Problem A. Chasing the Cheetahs

Source file name:	cheetahs.c, cheetahs.cpp, cheetahs.java, cheetahs.py
Input:	`Standard`
Output:	`Standard`

A National Geographic film crew is visiting the ZOO this week. They are creating a documentary about animal speed and they would like to film one or more cheetahs running at full pace. A solitary running cheetah has been filmed successfully many times. Therefore, the crew is also after a much more spectacular feat: As many as possible cheetahs sprinting on parallel tracks filmed together in one shot.

“No, that is impossible,” said the director. “We cannot squeeze those animals into some sort of a start box, as you probably imagine, and then open the box and make them run all at once. It is clearly too dangerous and unpredictable. No.”

“Then let us make more boxes and open some of them earlier and some of them later,” said one of the filmmakers. “Could that work?”

“And if we open the boxes with the slower cheetahs a bit earlier then after a while the faster ones will be overtaking the slower ones and that would be a great shot,” pointed out another filmmaker. “We would like to see the whole pack rather short with the last animals close the leading ones. As close as possible and at least for a moment.”

It was a long and difficult discussion which ensued, but in the end the circumstances of the experiment were agreed upon.

You are given the start time and the speed of each cheetah. The length of the pack, which is defined as the distance between the first and the last cheetah in the pack, might be different at different moments. Find the minimum length of the pack during the run, where all cheetahs must be running. You may also suppose that the track is so long that the minimum length of the pack happens at least a moment before the first cheetah reaches the finish line.

All start boxes will be so close that you may consider them to be in the same place. The k-th cheetah will be released from its start box at the given time t_k. at the same distance form the finish line. The k-th cheetah is expected to run the whole distance at constant speed v_k.

Input

There are more test cases. Each case occupies more lines. The first line of a case contains the number of cheetahs N ($1 \leq N \leq 100000$). Next, there are N lines, each line contains two integers t_k, v_k separated by spaces and representing the start time and velocity of the k-th cheetah ($1 \leq k \leq N$). All input values t_k and r_k are positive and less than 10^5. The input is terminated by a line containing zero.

Output

For each test case, print a single line with one floating point number L specifying the minimum length of the running pack. Your answer should not differ from the correct answer by more than 10^{-2}. The length of the pack is the distance between the first and the last animal in the pack. The length can be measured at any time $T \geq max\ (t_k,\ k = 1, \ldots, N)$. We suppose that each cheetah is running at a constant speed for all the time from the start and also at its moment of release from the start box.

Example

Input	Output
2	0.000
1 1	9999700002.000
1 1	0.500
2	
1 99999	
99999 99999	
3	
1 1	
3 2	
4 3	
0	

Problem B. Falcon Dive

Source file name:	dive.c, dive.cpp, dive.java, dive.py
Input:	`Standard`
Output:	`Standard`

“Our high speed camera failed at the most inappropriate moment,” said the director of the ZOO. “This sequence with the falcon hurtling towards the ground at 250 km/h is absolutely stunning. I had hopes that we could use the last frame as a promotion picture, it would look great with the autumn trees in the background. But the falcon is too high, even in this very last frame caught by the camera before it broke.”

“Cut out the falcon from the picture in Photoshop and just move it downwards,” said the falconer. “It’s a routine photo manipulation.”

“That would be unnatural,” objected the director. “We cannot show the public such obviously doctored pictures.”

“On the contrary, that would be quite natural,” replied the falconer. “Look, the falcon in such speed does not change its orientation so much, its shape in the picture remains virtually the same in a few consecutive frames. So if you move it down artificially it would still be a very good approximation of the natural situation which really occurred during the filming.”

After some hesitation, the director agreed with the proposition.

You are given two last frames of the camera with the silhouette of the falcon in both frames. The background in the frames is identical, only the silhouette of the falcon is at a different position in both frames. The falcon is moving at a constant speed and the time between consecutive camera frames is also constant. Your task is to reconstruct the missing next frame in which the position of the falcon silhouette is changed according to its speed and to the speed of the camera. The background in the new frame should be the same as the background in the previous two frames.

Input

There are more test cases. Each test case starts with a line containing two integers M, N $(2 \le M, N \le 1000)$ and a printable ASCII character C enclosed in single quotes. The values on the line are separated by spaces. Next, there are M lines, one empty line, and other M lines. The first M lines represent the first frame, the last M lines represent the second frame. Each nonempty line contains string of exactly N printable ASCII characters. Each character represents one pixel of the original frame. Each frame contains a complete silhouette of the falcon. In both frames all silhouette pixels are represented by the character C and all pixels which do not belong to the silhouette are represented by characters other than C. The pixels of the silhouettes in both frames do not overlap even partially, in other words, no coordinates of a pixel of the silhouette in the first frame are the same as the coordinates of any pixel of the silhouette in the second frame. The shapes of the silhouettes in both frames are identical. The silhouette in any frame can be shifted by some number of pixels horizontally and/or vertically so that its position exactly matches the position of the silhouette in the other frame. The silhouettes do not rotate. For various technical reasons the silhouette image might not be connected, it may comprise of more disconnected regions in the frame.

A printable ASCII character is an element of the subset of ASCII characters starting with the exclamation mark character (‘!’, ASCII code 33 in decimal) and ending with the tilde character (‘∼’, ASCII code 126 in decimal).

There is a blank line between successive cases. The input is terminated by a line containing “0 0 ’ ’ ”.

Output

For each test case, print a picture frame consisting of M lines with N characters each. The frame should represent the result of exact extrapolation of the falcon’s movement based on the two input frames. If the silhouette image in the second input frame is shifted horizontally and vertically by some number of pixels relatively to the first input frame then the silhouette image in the result frame should be shifted horizontally and vertically by the same number of pixels relatively to the second frame. It is possible that the falcon's silhouette might appear in the frame picture only partially or it may not appear there at all. Print one empty line after each case.

Example

Input

```
2 2 'X'
X^
--

.X
--

3 12 'A'
ABABABABABAC
BABABABABABB
ABABABABABAB

BABABABABABA
BBABABABABAB
BABABABABABA

6 26 '>'
../||\........00......\|/.
>//||\\.....000000....-0-.
/>>>|\\\....000000..../|\.
..>||........0000.........
...||.........||..........
||||||||||||||||||||||||||

../||\.....>..00......\|/.
.//||\\.....>>>000....-0-.
///||\\\....0>0000..../|\.
...||........0000.........
...||.........||..........
||||||||||||||||||||||||||

0 0 ' '
```

Output

```
.^
--

BBABABABABAC
BBBABABABABA
BBABABABABAB

../||\........00......\>>>
.//||\\.....000000....-0>.
///||\\\....000000..../|\.
...||........0000.........
...||.........||..........
||||||||||||||||||||||||||
```

Problem C. The Fox and the Owl

Source file name:	fox.c, fox.cpp, fox.java, fox.py
Input:	`Standard`
Output:	`Standard`

Fox Mithra has finally learned the numbers and he is now familiar with the concept of 'one', 'two', 'three' and also even 'zero', 'minus one', 'minus two' and so on. Really, an achievement for such a small fox. He took the textbook and copied the integers from the book one by one from the smallest to the biggest on the wall of his enclosure in the ZOO.

"Look, there is something wrong with your sequence on the wall", said the owl who just landed on the branch above Mithra's head. "You should put 30 between 20 and 22, there."

"Why?"

"Because the importance of a number is judged by the sum of its digits. 30 is therefore less important than 22 and it is more important than 20. And obviously, 30 should be equally close to 20 and 22 because its sum of digits differs only by one from both 20 and 22."

"I see," replied Mithra, "you are really clever. Can you help me please to rearrange the sequence correctly? Each time I tell you a number N you will tell me the closest smaller number with sum of digits bigger by one than the sum of digits of N."

"With pleasure," nodded the owl majestically.

Your task is to imitate the owl's task. Given an integer N you have to find the biggest integer which is smaller than N with sum of digits bigger by one than the sum of digits of N.

Input

There are more test cases. Each case consists of one line containing a single integer N ($|N| \leq 10^{100000}$). The input is terminated by a line containing string "END" and no other symbols.

Output

For each test case print on a separate line the number demanded by fox Mithra.

Example

Input	Output
30 199 1000 1520 END	22 -299 200 1512

Problem D. Feeding the Herrings

Source file name:	herrings.c, herrings.cpp, herrings.java, herrings.py
Input:	`Standard`
Output:	`Standard`

Zookeper Willy is feeding herrings today. He is feeding them to seals, as they are their preferred food. There are three separate pools in which the seals live. The ZOO is a modern institution and it demands their employees to keep track of feeding habits of animals. There is a touchscreen installed at the seals pools and Willy has to enter the number of herrings which he is going to deposit into each of the three pools. Unfortunately, the screen is not working properly - in particular, it is impossible to enter the digit '3'.

Willy called the chief marine mammals zookeper and asked for help.

"That is OK," said the chief, "just distribute the herrings in such a way that the number of herrings which go into each pool does not contain the digit '3'."

"But there is a lower limit L on the number of herrings which have to be put into each pool," reacted Willy, "I might not be able to find a suitable division."

"You will be able to find a suitable division," assured him the chief, "considering the numbers of herrings in the bucket, there should be zillions of possible divisions."

"Well, exactly how many?" wondered Willy for himself.

You will be given the total number N of herrings which are to be deposited into the seals pools and the lower limit L on the number of herrings in each of the pools. Find out in how many ways might these N herrings be placed into the pools in such a way that the number of herrings in each pool does not contain digit '3' in its decimal representation. In this problem, we do not distinguish between individual herrings as they are all more or less of the same size and nutrition value. We do distinguish between the pools, though, because they are populated by different groups of seals. Also, we suppose that no herring can be divided into pieces.

Input

There are more test cases. Each case consists of a single line containing two integers N, L ($1 \leq N \leq 10^{10000}$, $1 \leq L \leq N/3$) separated by space and representing the number of herrings in the bucket and the lower limit on the number of herrings which have to be deposited in each of the pools. The input is terminated by a line with two zeros.

Output

For each test case print on a separate line the number of possible divisions of the herrings into the three given pools. Express the result modulo 12345647.

Example

Input	Output

`3 1`
`4 1`
`7 2`
`99999 1`
`0 0`
	`1`
	`3`
	`0`
	`9521331`

Problem E. Jumping Yoshi

Source file name:	jump.c, jump.cpp, jump.java, jump.py
Input:	`Standard`
Output:	`Standard`

Yoshi is a frog. He lives in the ZOO under a log which was specially transported there from a distant equatorial rainforest. The log is big and wet and attracts the flies and that is what makes Yoshi satisfied.

There is also a line of pebbles which runs through the wetland in front of the log. There are various dark spots on the pebbles and sometimes Yoshi likes to look at them and dream that they are not just spots but some really big flies instead.

Yesterday, Yoshi's friend camel Addawser came to see him and suggested to play a game.

"Do you see those spots on the pebbles?" asked Addawser. "I challenge you to start on the leftmost pebble and then do some jumps from a pebble to another pebble but with some restrictions. You can jump from a pebble to another one only if the sum of numbers of spots on both pebbles is equal to the distance between the pebbles. And you have to get to as distant pebble as possible."

"All right, but you know that I can count at most to twenty three and no more," hesitated Yoshi.

"No problem, I will help you with the bigger numbers," said Addawser.

"Then, it's easy," said Yoshi, positioning himself on the first pebble and looking inquisitively over the rest of the line. "Although, one might not be quite so sure, after all," he added after a while.

You are given the sequence of numbers of dark spots on the pebbles in the line. You are asked to find the most distant pebble which can be reached by a sequence of jumps. The first jump starts at the first pebble in the line and a jump between two pebbles is possible if and only if the sum of numbers of spots on both pebbles is equal to the distance between them. You may suppose that the line of pebbles is straight and that the distance between each two neighboring pebbles is exactly one frog distance unit.

Input

There are more test cases. Each case starts with a line containing one integer N ($1 \leq N \leq 1000000$) representing the number of pebbles. The second line contains a list of N integers. The order of the integers in the list is the same as the order of the pebbles in the wetland, each integer represents the number of spots on the corresponding pebble. No pebble contains more than 10^9 spots. Suppose that Addawser knows all different pairs of pebbles where Yoshi can perform a jump from one pebble to another one during his sequence of jumps. You are guaranteed that the number of those pairs of pebbles never exceeds 1000000.

The input is terminated by a line with one zero.

Output

For each test case, print a single line with one integer denoting the distance of the pebble which can be reached by successive jumps according to the given rules and which is the most distant from the first pebble.

Example

Input	Output

7
2 1 0 1 2 3 3
11
7 6 1 4 1 2 1 4 1 4 5
0
	5
	10

Problem F. Lunch Menu

Source file name:	lunch.c, lunch.cpp, lunch.java, lunch.py
Input:	`Standard`
Output:	`Standard`

Willy is the youngest zookeper employed in the ZOO. His income is not exactly a billionaire's one and he obviously has to plan his regular expenses carefully. Take for example his daily visit to the ZOO cantine. From the very beginning of his service in the ZOO, Willy decided that his daily lunch expense will not exceed a fixed limit L. And while his budget is limited he still wants to have a complete lunch: A soup, a main dish, a dessert, and a beverage. Moreover, to keep himself amused, he wants to enjoy each day a different menu, different from all other menus he had eaten in the previous days. Now, he wonders how many days will it take until he is forced to eat a lunch which is an exact copy of another lunch he had already eaten before.

You are given Willy's lunch price limit L and the prices of all soups, main dishes, desserts, and beverages in the cantine. Determine how many different lunches can be assembled provided that two lunches are different if they differ in at least one of the four given parts.

Input

There are more test cases. Each case starts with a line containing five integers L, S, M, D, B ($1 \leq L \leq 10^8$, $1 \leq S, M, D, B \leq 5000$) representing (in this order) the lunch price upper limit, the number of soups, the number of main dishes, the number of desserts and the number of beverages in the cantine. Each of the next four lines contains a list of prices. The first line contains the soups price list, the second line contains the main dishes price list, the third line contains the desserts price list, and the fourth line contains the beverages price list. All items in all lists are positive integers not exceeding 10^8. There is one empty line after each test case. The input is terminated by a line with five zeros.

Output

For each test case print on a separate line the number of different lunches which can be assembled from the cantine offer and have price not exceeding L. Please note that the value of the solution might not fit into 32-bit integer.

Example

Input	Output

```
11 3 1 1 1
4 5 6
3
2
1

10 4 5 4 2
3 2 5 7
1 1 8 4 2
3 5 2 1
2 3

0 0 0 0 0
```
	```
	2
	48
	```

Problem G. The Owl and the Fox

Source file name:	owl.c, owl.cpp, owl.java, owl.py
Input:	`Standard`
Output:	`Standard`

Fox Mithra has finally learned the numbers and he is now familiar with the concept of 'one', 'two', 'three' and also even 'zero', 'minus one', 'minus two' and so on. Really, an achievement for such a small fox. He took the textbook and copied the integers from the book one by one from the smallest to the biggest on the wall of his enclosure in the ZOO.

"Look, there is something wrong with your sequence on the wall", said the owl who just landed on the branch above Mithra's head. "You should put 30 between 20 and 22, there."

"Why?"

"Because the importance of a number is judged by the sum of its digits. 30 is therefore less important than 22 and it is more important than 20. And obviously, 30 should be equally close to 20 and 22 because its sum of digits differs only by one from both 20 and 22."

"I see," replied Mithra, "you are really clever. Can you help me please to rearrange the sequence correctly? Each time I tell you a number N you will tell me the closest smaller number with sum of digits less by one than the sum of digits of N."

"With pleasure," nodded the owl majestically.

Your task is to imitate the owl's task. Given an integer N you have to find the biggest integer which is smaller than N with sum of digits less by one than the sum of digits of N.

Input

There are more test cases. Each case consists of one line containing a single integer N ($1 \leq N \leq 100000$). The input is terminated by a line containing string "END" and no other symbols.

Output

For each test case print on a separate line the number demanded by fox Mithra.

Example

Input	Output
30 199 1000 1520 END	20 198 0 1510

Problem H. Plankton Food

Source file name:	plankton.c, plankton.cpp, plankton.java, plankton.py
Input:	`Standard`
Output:	`Standard`

Only few ZOOs can afford to cultivate all types of plankton food they need to feed to various sea vertebrates and invertebrates which live in their voluminous aquariums. Some ZOOs might be from time to time in a short supply of a particular kind of plankton food which is momentarily difficult to obtain. On the other hand, they also might store some reserves of other plankton food types which they do not need to spend immediately and those reserves might be huge.

The director is currently in a pressing need for a particular type of plankton food as the construction of a new aquarium with thermal vents is being finished. Fortunately, there are many colleagues in other ZOOs willing to help. They presented the director with various offers. After many lengthy calls, the director has built a comprehensive list of all available offers. Now it is obvious to him that he might need to trade in more steps to obtain the desired food type. In the first trade step, he would exchange the type A, of which there is plenty in his ZOO, for some other type B, which in fact he does not need at all but which can be exchanged with another ZOO for some amount of another type C food, and so on until the final exchange is made which brings in the appropriate food type.

Looking into the list the director is not sure whether a desired chain of exchanges indeed does exist. He calls in his economy vice-director and asks for help. The vice-director studies the list carefully and finally says:

"The amount of food the ZOOs are willing to give in exchange is sometimes bigger and sometimes smaller than the amount of food they would receive and that depends, of course, on the type of the food and generosity of a particular ZOO. I cannot say now if the desired chain of exchanges exists. If it exists, then maybe it would be possible to utilize the offers in such a way that we receive theoretically an unlimited supply of the desired food for only a small investment of the food we have. All these possibilities have to be checked."

You are presented with the list of offers of all other ZOOs. You have to decide if a sequence of exchanges might be organized in such a way that it brings in a theoretically unlimited supply of the needed type of the plankton food in exchange for a limited volume of the unnecessary food (provided that the sources in the other ZOOs are unlimited). We suppose that any exchange based on a particular trade offer might happen arbitrary number of times.

Input

There are more test cases. Each test case starts with a line containing four positive integers T, F, F_u, F_n. T represents the number of trade offers, F represents the number of types of plankton food. The types of food are labeled 1, 2, . . . , F. F_u is the label of the unnecessary food which the organizing ZOO is offering, F_n is the label of the necessary food which it wants to obtain by the trade. Next, there are T lines, each line represents a particular trade offer of a particular ZOO. The offer is expressed by three values F_r, F_g, U. F_r and F_g are labels of particular food types. The third number, U, is the natural logarithm (the use of logarithms in the ZOO is very popular because of the large range of sizes of the animals) of the amount of units of plankton food of type F_g which the offering ZOO is willing to give in exchange of receiving 1 unit of plankton food of type F_r. The value of U is a decimal number with at most three digits after the decimal point and with absolute value less than or equal to 1000. Note that the identification of the offering ZOO has been stripped away, as it is not essential for the solution of the problem. The value of T does not exceed 10000, the value of F does not exceed 5000.

The input is terminated by a line with four zeros.

Output

For each test case print on a separate line either "TRUE" or "FALSE". Print "TRUE" if and only if there is a way to organize the exchanges in such way that an investment of a limited volume of the unnecessary food type can result in theoretically unlimited supply of the necessary food type.

Example

Input	Output

3 3 1 3
1 2 0.001
2 3 -0.002
3 1 0.001
3 3 1 3
1 2 1.5
2 3 0.5
3 1 -1.999
6 5 1 5
1 2 -0.1
2 3 0.1
3 4 0.1
4 2 0.1
2 5 -0.1
1 5 0
0 0 0 0
	FALSE
	TRUE
	TRUE

Problem I. Hacking the Screen

Source file name:	screen.c, screen.cpp, screen.java, screen.py
Input:	`Standard`
Output:	`Standard`

The ZOO management had been wondering for a long time how to increase the number of children visitors to the ZOO. The solution was surprising and unexpected in many ways. They installed a huge screen past the entrance and started to display short quizzes on it. The child in the crowd who first shouts out the answer to the quiz question is granted one day free access to the ZOO. The screen became soon very popular and various types of quizzes are routinely shown there. One type of the quiz is the math quiz containing arithmetic operations on integers. The management worries that older siblings and friends of the children might develop a math quiz screen hacking strategy: Snap the screen with the phone, run the image recognition SW which extracts formulas from the image, evaluates them, and presents the solution to the phone holder who immediately shouts out the answer.

Your task is to assess the difficulty of producing the screen hacking software. To get a better feel of the problem you will first develop a simple toy model application. Your code will read the formula presented in the preprocessed form of ASCII art and evaluate it.

Input

There are multiple test cases. First line of each test case contains two integers R and C ($1 \leq R \leq 3$, $1 \leq C \leq 1000$). Each of the following R lines contains C characters. The whole matrix of $R \times C$ characters represents a single arithmetic formula written in ASCII art and generated by the following set of rules:

```
FORMULA -> COMPLEX | FORMULA + COMPLEX | FORMULA - COMPLEX

COMPLEX -> SQRT | FRACTION | TERM
             ______
   SQRT -> \/SIMPLE

           SIMPLE
FRACTION -> ======
           SIMPLE

 SIMPLE -> TERM | SIMPLE + TERM | SIMPLE - TERM

   TERM -> INTEGER | INTEGER * TERM

INTEGER -> 0 | 1 | 2 | 3 | ... | 999999 | 1000000
```

There are also a few additional specifications regarding the layout of the formula.

- The horizontal bar of each `SQRT` is made of one or more underscore symbols ('_', ascii decimal code 95) and it always occupies the uppermost line of the formula in the screen.
- When the formula occupies exactly two lines, then the first line contains only horizontal bars of all `SQRT` parts of the formula.
- When the formula occupies exactly three lines, then all `TERMs` and all arithmetic operation symbols which are not part of any `FRACTION` or `SQRT` occupy the second line of the formula in the screen.

- The length of the horizontal bar of `SQRT` is the same as the length of `SIMPLE` under the bar.
- The fraction bar in `FRACTION` consists of one or more equality signs, its length is equal to the maximum of the lengths of `SIMPLE` above the bar and `SIMPLE` below the bar.
- There is always exactly one space preceding and following each arithmetic operation symbol ($+$, $-$, $*$) on a particular line.
- The formula exactly fits in to the $R \times C$ matrix, there are no blank/empty columns in front of the whole formula or behind it.

The whole formula is evaluated according to the standard arithmetic rules. Namely: Each `FORMULA` and each `TERM` is evaluated from left to right. Each `SIMPLE` is also evaluated from left to right with the additional standard condition that the multiplication has higher priority than the addition/subtraction. Evaluation of `SQRT` and `FRACTION` is also standard. The value of any evaluated `FORMULA`, `COMPLEX`, `SQRT`, `FRACTION`, `SIMPLE` and `TERM` is an integer whose absolute value does not exceed 1000000.

There is one empty line after each test case. The input is terminated by a line with two zeros.

Output

For each test case print a separate line with the value V of the input formula.

Example

Input

```
1 13
1 + 2 * 3 - 4

2 16
  ---------
\/3 * 4 - 3 + 10

3 5
6 * 4
=====
 12

3 13
    22     --
3 - == - \/16
    11

0 0
```

Output

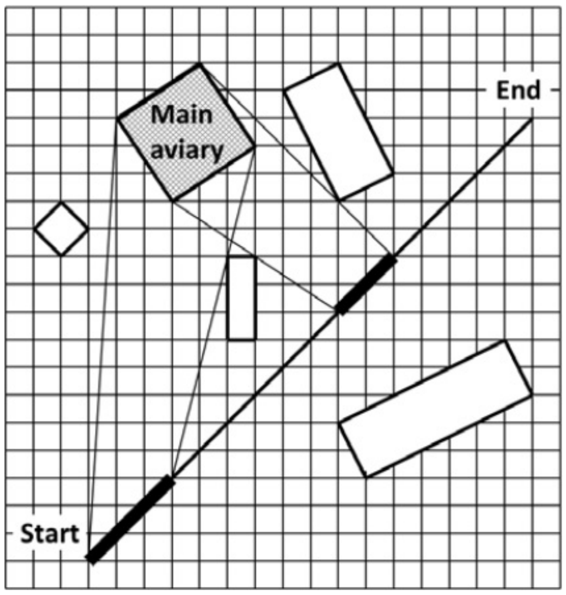
```
3
13
2
-3
```

Problem J. Visitors' Train

Source file name:	train.c, train.cpp, train.java, train.py
Input:	`Standard`
Output:	`Standard`

An aviary or a flight cage is a big cage for birds. An usual ZOO aviary typically measures tens of meters in diameter. In the aviaries, the birds can fly around and live in conditions imitating the conditions in the wild as closely as possible. At least in theory. There is one main big and spectacular aviary in the ZOO and some other less important ones.

The ZOO is planning to build a short straight electric train track to help visitors to move easily from one part of the ZOO to another. It has to be decided which of the free areas of the ZOO will the track run through. The director had noticed during his trips to other ZOOs that the visitors are more happy when they can take more photos of important ZOO structures. Now he wants to measure the quality of the planned railway by this parameter. The most important structure in the vicinity of the track will be the main aviary. The director worries that the main aviary might be obscured by the less important aviaries along the track and the visitors might be less happy. Help the director to assess the quality of the planned track.



You are given the coordinates of all aviaries. Also, you are given the coordinates of the start and the end of the planned railway track. Find the total length of the segments on the track from which the main aviary is visible and is not obscured, even not partially, by any other aviary. We suppose that the visitors can look out from the train in any direction.

Input

There are more test cases. Each case occupies more lines. The first line contains number N ($1 \le N \le 100$) of the aviaries. Next line contains the coordinates of the planned railway track in the format x_1 y_1 x_2 y_2 where $[x_1, y_1]$ and $[x_2, y_2]$ are the coordinates of the start and the end of the track. The track is considered to be infinitely thin in this representation. Next, there are N lines specifying the aviaries, each aviary is represented as a rectangle with nonzero area. Each of these lines specifies the coordinates of an aviary in the form x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4, where $[x_1, y_1]$, $[x_2, y_2]$, $[x_3, y_3]$, and $[x_4, y_4]$ are the coordinates of the aviary corners. The corners are presented in clockwise or anti-clockwise order. The main aviary is listed first. All coordinates are integers, their absolute value is less than 10000. You may assume that no aviary intersects or touches the track or another aviary. There is no blank line between consecutive test cases. The input is terminated by a line with one zero.

Output

For each test case print on a separate line the total length L of all segments of the planned track from which the main aviary is visible and it is not obscured, even not partially, by any other aviary. Your answer should not differ from the correct answer by more than 10^{-4}.

As shown in the third Sample Input, the main aviary is not considered obscured if *only its corners/edges are hidden.*

Example

Input	Output

```
5
3 1 17 15
6 14 4 17 7 19 9 16
2 12 1 13 2 14 3 13
8 9 8 12 9 12 9 9
12 14 10 18 12 19 14 15
12 6 18 9 19 7 13 4
1
0 0 0 2
4 -1 4 1 5 1 5 -1
2
0 0 0 1
4 0 4 2 5 2 5 0
2 0 3 0 3 -1 2 -1
2
0 0 0 1
4 0 4 2 5 2 5 0
2 0 3 0 3 1 2 1
0
```
	```
	7.07105
	2.00
	1.00
	0.00
	```

Problem K. How Sweet It Is!

Source file name:	sweet.c, sweet.cpp, sweet.java, sweet.py
Input:	`standard`
Output:	`standard`

Dr. Orooji's twins, Mack and Zack, love video games. We will assume that all games are $50. M/Z save all the money they get and, when they have $50 or more, they buy a game and say "Sweet!" out of happiness. If M/Z get a large amount of money at one time (e.g., on their birthday) and they can buy two or more games, they buy two or more games (as many as they can) and say "Totally Sweet!" since they are really in heaven! When M/Z buy game(s), they save the left-over money towards the next purchase.

Given the money (various amounts) M/Z receive, you are to write a program to tell Dr. O when Sweet or Totally Sweet is coming.

Input

Each input line contains a positive integer, indicating an amount M/Z are receiving. End-of-data is indicated with a zero.

Output

Print the input line numbers and the messages they generate. Follow the format illustrated in Example Output.

Example

Input	Output
10 20 30 10 90 10 10 30 0	Input #3: Sweet! Input #5: Totally Sweet! Input #8: Sweet!

Problem L. Lottery Coprimes

Source file name:	coprimes.c, coprimes.cpp, coprimes.java, coprimes.py
Input:	`standard`
Output:	`standard`

Lou lost the lottery last week, but he still plans to buy a ticket for this week's draw. He's also buying tickets for all his relatives. They are all mathematicians (who understand probability) and would never buy tickets for themselves. Lou insisted that they each choose their own numbers. When he looked at the numbers, it appeared as though all of his relatives had played a joke on him. They seemed to choose numbers by picking a pair of coprime integers, concatenating them, then splitting the digits up into the number spots on the lottery ticket.

Two integers are called coprimes, or relative primes, if they do not share any positive factors greater than 1. (That's the joke—they are "relative" primes from his relatives.)

These are the lottery numbers from one of Lou's math-loving relative's tickets.
The numbers 169 and 7203 are coprime.

Given a list of concatenated digits from a lottery ticket, determine whether this list can be split into two numbers which are coprimes. Note that the digits can not be reordered.

Input

The first line of input will contain only a single positive integer N, which is the number of lottery tickets to evaluate. Each of the next N input lines will contain 3 to 8 digits, representing a single ticket. Neither the first digit nor the last digit will ever be zero, and there will never be two consecutive zeroes. There will be no spaces or other characters on these lines, other than digits.

Output

For each ticket in the input, output `Ticket #T:`, where T is the ticket number (starting at 1). On the next line, output the two coprimes found by splitting the digits for that ticket. Separate the numbers by at least one space. If no coprimes are found, output the message `Not relative` instead, since the numbers were probably not picked by any of Lou's relatives. If there are multiple possible ways to split the digits into coprimes, use the one in which the first number is the lowest. If the split occurs before a zero digit, you must omit this leading zero when outputting the second number.

Follow the format shown in Example Output.

Example

Input	Output
4 47108 222 1697203 7203217	Ticket #1: 47 108 Ticket #2: Not relative Ticket #3: 1 697203 Ticket #4: 72 3217