Contents

A Euro Efficiency

On January 1st 2002, The Netherlands, and several other European countries abandoned their national currency in favour of the Euro. This changed the ease of paying, and not just internationally.

A student buying a 68 guilder book before January 1st could pay for the book with one 50 guilder banknote and two 10 guilder banknotes, receiving two guilders in change. In short: $50+10+10-1-1=68$. Other ways of paying were: $50+25-5-1-1$, or $100-25-5-1-1$. Either way, there are always 5 units (banknotes or coins) involved in the payment process, and it could not be done with less than 5 units.
Buying a 68 Euro book is easier these days: $50+20-2=68$, so only 3 units are involved. This is no coincidence; in many other cases paying with euros is more efficient than paying with guilders. On average the Euro is more efficient. This has nothing to do, of course, with the value of the Euro, but with the units chosen. The units for guilders used to be: 1, 2½, 5, 10, 25, 50, whereas the units for the Euro are: 1, 2, 5, 10, 20, 50.

For this problem we restrict ourselves to amounts up to 100 cents. The Euro has coins with values 1, 2, 5, 10, 20, 50 eurocents. In paying an arbitrary amount in the range [1, 100] eurocents, on average 2.96 coins are involved, either as payment or as change. The Euro series is not optimal in this sense. With coins 1, 24, 34, 39, 46, 50 an amount of 68 cents can be paid using two coins. The average number of coins involved in paying an amount in the range [1, 100] is 2.52.

Calculations with the latter series are more complex, however. That is, mental calculations. These calculations could easily be programmed in any mobile phone, which nearly everybody carries around nowadays. Preparing for the future, a committee of the European Central Bank is studying the efficiency of series of coins, to find the most efficient series for amounts up to 100 eurocents. They need your help.

Problem

Write a program that, given a series of coins, calculates the average and maximum number of coins needed to pay any amount up to and including 100 cents. You may assume that both parties involved have sufficient numbers of any coin at their disposal.

Input

The first line of the input contains the number of test cases. Each test case is described by 6 different positive integers on a single line: the values of the coins, in ascending order. The first number is always 1. The last number is less than 100.

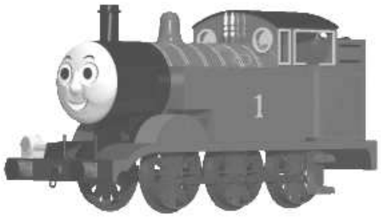
Output

For each test case the output is a single line containing first the average and then the maximum number of coins involved in paying an amount in the range [1, 100]. These values are separated by a space. As in the example, the average should always contain two digits behind the decimal point. The maximum is always an integer.

Example

input	output
3 1 2 5 10 20 50 1 24 34 39 46 50 1 2 3 7 19 72	2.96 5 2.52 3 2.80 4

B Markov Trains



Business is not going well for the Dutch Railway Company **NS**. Due to technical problems, they are forced to cancel many train services without advance notice. This is, of course, extremely frustrating for students who travel from home to school by train.

The worst thing about the whole situation is the randomness of the cancellations. Nobody knows in advance whether a train service will be cancelled; a cancellation is not announced until the official departure time. Since there is usually more than one possible route from home to school, people are often left with an '*if I had known this in advance I would have taken the other route*' sort of feeling.

Recently, the statistics department of the NS found a revolutionary solution to this problem. They noticed that some train services are cancelled more often than others. In order to help the passengers, they decided to publish this information. The new timetables will state not just the time of departure and arrival of each service, but also its *probability of cancellation*.

The travel-planner software from the NS, which normally finds the fastest route between stations, must be updated to find the route which gives the best chance of arriving in time. This helps passengers to avoid trains that are likely to cause problems, instead taking a slightly longer, but more reliable route to school.

Problem

Given the new timetables, a departure station and time, a destination station and a desired arrival time, find the route which gives the best chance of arriving at the destination in time.

A route in this case is simply an ordered list of stations visited by the passenger, starting with the departure station and ending with the destination. The passenger will stick to the route, each time taking the first possible train to the next station. If a train is cancelled, he will just wait for the next train to that station.

The chance of arriving in time is taken to be the probability that the passenger, when following the route as described above, arrives at the destination station before or at the desired arrival time. When calculating this probability, we assume that train services are cancelled independently of each other and according to the probabilities stated in the timetable.

Input

The first line of the input contains a single positive integer indicating the number of runs. For each run, the input is as follows:

- A line with a single positive integer n, the number of trains in the timetable ($n \leq 100$).
- n lines describing the timetable. Each line describes one train, stating its departure station x, the time of departure t_x, its destination station y ($x \neq y$), the time of arrival t_y ($t_x < t_y$) and its probability of cancellation p.
  Stations are identified by capital letters in the range `'A'` ... `'L'`. Times are in the format `hh:mm` with $00 \leq$ `hh` < 24 and $00 \leq$ `mm` < 60. The probability p is a decimal real number with $0.0 \leq p < 1.0$. Input elements are separated by spaces.
- A line with a departure station a, earliest departure time t_a, destination station b ($a \neq b$) and desired arrival time t_b ($t_a < t_b$). Station identifiers and times are like those in the timetable.

Output

The output consists of two lines for each run. The first line of each run contains the best possible route for the passenger as a list of station identifiers separated by spaces. The second line contains the probability that the passenger, when following the given route, arrives on time. The probability must be formatted as a decimal real number with exactly one digit before the decimal point, and exactly 4 digits after. The usual rules for rounding apply: round up if the next digit would be ≥ 5, otherwise round down.

Notes

- When changing trains at an intermediate station, the earliest possible departure time is one minute after the time of arrival.
- All times are on the same day; the journey does not cross midnight.
- It never happens that two or more trains depart from the same station at the same time to the same destination station.
- The input is such that there is a unique route with maximum probability.
- The passenger will stick to his route, always taking the *first available* train to the next station. If a train is cancelled he will wait for the next train to that station. He will never try to be smart by taking faster trains or different routes.

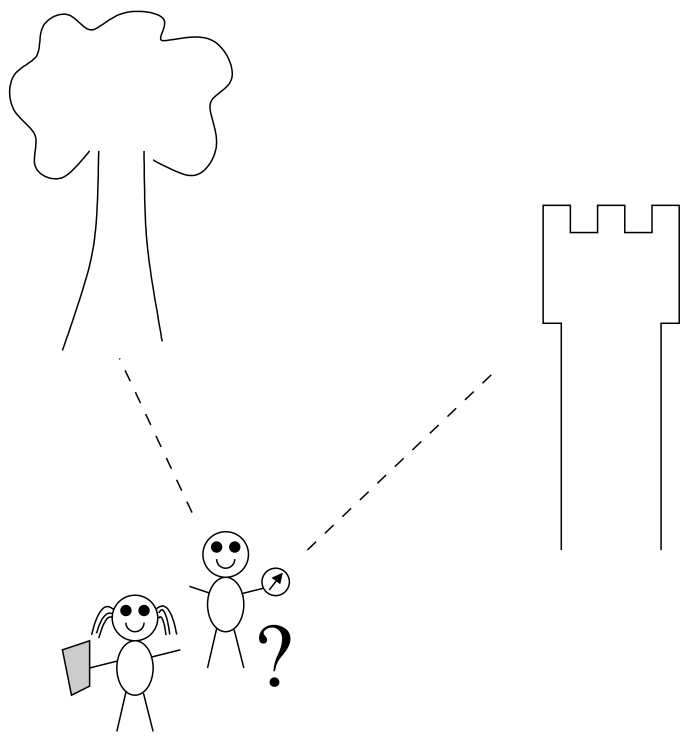
Example

input	output
2 3 A 12:00 B 12:15 0.1 A 12:10 B 12:14 0.23 A 12:20 B 12:30 0.456 A 12:00 B 12:30 4 A 12:00 B 12:15 0.1 A 12:05 B 12:13 0.15 B 12:20 C 12:35 0.12 A 12:15 C 12:33 0.4 A 12:00 C 13:00	A B 0.9895 A B C 0.8668

C Hansel and Grethel

On a warm summer afternoon, Hansel and Grethel are walking together in the fields. It is getting late and, to be honest, they are lost. Grethel is a little scared, still vividly remembering the last time they got lost in the forest. That time, an evil witch had locked them inside a house built of gingerbread and sugar! But Hansel can reassure her: this time they are well prepared. Hansel has taken a map and a compass with him!

Hansel picks two clearly outstanding features in the landscape, and uses the compass to measure the direction towards both objects. Grethel locates the objects on the map, and writes down the corresponding map coordinates. Based on this information, they will be able to accurately determine their own position on the map.



Problem

The coordinates of two marker objects, and the direction (angle from the North) towards these objects are known. Write a program which uses this data to calculate the coordinates of Hansel and Grethel's current location.

Input

The first line of the input contains one positive number: the number of situations in which a position must be determined. Following are two lines per situation, describing the two marker objects. Each marker object is described by a line containing three integer numbers:

- the x-coordinate of the object on the map ($0 \leq x \leq 100$);
  the x-axis runs West-to-East on the map, with increasing values towards the East.
- the y-coordinate of the object on the map ($0 \leq y \leq 100$);
  the y-axis runs South-to-North on the map, with increasing values towards the North.
- the direction d of the object in degrees ($0 \leq d < 360$);
  with $0° =$ North, $90° =$ East, $180° =$ South, and so on.

To keep the position calculations accurate, Hansel makes sure that the directions of the two objects are not exactly equal, and do not differ by exactly 180°.

Output

One line per situation, containing the result of the position calculation: two numbers, separated by a space, each having exactly 4 digits after the decimal point. These numbers represent the x and y coordinates of the position of Hansel and Grethel ($0 \leq x, y \leq 100$). Round the numbers as usual: up if the next digit would be ≥ 5, down otherwise.

Example

<table>
<tr><th>input</th><th>output</th></tr>
<tr><td><pre>2
30 50 90
20 40 180
30 40 96
20 20 150</pre></td><td><pre>20.0000 50.0000
7.0610 42.4110</pre></td></tr>
</table>

D Floors

The new highway promised a better and faster connection between A and B and a considerable reduction of congestion. Unfortunately there was an obstacle: the old mansion. This conflict was soon resolved in favor of the highway.

Shortly before the demolition of the mansion was about to start, a lover of old mansions found out that the colorfully tiled floors in the mansion were designed by the famous painter Mondriaan, and therefore had great cultural value. They should be saved. They should be removed from the mansion before the mansion will be demolished.

A floor moving expert was hired to accomplish the job. He decided to cut each floor into smaller pieces, in order to make it more tractable. He possessed a fine floor cutting tool which enabled him to cut a rectangular piece of floor into two smaller rectangular pieces, cutting parallel to one of the sides. Of course the cutting should be between tiles; cutting through a tile was not an option. This way the floor in Figure 1 could easily be cut into 9 tiles. The floor in Figure 2 however can not be cut into smaller pieces. The floor in Figure 3 can be cut into six pieces, but one of the parts will consist of several tiles.

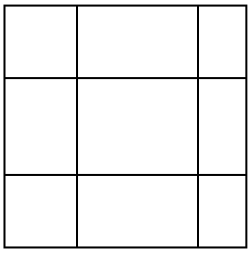


Figure 1

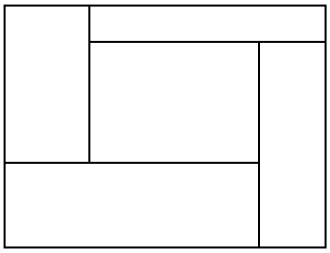


Figure 2

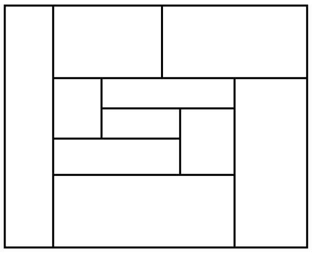


Figure 3

While preparing for the job, the floor moving expert was anxious to know how large the remaining pieces could be: would they be heavy, very heavy or extremely heavy? What kind of floor lifting tool should be hired? Because the floors have a fixed thickness, and a fixed density, the weight of a piece of floor only depends upon its area.

Problem

Given a rectangular floor covered with rectangular tiles, find the area of the largest piece, after the floor is cut into the smallest possible pieces. The words smallest and largest refer to the area of the pieces. Cutting through a tile is not allowed. A cut through a rectangle is always parallel to one of the sides, and through the full length (or width) of the rectangle.

Input

The input contains several floors. The first line of the input gives the number of floors.

Each floor is described in several lines. The first line contains two positive integers: the length and width of the floor, in millimeters. A floor is at most 40 000 mm long or wide. The next line contains a single number: the number t of tiles ($1 \leq t \leq 100$). The following t lines each contain the description of a tile. A tile is given as four integers:
xl yl xh yh
where (xl, yl) are the coordinates of the lower left corner of the tile, and (xh, yh) are the coordinates of the upper rightmost corner of the tile. A tile always has a positive area. The order of the coordinates of the floor and those of the tile coincide, of course.

You may assume that the tiles are mutually disjoint, and cover the floor, the whole floor, and nothing but the floor.

Output

For each test case (each floor) the output contains a number, on a single line: the area of the largest piece of floor (in square millimeters), after cutting the floor into the smallest pieces possible, with the given restrictions.

Example

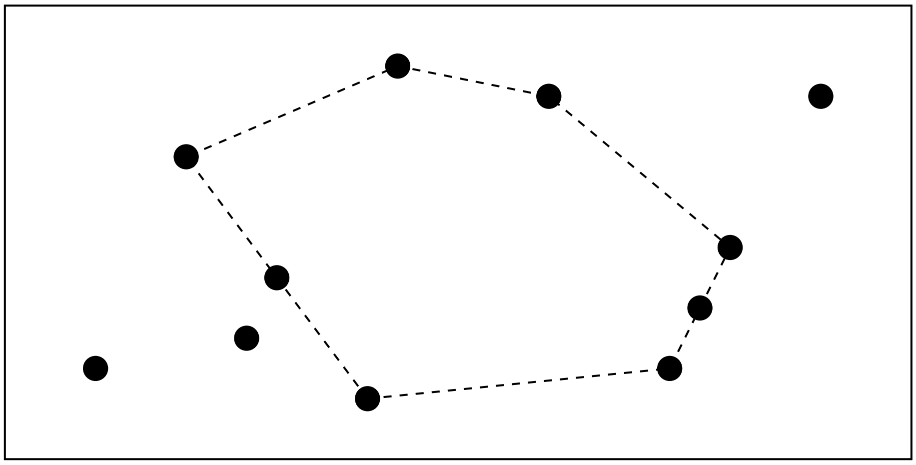
input	**output**

`2`
`300 400`
`3`
`0 0 300 200`
`0 200 200 400`
`200 200 300 400`
`300 300`
`5`
`0 0 200 100`
`200 0 300 200`
`0 100 100 300`
`100 200 300 300`
`100 100 200 200`
	`60000`
	`90000`

E The Picnic

The annual picnic of the Zeron company will take place tomorrow. This year they have agreed on the Gloomwood park as the place to be. The girl responsible for the arrangement, Lilith, thinks it would be nice if everyone is able to watch everyone else during the occasion. From geometry class she remembers that a region in the plane with the property that a straight line between any two points in the region, lies entirely in the region, is called convex. So that is what she is looking for. Unfortunately, this seems hard to fulfil, since Gloomwood has many opaque obstacles, such as large trees, rocks, and so on.

Owing to the fact that the staff of the Zeron company is pretty large, Lilith has a rather intricate problem to solve: finding a location to hold them all. Therefore, some of her friends help her to draw a map of the whereabouts of the largest obstacles. To mark out the place, she will use a ribbon stretched around the obstacles on the circumference of the chosen region. The opaque obstacles should be thought of as points of zero extension.



The Gloomwood park from above with black dots representing obstacles. The picnic area is the region whose circumference is dashed.

Input

The first line of the input contains a single positive integer n, specifying the number of test scenarios to follow. Each test scenario begins with a line containing an integer m, the number of obstacles in the park ($2 < m < 100$). The next line contains the coordinates of the m obstacles, in the order $x_1\ y_1\ x_2\ y_2\ x_3\ y_3 \ldots$. All coordinates are integers in the range [0, 1000]. Each scenario has at least three obstacles that are not on a straight line, and no two obstacles have the same coordinates.

Output

For each test scenario, one line of output should be generated, stating the area with one decimal of the largest convex polygon having obstacles as corners, but no enclosed obstacles.

Example

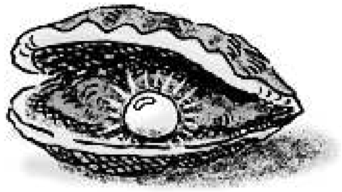
input

```
1
11
3 3 8 4 12 2 22 3 23 5 24 7 27 12 18 12 13 13 6 10 9 6
```

output

```
129.0
```

F Pearls



In Pearlania everybody is fond of pearls. One company, called *The Royal Pearl*, produces a lot of jewelry with pearls in it. *The Royal Pearl* has its name because it delivers to the royal family of Pearlania. But it also produces bracelets and necklaces for ordinary people. Of course the quality of the pearls for these people is much lower then the quality of pearls for the royal family. In Pearlania pearls are separated into 100 different quality classes. A quality class is identified by the price for one single pearl in that quality class. This price is unique for that quality class and the price is always higher then the price for a pearl in a lower quality class.

Every month the stock manager of *The Royal Pearl* prepares a list with the number of pearls needed in each quality class. The pearls are bought on the local pearl market. Each quality class has its own price per pearl, but for every complete deal in a certain quality class one has to pay an extra amount of money equal to ten pearls in that class. This is to prevent tourists from buying just one pearl.

Also *The Royal Pearl* is suffering from the slow-down of the global economy. Therefore the company needs to be more efficient. The CFO (chief financial officer) has discovered that he can sometimes save money by buying pearls in a higher quality class than is actually needed. No customer will blame *The Royal Pearl* for putting better pearls in the bracelets, as long as the prices remain the same.

For example 5 pearls are needed in the 10 Euro category and 100 pearls are needed in the 20 Euro category. That will normally cost: $(5+10)\times 10+(100+10)\times 20 = 2350$ Euro.
Buying all 105 pearls in the 20 Euro category only costs: $(5+100+10)\times 20 = 2300$ Euro.

The problem is that it requires a lot of computing work before the CFO knows how many pearls can best be bought in a higher quality class. You are asked to help *The Royal Pearl* with a computer program.

Problem

Given a list with the number of pearls and the price per pearl in different quality classes, give the lowest possible price needed to buy everything on the list. Pearls can be bought in the requested, or in a higher quality class, but not in a lower one.

Input

The first line of the input contains the number of test cases. Each test case starts with a line containing the number of categories c ($1 \leq c \leq 100$). Then, c lines follow, each with two numbers a_i and p_i. The first of these numbers is the number of pearls a_i needed in a class ($1 \leq a_i \leq 1000$). The second number is the price per pearl p_i in that class ($1 \leq p_i \leq 1000$). The qualities of the classes (and so the prices) are given in ascending order. All numbers in the input are integers.

Output

For each test case a single line containing a single number: the lowest possible price needed to buy everything on the list.

Example

input	output
2 2 100 1 100 2 3 1 10 1 11 100 12	330 1344

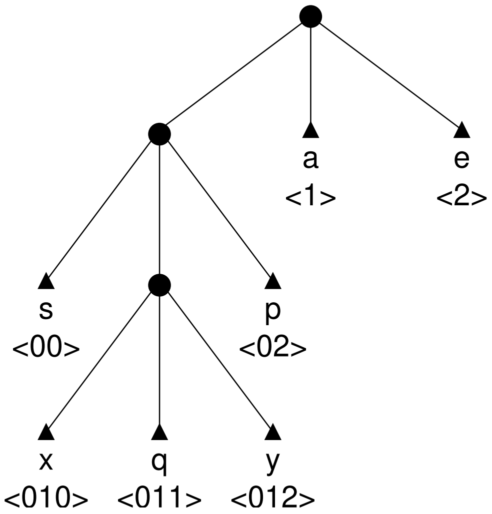
G Huffman Trees

A relatively simple method for compressing data works by creating a so-called *Huffman tree* for a file and using it to compress and decompress the data it contains. For most applications, binary Huffman trees are used (i.e., each node is either a leaf or has exactly two sub-nodes). One can, however, construct Huffman trees with an arbitrary number of sub-trees (i.e, ternary or, in general, N-ary trees).

A Huffman tree for a file containing Z different characters has Z leaves. The path from the root to a leaf that represents a certain character determines its encoding; each step towards the leaf determines an encoding character (which can be 0, 1, ..., $(N-1)$). By placing often-occurring characters closer to the root, and less often occurring characters further away from the root, the desirable compression is achieved. Strictly speaking, such a tree is a Huffman tree only if the resulting encoding takes the minimal number of N-ary symbols to encode the complete file.

For this problem, we only consider trees where each node is either an internal node, or a leaf encoding a character; there are no *dangling leaves* that do not encode for a character.

The figure below shows a sample ternary Huffman tree; the characters ’a’ and ’e’ are encoded using one ternary symbol; the less frequently occurring characters ’s’ and ’p’ are encoded using two ternary symbols; and the very rare symbols ’x’, ’q’ and ’y’ are encoded using three ternary symbols each.



Sample ternary Huffman tree

Of course, if we want to decompress a list of N-ary symbols later on, it is important to know which tree was used to compress the data. This can be done in several ways. In this problem we use the following method: the stream of data will be preceded by a header consisting of the encoded versions of the Z characters occurring in the original file, in lexicographical order.

Problem

Given the number of source symbols Z, a value N denoting the 'arity' of the Huffman tree, and a header, give the mapping from source symbols to encoded symbols.

Input

The input starts with a single integer T on a separate line, denoting the number of test cases that follow. Next, for each of the T test cases, three lines follow:

- The number of different characters in the file ($2 \leq Z \leq 20$);
- The number N, denoting the 'arity' of the Huffman tree ($2 \leq N \leq 10$);
- A string representing the header of the received message; each character will be a digit in the range [0, $(N-1)$]. This string will contain less than 200 characters.

Note: Although rare, it is possible for a header to have multiple interpretations (e.g., the header '`010011101100`' with $Z = 5$ and $N = 2$). It is guaranteed that *all cases in the test input have a unique solution*.

Output

For each of the T test-cases, print Z lines giving the encoded version of each of the Z characters in the testcase in ascending order. Use the format `original->encoding`, where *original* is a decimal value in the range [0, $(Z-1)$], and *encoding* is the string of encoding digits for this symbol (each digit is ≥ 0 and $< N$).

Example

input	output
3	0->10
3	1->0
2	2->11
10011	0->00
4	1->011
2	2->1
000111010	3->010
19	0->0
10	1->1
01234678950515253545556575859	2->2
	3->3
	4->4
	5->6
	6->7
	7->8
	8->9
	9->50
	10->51
	11->52
	12->53
	13->54
	14->55
	15->56
	16->57
	17->58
	18->59

H Input

In a recent programming contest, one of the problems was about tiling floors with rectangular tiles. The input specification reads like this:

> The input contains several floors. The first line of the input gives the number of floors. Each floor is described in several lines. The first line contains two positive integers: the length and width of the floor, in millimeters. A floor is at most 40 000 mm long or wide. The next line contains a single number: the number t of tiles ($1 \leq t \leq 100$). The following t lines each contain the description of a tile. A tile is given as four integers:
> *xl yl xh yh*
> where (*xl*, *yl*) are the coordinates of the lower left corner of the tile, and (*xh*, *yh*) are the coordinates of the upper rightmost corner of the tile. A tile always has a positive area. The order of the coordinates of the floor and those of the tile coincide, of course.
> You may assume that the tiles are mutually disjoint, and cover the floor, the whole floor, and nothing but the floor.

The last line of this specification raised some problems. Not for the contestants, but for the judges. Some of the test cases consist of many tiles. How can we be sure that our input file meets this condition? What we need is a checking program that verifies this condition.

Problem

Given an input file in the above format, find out for each floor whether the tiles
1. are disjoint,
2. do not lie outside the floor,
3. do cover the floor.

Input

The input contains several floors. The first line of the input gives the number of floors. Each floor is described in several lines. The first line contains two positive integers: the length and width of the floor, in millimeters. A floor is at most 40 000 mm long or wide. The next line contains a single number: the number t of tiles ($1 \leq t \leq 100$). The following t lines each contain the description of a tile. A tile is given as four integers:
xl yl xh yh
where (*xl*, *yl*) are the coordinates of the lower left corner of the tile, and (*xh*, *yh*) are the coordinates of the upper rightmost corner of the tile. A tile always has a positive area. The order of the coordinates of the floor and those of the tile coincide, of course.

Output

For each floor the output contains a single line, containing one of the following words:

`NONDISJOINT`	if overlapping tiles occur;
`NONCONTAINED`	if no overlapping tiles occur, but some tiles go outside the floor;
`NONCOVERING`	if no overlapping tiles occur, and no tiles go outside the floor, but some parts of the floor are not covered;
`OK`	if none of these is true.

Example

input	output
4 4 3 2 0 0 2 2 1 1 5 5 4 3 2 0 0 2 2 -2 2 5 5 4 3 2 0 0 2 2 2 0 4 2 4 3 3 0 0 2 2 2 0 4 2 0 2 4 3	NONDISJOINT NONCONTAINED NONCOVERING OK