Problem A
Calling Extraterrestrial Intelligence Again
Input: et.txt

A message from humans to extraterrestrial intelligence was sent through the Arecibo radio telescope in Puerto Rico on the afternoon of Saturday November 16, 1974. The message consisted of 1679 bits and was meant to be translated to a rectangular picture with 23×73 pixels. Since both 23 and 73 are prime numbers, 23×73 is the unique possible size of the translated rectangular picture each edge of which is longer than 1 pixel. Of course, there was no guarantee that the receivers would try to translate the message to a rectangular picture. Even if they would, they might put the pixels into the rectangle incorrectly. The senders of the Arecibo message were optimistic.

We are planning a similar project. Your task in the project is to find the most suitable width and height of the translated rectangular picture. The term "most suitable" is defined as follows. An integer m greater than 4 is given. A positive fraction a/b less than or equal to 1 is also given. The area of the picture should not be greater than m. Both of the width and the height of the translated picture should be prime numbers. The ratio of the width to the height should not be less than a/b nor greater than 1. You should maximize the area of the picture under these constraints.

In other words, you will receive an integer m and a fraction a/b. It holds that $m > 4$ and $0 < a/b \leq 1$. You should find the pair of prime numbers p, q such that $pq \leq m$ and $a/b \leq p/q \leq 1$, and furthermore, the product pq takes the maximum value among such pairs of two prime numbers. You should report p and q as the "most suitable" width and height of the translated picture.

Input

The input is a sequence of at most 2000 triplets of positive integers, delimited by a space character in between. Each line contains a single triplet. The sequence is followed by a triplet of zeros, `0 0 0`, which indicates the end of the input and should not be treated as data to be processed.

The integers of each input triplet are the integer m, the numerator a, and the denominator b described above, in this order. You may assume $4 < m \leq 100\,000$ and $1 \leq a \leq b \leq 1000$.

Output

The output is a sequence of pairs of positive integers. The i-th output pair corresponds to the i-th input triplet. The integers of each output pair are the width p and the height q described above, in this order.

Each output line contains a single pair. A space character is put between the integers as a delimiter. No other characters should appear in the output.

Sample Input

```
5 1 2
99999 999 999
1680 5 16
1970 1 1
2002 4 11
0 0 0
```

Output for the Sample Input

```
2 2
313 313
23 73
43 43
37 53
```

Problem B
Equals are Equals
Input: expressions.txt

Mr. Simpson got up with a slight feeling of tiredness. It was the start of another day of hard work. A bunch of papers were waiting for his inspection on his desk in his office. The papers contained his students' answers to questions in his Math class, but the answers looked as if they were just stains of ink.

His headache came from the "creativity" of his students. They provided him a variety of ways to answer each problem. He has his own answer to each problem, which is correct, of course, and the best from his aesthetic point of view.

Some of his students wrote algebraic expressions equivalent to the expected answer, but many of them look quite different from Mr. Simpson's answer in terms of their literal forms. Some wrote algebraic expressions not equivalent to his answer, but they look quite similar to it. Only a few of the students' answers were exactly the same as his.

It is his duty to check if each expression is mathematically equivalent to the answer he has prepared. This is to prevent expressions that are equivalent to his from being marked as "incorrect", even if they are not acceptable to his aesthetic moral.

He had now spent five days checking the expressions. Suddenly, he stood up and yelled, "I've had enough! I must call for help."

Your job is to write a program to help Mr. Simpson to judge if each answer is equivalent to the "correct" one. Algebraic expressions written on the papers are multi-variable polynomials over variable symbols $a, b, ..., z$ with integer coefficients, e.g., $(a + b^2)(a - b^2)$, $ax^2 + 2bx + c$ and $(x^2 + 5x + 4)(x^2 + 5x + 6) + 1$.

Mr. Simpson will input every answer expression as it is written on the papers; he promises you that an algebraic expression he inputs is a sequence of terms separated by additive operators '`+`' and '`-`', representing the sum of the terms with those operators, if any; a term is a juxtaposition of multiplicands, representing their product; and a multiplicand is either (a) a non-negative integer as a digit sequence in decimal, (b) a variable symbol (one of the lowercase letters '`a`' to '`z`'), possibly followed by a symbol '`^`' and a non-zero digit, which represents the power of that variable, or (c) a parenthesized algebraic expression, recursively. Note that the operator '`+`' or '`-`' appears only as a binary operator and not as a unary operator to specify the sign of its operand.

He says that he will put one or more space characters before an integer if it immediately follows another integer or a digit following the symbol '`^`'. He also says he may put spaces here and there in an expression as an attempt to make it readable, but he will never put a space between two consecutive digits of an integer. He remarks that the expressions are not so complicated, and that any expression, having its '`-`'s replaced with '`+`'s, if any, would have no variable raised to its 10th power, nor coefficient more than a billion, even if it is fully expanded into a form of a sum of products of coefficients and powered variables.

Input

The input to your program is a sequence of blocks of lines. A block consists of lines, each containing an expression, and a terminating line. After the last block, there is another terminating line. A terminating line is a line solely consisting of a period symbol.

The first expression of a block is one prepared by Mr. Simpson; all that follow in a block are answers by the students. An expression consists of lowercase letters, digits, operators '+', '-' and '^', parentheses '(' and ')', and spaces. A line containing an expression has no more than 80 characters.

Output

Your program should produce a line solely consisting of "yes" or "no" for each answer by the students corresponding to whether or not it is mathematically equivalent to the expected answer. Your program should produce a line solely containing a period symbol after each block.

Sample Input

```
a+b+c
(a+b)+c
a-(b-c)+2
.
4ab
(a - b)(0-b+a) - 1a ^ 2 - b ^ 2
2 b 2 a
.
108 a
2 2 3 3 3 a
4 a^1 27
.
.
```

Output for the Sample Input

```
yes
no
.
no
yes
.
yes
yes
.
```

Problem C
GIGA Universe Cup
Input: gucup.txt

Following FIFA World Cup, a larger competition called "GIGA Universe Cup" is taking place somewhere in our universe. Both FIFA World Cup and GIGA Universe Cup are two rounds competitions that consist of the first round, also known as "group league," and the second called "final tournament." In the first round, participating teams are divided into groups of four teams each. Each team in a group plays a match against each of the other teams in the same group. For example, let's say we have a group of the following four teams, "Engband, Swedon, Argontina, and Nigerua." They play the following six matches: Engband – Swedon, Engband – Argontina, Engband – Nigerua, Swedon – Argontina, Swedon – Nigerua, and Argontina – Nigerua.

The result of a single match is shown by the number of goals scored by each team, like "Engband 1 – 0 Argontina," which says Engband scored one goal whereas Argontina zero. Based on the result of a match, *points* are given to the two teams as follows and used to rank teams. If a team wins a match (i.e., scores more goals than the other), three points are given to it and zero to the other. If a match draws (i.e., the two teams score the same number of goals), one point is given to each.

The *goal difference* of a team in given matches is the total number of goals it scored minus the total number of goals its opponents scored in these matches. For example, if we have three matches "Swedon 1 – 2 Engband," "Swedon 3 – 4 Nigerua," and "Swedon 5 – 6 Argontina," then the goal difference of Swedon in these three matches is $(1 + 3 + 5) - (2 + 4 + 6) = -3$.

Given the results of all the six matches in a group, teams are ranked by the following criteria, listed in the order of priority (that is, we first apply (a) to determine the ranking, with ties broken by (b), with ties broken by (c), and so on).

(a) greater number of points in all the group matches;
(b) greater goal difference in all the group matches;
(c) greater number of goals scored in all the group matches.

If two or more teams are equal on the basis of the above three criteria, their place shall be determined by the following criteria, applied in this order:

(d) greater number of points obtained in the group matches between the teams concerned;
(e) greater goal difference resulting from the group matches between the teams concerned;
(f) greater number of goals scored in the group matches between the teams concerned;

If two or more teams are still equal, apply (d), (e), and (f) as necessary to each such group. Repeat this until those three rules to equal teams do not make any further resolution. Finally, teams that still remain equal are ordered by:

(g) drawing lots by the Organizing Committee for the GIGA Universe Cup.

The two teams coming first and second in each group qualify for the second round.

Your job is to write a program which, given the results of matches played so far in a group and one team specified in the group, calculates the probability that the specified team will qualify for the second round. You may assume each team has played exactly two matches and has one match to play. In total, four matches have been played and two matches are to be played.

Assume the probability that any team scores (exactly) p goals in any match is:

$$\frac{8!}{p!(8-p)!}\left(\frac{1}{4}\right)^{p}\left(\frac{3}{4}\right)^{8-p},$$

for $p \leq 8$, and zero for $p > 8$. Assume the lot in the step (g) is fair.

Input

The first line of the input is an integer, less than 1000, that indicates the number of subsequent records.

The rest of the input is the indicated number of records. A single record has the following format:

$\langle empty \rangle$	$\langle \_ \rangle$	$\langle team \rangle_1$	$\langle \_ \rangle$	$\langle team \rangle_2$	$\langle \_ \rangle$	$\langle team \rangle_3$	$\langle \_ \rangle$	$\langle team \rangle_4$
$\langle team \rangle_1$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle m \rangle_{12}$	$\langle \_ \rangle$	$\langle m \rangle_{13}$	$\langle \_ \rangle$	$\langle m \rangle_{14}$
$\langle team \rangle_2$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle m \rangle_{23}$	$\langle \_ \rangle$	$\langle m \rangle_{24}$
$\langle team \rangle_3$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle m \rangle_{34}$
$\langle team \rangle_4$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle empty \rangle$	$\langle \_ \rangle$	$\langle empty \rangle$

In the above, $\langle \_ \rangle$ is a single underscore (`_`) and $\langle empty \rangle$ a sequence of exactly four underscores (`____`). Each of $\langle team \rangle_1, \cdots, \langle team \rangle_4$ is either an asterisk character (`*`) followed by exactly three uppercase letters (e.g., `*ENG`), or an underscore followed by exactly three uppercase letters (e.g., `_SWE`). The former indicates that it is the team you are asked to calculate the probability of the second round qualification for. You may assume exactly one of $\langle team \rangle_1, \cdots, \langle team \rangle_4$ is marked with an asterisk. Each $\langle m \rangle_{ij}$ $(1 \leq i < j \leq 4)$ is a match result between the $\langle team \rangle_i$ and $\langle team \rangle_j$. Each match result is either `__-_` (i.e., two underscores, hyphen, and another underscore) or of the form `_`x`-`y where each of x and y is a single digit (≤ 8). The former indicates that the corresponding match has not been played, whereas the latter that the result of the match was x goals by $\langle team \rangle_i$ and y goals by $\langle team \rangle_j$. Since each team has played exactly two matches, exactly two match results are in the former format.

Output

The output should consist of n lines where n is the number of records in the input. The ith line should show the probability that the designated team (marked with an asterisk) will qualify for the second round in the ith record.

Numbers should be printed with exactly seven digits after the decimal point. Each number should not contain an error greater than 10^{-7}.

Sample Input

```
5
_____*AAA__BBB__CCC__DDD
*AAA_______0-0__0-0___-_
_BBB_____________-___0-0
_CCC_________________0-0
_DDD____________________
______CHN__CRC__TUR_*BRA
_CHN_______0-2___-___0-4
_CRC____________1-1___-_
_TUR_________________1-2
*BRA____________________
______CMR_*KSA__GER__IRL
_CMR_______1-0___-___1-1
*KSA____________0-8___-_
_GER_________________1-1
_IRL____________________
______TUN__JPN_*BEL__RUS
_TUN________-___1-1__0-2
_JPN____________2-2__1-0
*BEL___________________-_
_RUS____________________
______MEX__CRO_*ECU__ITA
_MEX_______1-0__2-1___-_
_CRO_____________-___2-1
*ECU_________________0-2
_ITA____________________
```

Output for the Sample Input

```
0.5000000
1.0000000
0.0000000
0.3852746
0.0353304
```

Problem D
Life Line
Input: sai.txt

Let's play a new board game "Life Line".

The number of the players is greater than 1 and less than 10.

In this game, the board is a regular triangle in which many small regular triangles are arranged (See Figure 1). The edges of each small triangle are of the same length.

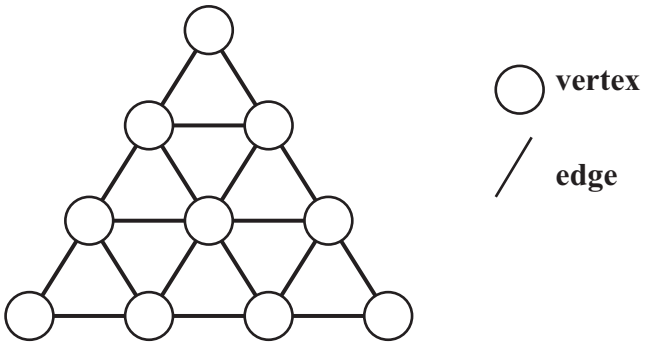


Figure 1: The board

The size of the board is expressed by the number of vertices on the bottom edge of the outer triangle. For example, the size of the board in Figure 1 is 4.

At the beginning of the game, each player is assigned his own identification number between 1 and 9, and is given some stones on which his identification number is written.

Each player puts his stone in turn on one of the "empty" vertices. An "empty vertex" is a vertex that has no stone on it.

When one player puts his stone on one of the vertices during his turn, some stones might be removed from the board. The player gains points which is equal to the number of the removed stones of others, but loses points which is equal to the number of the removed stones of himself. The points of a player for a single turn is the points he gained minus the points he lost in that turn.

The conditions for removing stones are as follows:

- The stones on the board are divided into groups. Each group contains a set of stones whose numbers are the same and placed adjacently. That is, if the same numbered stones are placed adjacently, they belong to the same group.

- If none of the stones in a group is adjacent to at least one "empty" vertex, all the stones in that group are removed from the board.

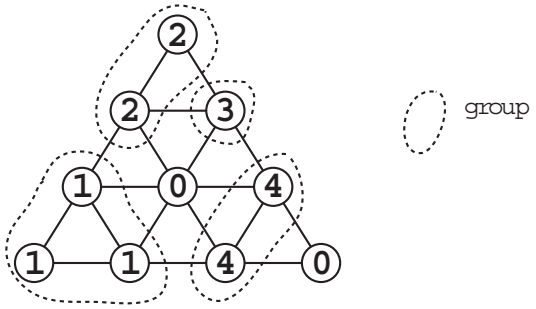


Figure 2: The groups of stones

Figure 2 shows an example of the groups of stones.

Suppose that the turn of the player '4' comes now. If he puts his stone on the vertex shown in Figure 3a, the conditions will be satisfied to remove some groups of stones (shadowed in Figure 3b). The player gains 6 points, because the 6 stones of others are removed from the board (See Figure 3c).

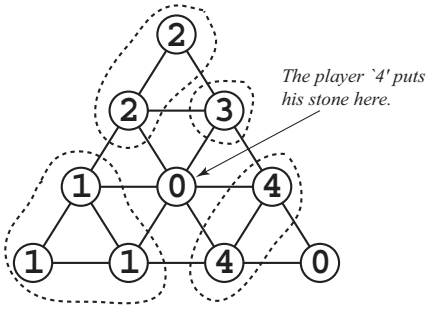


Figure 3a

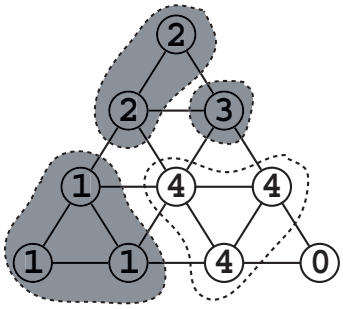


Figure 3b

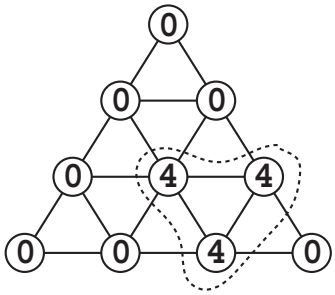


Figure 3c

As another example, suppose that the turn of the player '2' comes in Figure 2. If the player puts his stone on the vertex shown in Figure 4a, the conditions will be satisfied to remove some groups of stones (shadowed in Figure 4b). The player gains 4 points, because the 4 stones of others are removed. But, at the same time, he loses 3 points, because his 3 stones are removed. As the result, the player's points of this turn is $4 - 3 = 1$ (See Figure 4c).

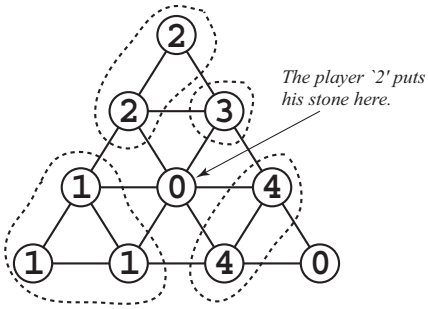


Figure 4a

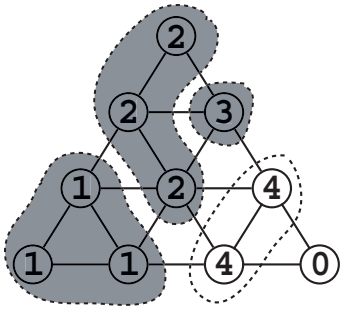


Figure 4b

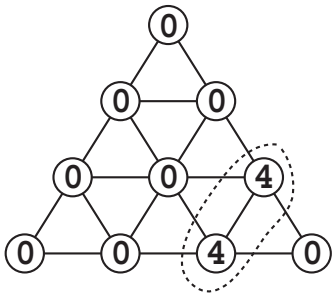


Figure 4c

When each player puts all of his stones on the board, the game is over. The total score of a player is the summation of the points of all of his turns.

Your job is to write a program that tells you the maximum points a player can get (i.e., the points he gains − the points he loses) in his current turn.

Input

The input consists of multiple data. Each data represents the state of the board of the game still in progress.

The format of each data is as follows.

$$
\begin{array}{ccccccc}
N & C & & & & & \\
 & & & S_{1,1} & & & \\
 & & S_{2,1} & & S_{2,2} & & \\
 & S_{3,1} & & S_{3,2} & & S_{3,3} & \\
 & & & \cdots & & & \\
S_{N,1} & & & \cdots & & & S_{N,N}
\end{array}
$$

N is the size of the board ($3 \leq N \leq 10$).

C is the identification number of the player whose turn comes now ($1 \leq C \leq 9$). That is, your program must calculate his points in this turn.

$S_{i,j}$ is the state of the vertex on the board ($0 \leq S_{i,j} \leq 9$). If the value of $S_{i,j}$ is positive, it means that there is the stone numbered by $S_{i,j}$ there. If the value of $S_{i,j}$ is 0, it means that the vertex is "empty".

Two zeros in a line, i.e., 0 0, represents the end of the input.

Output

For each data, the maximum points the player can get in the turn should be output, each in a separate line.

Sample Input

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

```
   2
  2 3
 3 0 4
1 1 4 0
4 1
   1
  1 1
 1 1 1
1 1 1 0
4 2
   1
  1 1
 1 1 1
1 1 1 0
4 1
   0
  2 2
 5 0 7
0 5 7 0
4 2
   0
  0 3
 1 0 4
0 1 0 4
4 3
   0
  3 3
 3 2 3
0 3 0 3
4 2
   0
  3 3
 3 2 3
0 3 0 3
6 1
     1
    1 2
   1 1 0
  6 7 6 8
 0 7 6 8 2
6 6 7 2 2 0
5 9
    0
   0 0
  0 0 0
 0 0 0 0
0 0 0 0 0
```

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

Output for the Sample Input

```
6
5
1
-10
8
-1
0
1
-1
5
0
5
```

Problem E
Map of Ninja House
Input: ninja.txt

An old document says that a Ninja House in Kanazawa City was in fact a defensive fortress, which was designed like a maze. Its rooms were connected by hidden doors in a complicated manner, so that any invader would become lost. Each room has at least two doors.

The Ninja House can be modeled by a graph, as shown in Figure 1. A circle represents a room. Each line connecting two circles represents a door between two rooms.

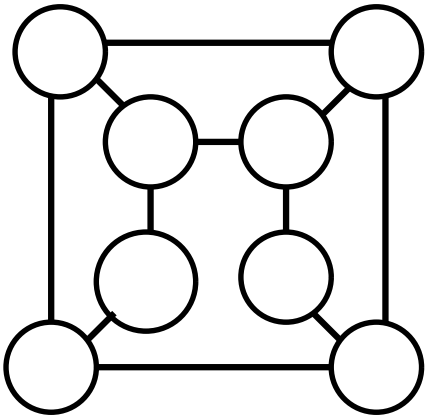


Figure 1. Graph Model of Ninja House.

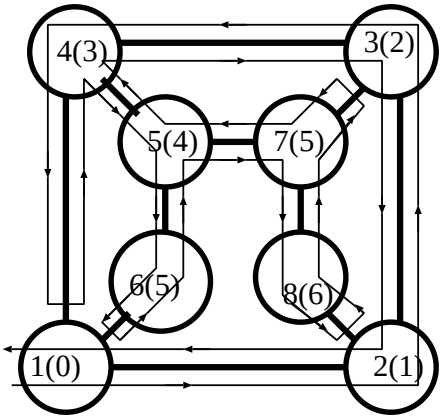


Figure 2. Ninja House exploration.

I decided to draw a map, since no map was available. Your mission is to help me draw a map from the record of my exploration.

I started exploring by entering a single entrance that was open to the outside. The path I walked is schematically shown in Figure 2, by a line with arrows. The rules for moving between rooms are described below.

> After entering a room, I first open the rightmost door and move to the next room. However, if the next room has already been visited, I close the door without entering, and open the next rightmost door, and so on. When I have inspected all the doors of a room, I go back through the door I used to enter the room.

I have a counter with me to memorize the *distance* from the first room. The counter is incremented when I enter a new room, and decremented when I go back from a room. In Figure 2, each number in parentheses is the value of the counter when I have entered the room, i.e., the distance from the first room. In contrast, the numbers not in parentheses represent the order of my visit.

I take a record of my exploration. Every time I open a door, I record a single number, according to the following rules.

1. If the opposite side of the door is a new room, I record the number of doors in that room, which is a positive number.
2. If it is an already visited room, say R, I record *"the distance of R from the first room"* minus *"the distance of the current room from the first room"*, which is a negative number.

In the example shown in Figure 2, as the first room has three doors connecting other rooms, I initially record "3". Then when I move to the second, third, and fourth rooms, which all have three doors, I append "3 3 3" to the record. When I skip the entry from the fourth room to the first room, the distance difference "−3" (minus three) will be appended, and so on. So, when I finish this exploration, its record is a sequence of numbers "3 3 3 3 −3 3 2 −5 3 2 −5 −3".

There are several dozens of Ninja Houses in the city. Given a sequence of numbers for each of these houses, you should produce a graph for each house.

Input

The first line of the input is a single integer n, indicating the number of records of Ninja Houses I have visited. You can assume that n is less than 100. Each of the following n records consists of numbers recorded on one exploration and a zero as a terminator. Each record consists of one or more lines whose lengths are less than 1000 characters. Each number is delimited by a space or a newline. You can assume that the number of rooms for each Ninja House is less than 100, and the number of doors in each room is less than 40.

Output

For each Ninja House of m rooms, the output should consist of m lines. The i-th line of each such m lines should look as follows:

$$i \;\; r_1 \;\; r_2 \;\; \cdots \;\; r_{k_i}$$

where $r_1, \cdots, r_{k_i}$ should be rooms adjoining room i, and k_i should be the number of doors in room i. Numbers should be separated by exactly one space character. The rooms should be numbered from 1 in visited order. $r_1, r_2, \cdots, r_{k_i}$ should be in ascending order. Note that the room i may be connected to another room through more than one door. In this case, that room number should appear in $r_1, \cdots, r_{k_i}$ as many times as it is connected by different doors.

Sample Input

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

Output for the Sample Input

```
1 2 4 6
2 1 3 8
3 2 4 7
4 1 3 5
5 4 6 7
6 1 5
7 3 5 8
8 2 7
1 2 3 4
2 1 3 3 4 4
3 1 2 2 4
4 1 2 2 3
```

Problem F
Shredding Company
Input: paper.txt

You have just been put in charge of developing a new shredder for the Shredding Company. Although a "normal" shredder would just shred sheets of paper into little pieces so that the contents would become unreadable, this new shredder needs to have the following unusual basic characteristics.

- The shredder takes as input a *target number* and a sheet of paper with a number written on it.
- It shreds (or cuts) the sheet into pieces each of which has one or more digits on it.
- The sum of the numbers written on each piece is the closest possible number to the target number, without going over it.

For example, suppose that the target number is *50*, and the sheet of paper has the number *12346*. The shredder would cut the sheet into four pieces, where one piece has *1*, another has *2*, the third has *34*, and the fourth has *6*. This is because their sum 43 ($= 1 + 2 + 34 + 6$) is closest to the target number *50* of all possible combinations without going over 50. For example, a combination where the pieces are *1*, *23*, *4*, and *6* is not valid, because the sum of this combination 34 ($= 1 + 23 + 4 + 6$) is less than the above combination's 43. The combination of *12*, *34*, and *6* is not valid either, because the sum 52 ($= 12 + 34 + 6$) is greater than the target number of 50.

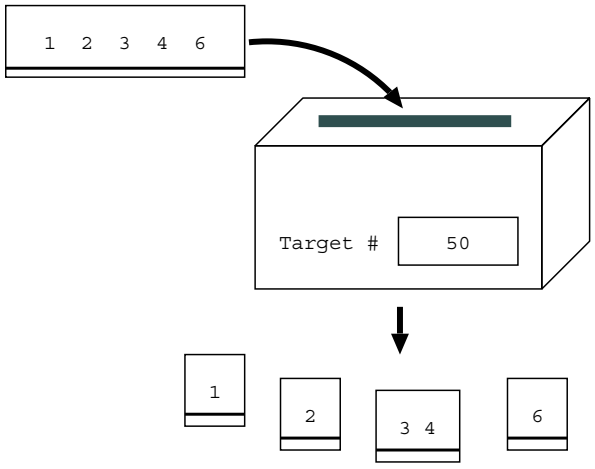


Figure 1. Shredding a sheet of paper having the number *12346* when the target number is *50*

There are also three special rules:

- If the target number is the same as the number on the sheet of paper, then the paper is not cut. For example, if the target number is *100* and the number on the sheet of paper is also *100*, then the paper is not cut.

- If it is not possible to make any combination whose sum is less than or equal to the target number, then *error* is printed on a display. For example, if the target number is *1* and the number on the sheet of paper is *123*, it is not possible to make any valid combination, as the combination with the smallest possible sum is *1*, *2*, *3*. The sum for this combination is 6, which is greater than the target number, and thus *error* is printed.
- If there is more than one possible combination where the sum is closest to the target number without going over it, then *rejected* is printed on a display. For example, if the target number is *15*, and the number on the sheet of paper is *111*, then there are two possible combinations with the highest possible sum of 12: (a) *1* and *11* and (b) *11* and *1*; thus *rejected* is printed.

In order to develop such a shredder, you have decided to first make a simple program that would simulate the above characteristics and rules. Given two numbers, where the first is the target number and the second is the number on the sheet of paper to be shredded, you need to figure out how the shredder should "cut up" the second number.

Input

The input consists of several test cases, each on one line, as follows:

t_1 num_1
t_2 num_2
$\cdots$
t_n num_n
0 0

Each test case consists of the following two positive integers, which are separated by one space: (1) the first integer (t_i above) is the target number; (2) the second integer (num_i above) is the number that is on the paper to be shredded.

Neither integers may have a 0 as the first digit, e.g., 123 is allowed but 0123 is not. You may assume that both integers are at most 6 digits in length. A line consisting of two zeros signals the end of the input.

Output

For each test case in the input, the corresponding output takes one of the following three types:

- sum $part_1$ $part_2$ $\cdots$
- rejected
- error

In the first type, $part_j$ and sum have the following meaning:

- Each $part_j$ is a number on one piece of shredded paper. The order of $part_j$ corresponds to the order of the original digits on the sheet of paper.

- sum is the sum of the numbers after being shredded, i.e., $sum = part_1 + part_2 + \cdots$.

Each number should be separated by one space.

The message `error` is printed if it is not possible to make any combination, and `rejected` if there is more than one possible combination.

No extra characters including spaces are allowed at the beginning of each line, nor at the end of each line.

Sample Input

```
50 12346
376 144139
927438 927438
18 3312
9 3142
25 1299
111 33333
103 862150
6 1104
0 0
```

Output for the Sample Input

```
43 1 2 34 6
283 144 139
927438 927438
18 3 3 12
error
21 1 2 9 9
rejected
103 86 2 15 0
rejected
```

Problem G
True Liars
Input: liar.txt

After having drifted about in a small boat for a couple of days, Akira Crusoe Maeda was finally cast ashore on a foggy island. Though he was exhausted and despaired, he was still fortunate to remember a legend of the foggy island, which he had heard from patriarchs in his childhood. This must be the island in the legend.

In the legend, two tribes have inhabited the island, one is divine and the other is devilish; once members of the divine tribe bless you, your future is bright and promising, and your soul will eventually go to Heaven; in contrast, once members of the devilish tribe curse you, your future is bleak and hopeless, and your soul will eventually fall down to Hell.

In order to prevent the worst-case scenario, Akira should distinguish the devilish from the divine. But how? They looked exactly alike and he could not distinguish one from the other solely by their appearances. He still had his last hope, however. The members of the divine tribe are truth-tellers, that is, they always tell the truth and those of the devilish tribe are liars, that is, they always tell a lie.

He asked some of them whether or not some are divine. They knew one another very much and always responded to him "faithfully" according to their individual natures (i.e., they always tell the truth or always a lie). He did not dare to ask any other forms of questions, since the legend says that a devilish member would curse a person forever when he did not like the question. He had another piece of useful information: the legend tells the populations of both tribes. These numbers in the legend are trustworthy since everyone living on this island is immortal and none have ever been born at least these millennia.

You are a good computer programmer and so requested to help Akira by writing a program that classifies the inhabitants according to their answers to his inquiries.

Input

The input consists of multiple data sets, each in the following format:

$$
\begin{array}{lll}
n & p_1 & p_2 \\
x_1 & y_1 & a_1 \\
x_2 & y_2 & a_2 \\
 & \cdots & \\
x_i & y_i & a_i \\
 & \cdots & \\
x_n & y_n & a_n
\end{array}
$$

The first line has three non-negative integers n, p_1, and p_2. n is the number of questions Akira asked. p_1 and p_2 are the populations of the divine and devilish tribes, respectively, in the legend. Each of the following n lines has two integers x_i, y_i and one word a_i. x_i and y_i are the identification numbers of inhabitants, each of which is between 1 and $p_1 + p_2$, inclusive. a_i is either `yes`, if the inhabitant x_i said that the inhabitant y_i was a member of the divine tribe, or `no`, otherwise. Note that x_i and y_i can be the same number since "are you a member of the divine tribe?" is a valid question. Note also that two lines may have the same x's and y's since Akira was very upset and might have asked the same question to the same one more than once.

You may assume that n is less than 1000 and that p_1 and p_2 are less than 300. A line with three zeros, i.e., `0 0 0`, represents the end of the input. You can assume that each data set is consistent and no contradictory answers are included.

Output

For each data set, if it includes sufficient information to classify all the inhabitants, print the identification numbers of all the divine ones in ascending order, one in a line. In addition, following the output numbers, print `end` in a line. Otherwise, i.e., if a given data set does not include sufficient information to identify all the divine members, print `no` in a line.

Sample Input

```
2 1 1
1 2 no
2 1 no
3 2 1
1 1 yes
2 2 yes
3 3 yes
2 2 1
1 2 yes
2 3 no
5 4 3
1 2 yes
1 3 no
4 5 yes
5 6 yes
6 7 no
0 0 0
```

Output for the Sample Input

```
no
no
```

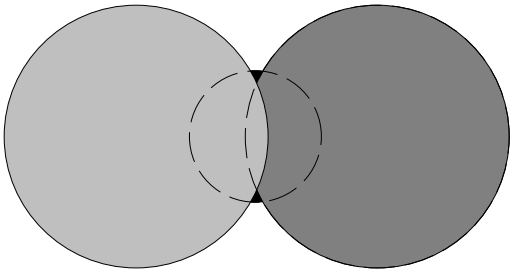
```
1
2
end
3
4
5
6
end
```

Problem H
Viva Confetti
Input: confetti.txt

Do you know *confetti*? They are small discs of colored paper, and people throw them around during parties or festivals. Since people throw lots of confetti, they may end up stacked one on another, so there may be hidden ones underneath.

A handful of various sized confetti have been dropped on a table. Given their positions and sizes, can you tell us how many of them you can see?

The following figure represents the disc configuration for the first sample input, where the bottom disc is still visible.



Java specific Submitted Java programs may not use packages whose names start by `java.awt` (you may use them for your own testing purposes).

Input

The input is composed of a number of configurations of the following form.

$$
\begin{array}{l}
n \\
x_1 \;\; y_1 \;\; r_1 \\
x_2 \;\; y_2 \;\; r_2 \\
\vdots \\
x_n \;\; y_n \;\; r_n
\end{array}
$$

The first line in a configuration is the number of discs in the configuration (a positive integer not more than 100), followed by one line descriptions of each disc: coordinates of its center and radius, expressed as real numbers in decimal notation, with up to 12 digits after the decimal point. The imprecision margin is $\pm 5 \times 10^{-13}$. That is, it is guaranteed that variations of less than $\pm 5 \times 10^{-13}$ on input values do not change which discs are visible. Coordinates of all points contained in discs are between -10 and 10.

Confetti are listed in their stacking order, x_1 y_1 r_1 being the bottom one and x_n y_n r_n the top one. You are observing from the top.

The end of the input is marked by a zero on a single line.

Output

For each configuration you should output the number of visible confetti on a single line.

Sample Input

```
3
0 0 0.5
-0.9 0 1.00000000001
0.9 0 1.00000000001
5
0 1 0.5
1 1 1.00000000001
0 2 1.00000000001
-1 1 1.00000000001
0 -0.00001 1.00000000001
5
0 1 0.5
1 1 1.00000000001
0 2 1.00000000001
-1 1 1.00000000001
0 0 1.00000000001
2
0 0 1.0000001
0 0 1
2
0 0 1
0.00000001 0 1
0
```

Output for the Sample Input

```
3
5
4
2
2
```