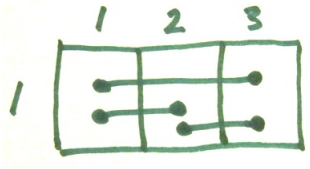


Only when he came to unpack and get his model set up again did he realise that there might be a problem. Whether each wire had been laid down to the left or to the right had not been recorded. Even worse, his assistant had not followed the packing instructions correctly. Not all wires had been laid left or right. Some had been laid forward and backward on the card. Of course the original data was not available. You have been asked to write software to figure out how to stand the wires up. In case the problem occurs again, you have been asked to make the software quite general.

Some problems will have solutions. For example, in the sketch to the right there are four pins of lengths 1, 1, 3 and 2. They can only stand up with the two length 1 pins in column 1; and the length 2 and 3 pins in column2 (rows 1 and 2 respectively).



Other problems will not have unique solutions. For example consider the second sketch. The three pins can be stood up with the length 2 pin either at the right or at the left. When there are multiple solutions in this way we cannot be sure as to which is correct, therefore we must state that there is no solution. Note however that it would not have been a problem if the length 2 pin was only 1 unit long. In that case we might not be sure which cell pins had originally occupied, but we would be sure that each cell had started with a pin of height one. That is ok.



Input

The input consists of a number of problems. Each problem starts with a line holding two numbers R, and C, the number of Rows and the number of columns of the grid. 1 <= R, C <= 100. This will be followed by R * C lines. Each line will hold the grid coordinates of the two ends of a piece of wire as four numbers r1, c1, r2, c2. One end will be in its correct grid cell. The other end will be wherever its length dictates, as it was either horizontally or vertically laid down. Note that the 'other end' coordinates may lie outside the grid (see examples below). Wires always have integer lengths. Wire lengths lie in the range 1 .. 9 (inclusive). Input is terminated by a line with two zeroes.

Output

For each problem output one blank line. Then, for problems for which there is no unique solution, output "No solution". For problems with a unique solution you should output R rows. Each row will have C numbers, giving the heights of the wires in each column. These numbers will be output without spaces.

Sample Input

```
2 2
1 0 1 1
1 1 2 1
2 -1 2 2
1 2 3 2
1 3
1 1 1 3
1 1 1 2
1 2 1 3
0 0
```

Sample Output

```

12
13

No solution
```

Problem L **The God Delusion** **30 points**

You are God, building the universe. You're up to arranging the silicon crystals when you run into a problem. The silicon crystals are complaining that the aluminium impurities are stealing their electrons and are demanding their original electrons back. Luckily you've caught the problem early and the crystals you've been building are still small.

Unfortunately you can only move electrons by moving an electron from an atom with an electron to a connected neighbouring atom which is missing an electron. This can be a slow process--and as God you don't have a lot of time to spare--so you want to do it in the minimum number of moves.

For the purpose of this problem, you can think of the crystal lattice as a planar rectangular grid, with each atom connected to its 4 neighbours (up, down, left, right). You will be given a series of rectangular crystals with misplaced electrons, for which you have to find solutions. The atoms are numbered 0 to n-1 (n is the number of atoms in the lattice). Atom number i*width + j is at the position with row i and column j of the rectangular grid. The electrons are also from 1 to n-1. Your task is to move electrons to their corresponding (same number) atoms, with atom 0 ending up without an electron.

Input

Input will consist of a number of test cases. The first line of a test case will contain 2 integers h w (2 <= h,w <= 5, h*w <= 10). The next h lines will each contain w integers, identifying the electron at that location on the crystal lattice. A 0 represents the missing electron.

"0 0" on a line by itself indicates the end of the input.

Output:

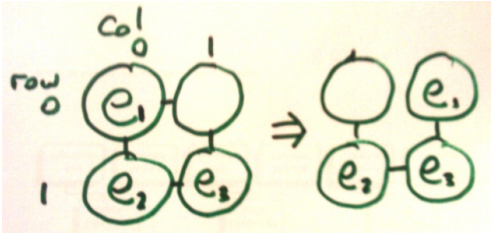
For each test case, output on one line the minimum number of steps required to return all electrons to their corresponding atoms.

Sample Input

```
2 2
1 0
2 3
2 3
1 2 5
3 4 0
0 0
```

Sample Output

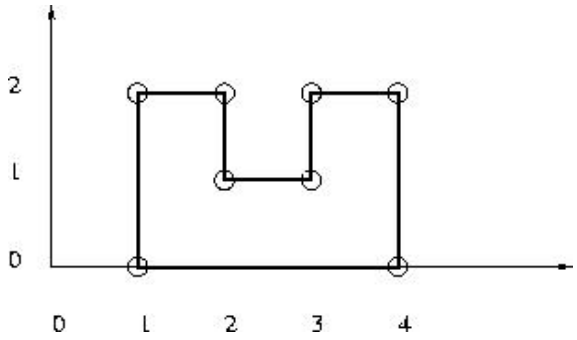
```
1
3
```



Problem M **Connecting Dots** **100 points**

(Adapted from Polygon Construction, an old University of Waterloo problem)

The algorithm will construct an intersecting line – so is it possible to construct a closed rectilinear path visiting all nodes exactly once. Note – lines may cross, but not on nodes.



Given are n points with integer coordinates in the plane. Is it is possible to draw a closed rectilinear polyline with the given points as vertices? In a rectilinear polyline there are at least 4 vertices and every edge is either horizontal or vertical; each vertex is an endpoint of exactly one horizontal edge and one vertical edge. The polyline connects all of the vertices – ie: it cannot be composed of two or more separate components. Being closed means that, starting from any vertex and following the polyline, we will arrive back at our starting vertex. It is permissible for the polyline to cross itself (eg: to form a figure eight), possibly many times, but a crossing may not occur at a vertex.

Input

The first line of input is an integer giving the number of cases that follow. The input of each case starts with an integer $4 \le n \le 100000$ giving the number of points for this test case. It is followed by n pairs of integers specifying the x and y coordinates of the points for this case, each on their own line and separated by a single space.

Output

The output should contain one line for each case on input. Each line should contain one integer number giving the length of the rectilinear polyline passing through the given points when it exists; otherwise, it should contain -1.

[Turn over for sample data]

Sample Input

```
1
8
1 2
1 0
2 1
2 2
3 2
3 1
4 0
4 2
```

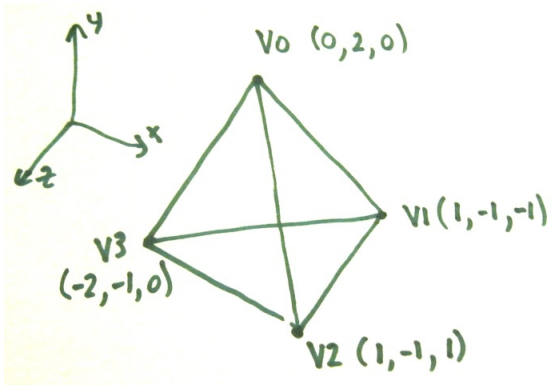
Sample Output

```
12
```

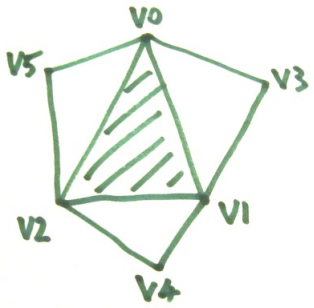
Problem N **Adjacent Edges** **100 points**

Triangle adjacency is an issue that arises in Computer graphics. Modelling packages may output 3D models as a list of triangles. For example the tetrahedron shown in the sketch has 4 triangle shaped faces. Each triangle is described by listing the positions of its three corners (they are listed in clockwise order from the viewpoint of someone just outside the solid shape. A simple file format is shown in the Sample Input section below. The first line holds the number of triangles. The next three lines hold the x, y, z coordinates for the three corners of the first triangle, and so on

This is satisfactory for many purposes, but some graphics algorithms need to know which triangles are adjacent – ie: share edges. Your task here is to work this out for given models. The process of working out adjacent triangles also allows us to check that all triangle edges are properly adjacent to one other edge – this is a way of checking that a model is a fully enclosed shell, without gaps or holes.



In general, if you consider any face in a model, it can have up to three adjacent triangles, with each of which it shares one edge. Each edge in a well formed model occurs in exactly two triangles and the vertices of the edge occur in reverse order as you travel clockwise around each triangle. If we flatten a typical triangle and show its adjacent triangles we see something like this. If we know the centre triangle, we need only one extra vertex for each adjacent triangle. Knowing V0, V1 and V2; knowledge of V3 gives us the first adjacent triangle. The output format for this problem is based on this idea. For each triangle we output V0, V1, V2, V3, V4 and V5. In situations where there an adjacent triangle is missing, we output an X for the corresponding vertex.



Input

The input consists of a number of 3D models for which you need to compute triangle adjacency. The first line for each problem is a single integer T in the range 1 to 400000, being the number of triangles in the model. For each triangle three lines follow – giving a total of 3T lines. Each line holds three floating point values, being the x, y, z coordinates of a vertex, separated by a comma and a space. Vertices of a triangle are in clockwise order. Note that, although the sample data uses only integer coordinates, the judging data will use floating point values, including some in exponent form. The input is organised so that different occurrences of the same vertex will have identical floating point values (expressed as identical strings in the input). Otherwise all vertices are distinct when represented as single precision (32 bit) floats. All coordinates are in the range -2 to 2 (inclusive). Input is terminated by a line with a zero value.

Output

Distinct vertices should be numbered in the order in which they first occur in the input. Numbers start with zero and then are 1, 2, 3, etc. These vertex numbers will be output to identify each vertex, rather than using the coordinate values as in the input. Output for each model should consist of one blank line, followed by one line per triangle (ie: T lines). Triangle lines will be output in the same order as triangles were read from the input. Each line should have an integer, being the number of the triangle (0, 1, 2, …), then a colon ':', a list of the vertex numbers of the sides of the triangle (in the same order as in the original input), and then the vertex numbers of the adjacent triangles' third vertices in the order shown in the sketch above. Any missing adjacent index values should be output as upper case X's. Values in the output should be separated by single space characters.

Sample Input

```
4
0, 2, 0
1, -1, -1
1, -1, 1
0, 2, 0
-2, -1, 0
1, -1, -1
0, 2, 0
1, -1, 1
-2, -1, 0
1, -1, -1
-2, -1, 0
1, -1, 1
4  2
0, 2, 0
1, -1, -1
1, -1, 1
0, 2, 0
-2, -1, 0
1, -1, -1
0
```

Sample Output

```

0: 0 1 2 3 3 3
1: 0 3 1 2 2 2
2: 0 2 3 1 1 1
3: 1 3 2 0 0 0

0: 0 1 2 3 X X
1: 0 3 1 X X 2
```