A. The mysterious X network

One of the reasons for which École polytechnique (nicknamed X for reasons to be explained during the debrieng talk) is so deeply rooted in French society is its famous network of camarades -former students of the same school-. When one camarade wants something (money, job, etc.), he can ask this network for help and support. In practice, this means that when s/he wants to reach some other camarade, not always of the same year, then surely he can find intermediate camarades to get to her/him.

Note that the camarade relationship is symmetric. Due to the magic of the X network, there is always a means to reach anybody. The program you have to write is supposed to help to minimize the number of these intermediate camarades.

Input

There are several cases, each case is simplied so as to obey the following format. The first line in the case is the number of camarades, say N , an integer $1 \leq N \leq 10^5$. Camarades are labeled from 0 to $N-1$. Follow N lines. Each line starts with the camarade label c , followed by the number of other camarades s/he knows, say n_c , followed by the labels of those n_c camarades. All these integers are separated by a single blank. It is assumed that n_c is always less than 100. The last line in the case is the label of the camarade seeking help (say c_1) followed by the label of the camarade he wants help from, say c_2 ($c_1 \neq c_2$). The file ends with a 0. Filename: a.in

Output

For each case your program should output one line with three integers separated by a blank: c_1 , c_2 and the minimal number of intermediate camarades to reach c_2.

Sample Input

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

Sample Output

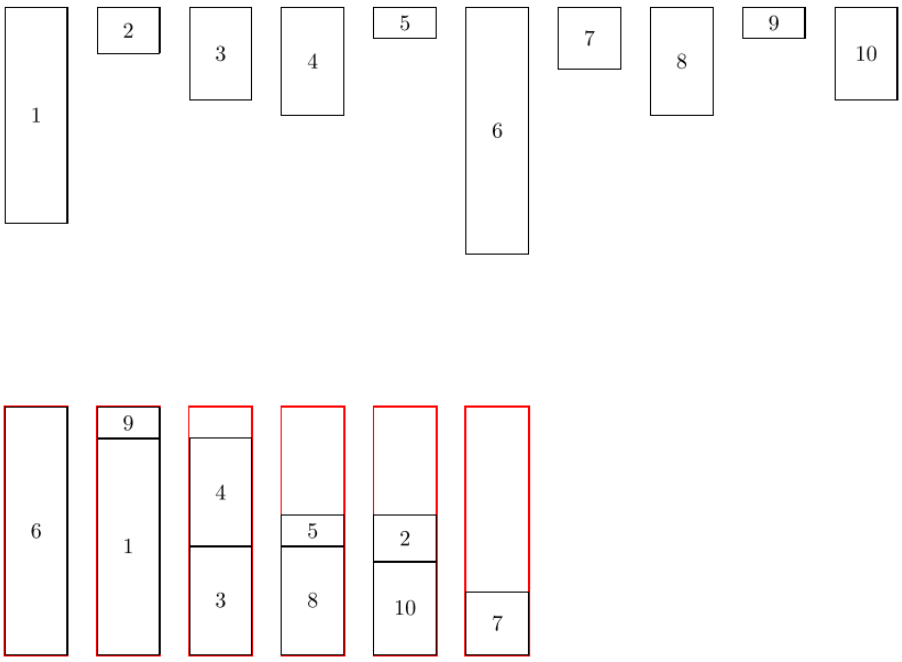
```
1  2  1
```

B. Bin Packing

A set of n 1-dimensional items have to be packed in identical bins. All bins have exactly the same length l and each item i has length $l_i \leq l$. We look for a minimal number of bins q such that

- each bin contains at most 2 items,
- each item is packed in one of the q bins,
- the sum of the lengths of the items packed in a bin does not exceed l.

You are requested, given the integer values $n, l, l_1, ..., l_n$, to compute the optimal number of bins q. The following figure shows a 10 items example whose fits in 6 bins.



Input

There are several test cases, The first line of the test case contains the number of items n ($1 \leq n \leq 10^5$). The second line contains one integer that corresponds to the bin length $l \leq 10000$. We then have n lines containing one integer value that represents the length of the items. The file ends with a 0. Filename: b.in

Output

For each case, your program has to write the minimal number of bins required to pack all items.

Sample Input

```
10
80
70
15
30
35
10
80
20
35
10
30
0
```

Sample Output

```
6
```

C. On storing clothes

In the laundry next to my flat, clothes are stored on coat hangers that are put on hooks fixed on a circular rail moved electrically by a computer. Hooks are numbered so that finding a cloth is easy. The rail moves in front of a mark.
We model the rail as an array of dimension n, referenced in a circular way, that is indices are to be considered *modulo* N. When a batch of n clothes must be stored, the launderer types the number n on a keyboard. The computer then looks for the first location of $n+2$ free hooks, from the current position of the rail to the right, yielding zone $k..k+n+1$ (all indices considered *modulo* N). Once this is done, the rail moves so that hook numbered $k+n+1$ arrives on the mark, and the launderer puts the n clothes on the hooks $k+1..k+n$. Hooks k and $k+n+1$ are not used to store clothes, but are used as "separators" between batches of clothes. The launderer then gives the ticket number k to the customer. Hooks used to hang the clothes of some customer are assigned to the customer, even during the actual cleaning of the batch of clothes.
When the customer comes back with the ticket number k, the launderer types k on the keyboard and the computer makes the rail moves so that the separating hook k of the corresponding batch is in front of the mark. The launderer takes the batch back (during this operation, the rail does not move) and gives it back to the customer. When a cloth is handed back to a customer, the corresponding hook is then free. Note that, when both its left and right neighbors are empty, a separating hook can be used for any purpose (either to hang a cloth or to become a separating hook again). The aim of the program is to model deposits and withdrawals of batches of clothes. We assume that at the beginning of the reading, the rail is empty and that hook 0 is in front of the mark. Only clothes that have been deposited can be withdrawn.

Input

There are several test cases. Each test case has the following format. The first line contains the number N of hooks, $(1 \leq N \leq 300)$. We then have the number l of lines in the le after the current one. Follow l lines with two dierent possible formats. The first one is:
D n
to deposit n clothes. The second one is:
W k
to indicate that clothes corresponding to ticket k must be withdrawn $(0 \leq k < N)$. The file ends with a 0.
Filename: c.in

Output

For each test case, when the customer makes a deposit of clothes, the program looks for an empty place for the whole batch. If this cannot be found, the programs output is:
No space left, please come back later.
If ticket k can be issued, the programs output is:
The launderer gives ticket k**.**
When ticket k is given back, the programs output is:
The launderer gives back batch k**.**
and all hooks used to hang the clothes of the corresponding customer are made free. Moreover, a separating hook of a batch that has been removed is also made free if both its right and left neighbors are free. Whenever hooks $h, ..., h+q$ become free, the program should output:
i **is freed.**
for all i between h and $h+q$.

Sample Input

```
22
5
D 1
D 3
W 0
D 3
D 11
0
```

Sample Output

```
The launderer gives ticket 0.
The launderer gives ticket 2.
The launderer gives back batch 0.
0 is freed.
1 is freed.
The launderer gives ticket 6.
The launderer gives ticket 10.
```

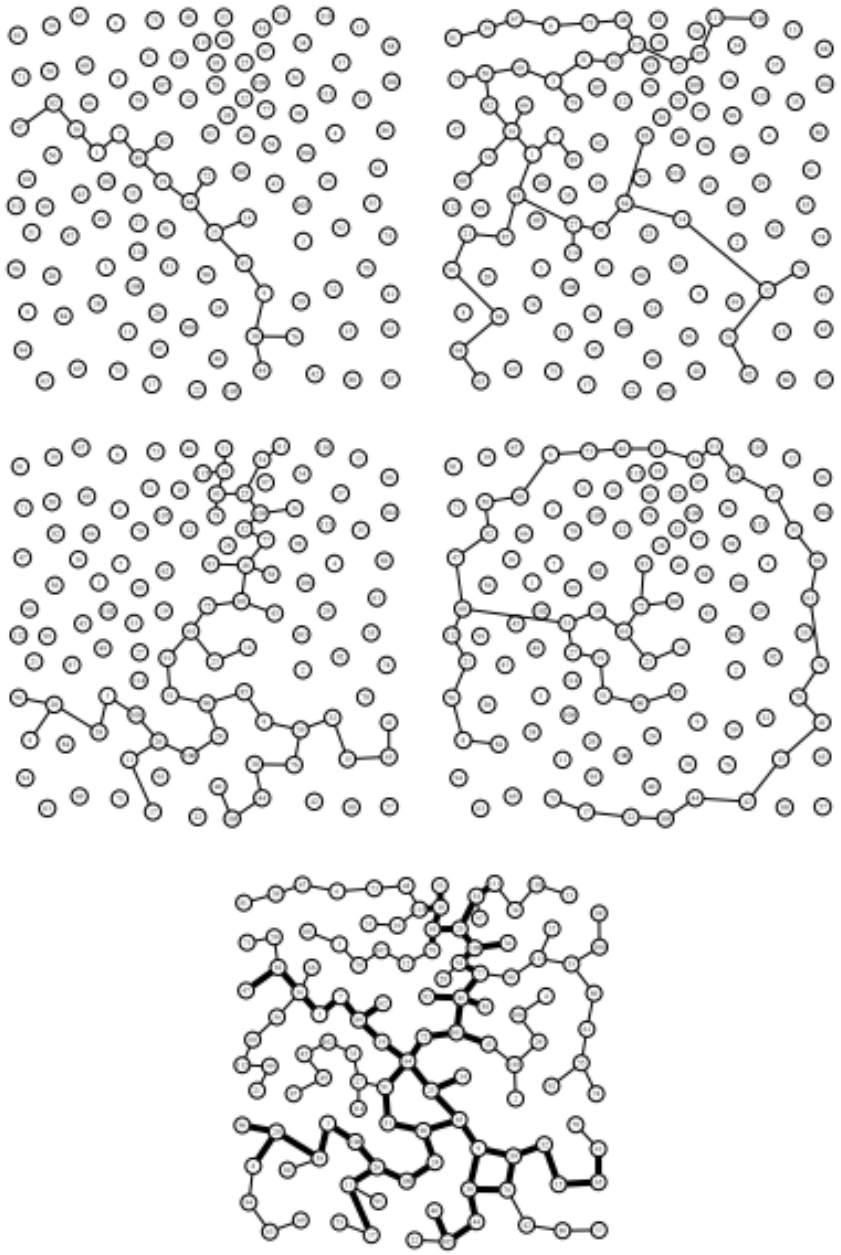
D. Buy or Build

World Wide Networks (WWN) is a leading company that operates large telecommunication networks. WWN would like to setup a new network in Borduria, a nice country that recently managed to get rid of its military dictator Kurvi-Tasch and which is now seeking for investments of international companies (for a complete description of Borduria, have a look to the following Tintin albums "King Ottokars Sceptre", "The Calculus Aair" and "Tintin and the Picaros"). You are requested to help WWN to decide how to setup its network for a minimal total cost. There are several local companies running small networks (called subnetworks in the following) that partially cover the n largest cities of Borduria. WWN would like to setup a network that connects all n cities. To achieve this, it can either build edges between cities from scratch or it can buy one or several subnetworks from local companies. You are requested to help WWN to decide how to setup its network for a minimal total cost.

- All n cities are located by their two-dimensional Cartesian coordinates.
- There are q existing subnetworks. If $q \geq 1$ then each subnetwork c ($1 \leq c \leq q$) is dened by a set of interconnected cities (the exact shape of a subnetwork is not relevant to our problem).
- A subnetwork c can be bought for a total cost w_c and it cannot be split (i.e., the network cannot be fractioned).
- To connect two cities that are not connected through the subnetworks bought, WWN has to build an edge whose cost is exactly the square of the Euclidean distance between the cities.

You have to decide which existing networks you buy and which edges you setup so that the total cost is minimal. Note that the number of existing networks is always very small (typically smaller than 8).
A 115 Cities Instance. Consider a 115 cities instance of the problem with 4 subnetworks (the 4 first graphs in Figure 1). As mentioned earlier the exact shape of a subnetwork is not relevant still, to keep figures easy to read, we have assumed an arbitrary tree like structure for each subnetworks. The bottom network in Figure 1 corresponds to the solution in which the first and the third networks have been bought. Thin edges correspond to edges build from scratch while thick edges are those from one of the initial networks..

Figure 1: A 115 Cities Instance and a Solution (Buying the First and the Third Network)



Input

Each test case has the following form. The first line contains the number n of cities in the country ($1 \leq n \leq 1000$) followed by the number q of existing subnetworks ($0 \leq q \leq 8$). Cities are identied by a unique integer value ranging from 1 to n . The first line is followed by q lines (one per subnetwork), all of them following the same pattern: The first integer is the number of cities in the subnetwork. The second integer is the the cost of the subnetwork (not greater than 2 x 10^6. The remaining integers on the line (as many as the number of cities in the subnetwork) are the identiers of the cities in the subnetwork. The last part of the le contains n lines that provide the coordinates of the cities (city 1 on the first line, city 2 on the second one,

etc). Each line is made of 2 integer values (ranging from 0 to 3000) corresponding to the integer coordinates of the city. The file ends with a 0 0 line. Filename: d.in

Output

For each test case, your program has to write the optimal total cost to interconnect all cities.

Sample Input

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

Sample Output

```
17
```

E. 4 Values whose sum is 0

The SUM problem can be formulated as follows: given four lists A, B, C, D of integer values, compute how many quadruplet $(a, b, c, d) \in$ A x B x C x D are such that $a + b + c + d = 0$. In the following, we assume that all lists have the same size n.

Input

There are several test cases, each test case in his first line contains the size of the lists n (this value can be as large as 4000). We then have n lines containing four integer values (with absolute value as large as 2^{28}) that belong respectively to A, B, C *and* D. The file ends with a 0 line. Filename: e.in

Output

For each test case, your program has to write the number quadruplets whose sum is zero.

Sample Input

```
6
-45  22  42  -16
-41  -27  56  30
-36  53  -37  77
-36  30  -75  -46
26  -38  -10  62
-32  -54  -6  45
0
```

Sample Output

```
5
```

F. Keep the customer Satisfied

Simon and Garfunkel Corporation (SG Corp.) is a large steel-making company with thousand of customers. Keeping the customer satisfied is one of the major objective of Paul and Art, the managers. Customers issue orders that are characterized by two integer values q, the amount of steel required (in tons) and d, the due date (a calender date converted in seconds). The due date has to be met if SG Corp. accepts the order. Stated another way, when an order is accepted, the corresponding amount of steel has to be produced before its due date. Of course, the factory can process no more than one order at a time. Although the manufacturing process is rather complex, it can be seen as a single production line with a constant throughput. In the following, we assume that producing q tons of steel takes exactly q seconds (i.e., throughput is 1). The factory runs on a monthly production plan. Before the beginning of the month, all customers orders are collected and Paul and Art determine which of them are going to be accepted and which ones are to be rejected in the next production period. A production schedule is then designed. To keep customers satisfied, Paul and Art want to minimize the total number of orders that are rejected. In the following, we assume that the beginning of the next production plan (i.e., the rst day of the next month) corresponds to date 0. Hogdson and Moore have been appointed as Chief Scientic Ocers and you are requested to help them to compute an optimal solution and to build a schedule of all accepted orders (starting time and completion time).
Consider the following data set made of 6 orders $J_1, ..., J_6$. For a given order, J_j, q_j denotes the amount of steel required and d_j is the associated due date.

Order	q_j	d_j
J_1	6	8
J_2	4	9
J_3	7	15
J_4	8	20
J_5	3	21
J_6	5	22

You can check by hand that all orders cannot be accepted and its very unlikely you could find a solution with less than two rejected orders. Here is an optimal solution: Reject J_1 and J_4, accept all other orders and process them as follows. Note that the production line is never idle.

Accepted Order	Starting Time	Completion Time
J_2	0	4
J_3	4	11
J_5	11	14
J_6	14	19

Some Hints from Hogdson and Moore:

- Hogdson and Moore claim that it is optimal to sequence accepted orders in non-decreasing order of due dates.
- They also claim that there is an optimal solution such that for any two orders J_u and J_v with $q_u > q_v$ and $d_u < d_v$, if J_u is accepted then J_v is also accepted.
- Finally, Hogdson and Moore advise you to Keep the Customer Satised

Keep the Customer Satised
Gee but its great to be back home
Home is where I want to be.
Ive been on the road so long my friend,
And if you came along
I know you couldnt disagree.
Its the same old story
Everywhere I go,
I get slandered,
Libeled,
I hear words I never heard
In the bible
And Iḿ on step ahead of the shoe shine
Two steps away from the county line
Just trying to keep my customers satisfied,
Satisfied.
Deputy sheriff said to me
Tell me what you come here for, boy.
You better get your bags and flee.
Youre in trouble boy,
And youre heading into more.

Input

Each test case is described by the following format: The first line contains the number n of orders (n can be as large as 800000 for some test cases). It is followed by n lines. Each of which describes an order made of two integer values: the amount of steel (in tons) required for the order (lower than 1000) and its due date (in seconds; lower than 2 x 10^6). The file ends with a 0 line. Filename: f.in

Output

For each test case, you are required to compute an optimal solution and your program has to write the number of orders that are accepted.

Sample Input

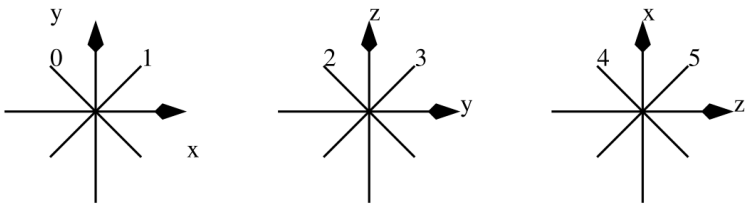
```
6
7  15
8  20
6  8
4  9
3  21
5  22
0
```

Sample Output

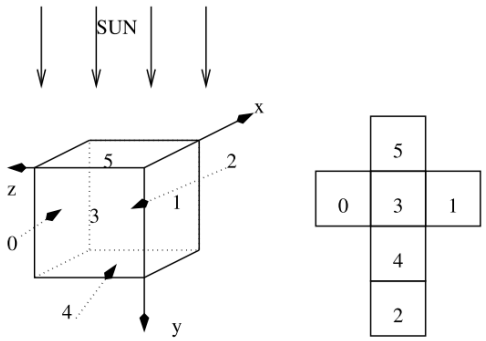
```
4
```

G. UFO Cubes in Roswell

Bob and Alice live in Roswell (New Mexico, USA) and they have found some strange item on the floor left by an Unidentied Flying Object (UFO). Let us recall that on July 8, 1947, as stated by Col. William Blanchard, Commander of the 509th Bomb Group at Roswell, "the wreckage of a crashed disk has been recovered". We believe that this strange item comes from this crash. It looks like a big cube of size s, $1 \leq s \leq 2^{15}$ with tiny holes. It shines dierently on every face. Bob and Alice are rather smart and they have quickly understood that some complex system inside the cube reects the light of the sun so that sunlight emerges from some but not all holes of every face. Experiments led by Bob and Alice have shown that the cube is full of n, $1 \leq n \leq 10^6$ double-sided mirrors. Mirrors are located at integer coordinates. They are small (less than one unit in diameter). They are oriented in one of the following directions:



Bob and Alice analyze the cube by putting a light sensor in front of each face of the cube. This simple experiment allows them to compute the amount of light reected through the corresponding face (i.e., the number of holes through which a light ray emerges). More precisely, the cube is oriented in the following way (the sun is on the $y < 0$ axis) and there is one unit of light which enters the cube on every integer coordinate of the upper face ($y = 0, 0 \leq x \leq s, 0 \leq z \leq s$).



You are requested to simulate the behavior of the cube. Given the position and the orientation of each mirror, you are kindly asked to compute, for each face, the number of rays that go out of the face of and the total number of times the corresponding rays have been deviated. Note that when the direction of a ray belongs to the plane of a mirror, it passes through without any interaction.

Input

In each test case the first integer (on the first line) is the size s of the box. The second line contains an integer n that corresponds to the total number of mirrors in the box. We then have n lines, each of which containing 4 integers : the coordinates (x, y, z) of the corresponding mirror and its orientation (with respect to the above figure). Notice that the coordinates (x, y, z) of mirrors are pairwise distinct. The file ends with a 0 line. Filename: g.in

Output

The output is made of 12 integers (one per line). The 12 integers are made of 6 groups of 2 integers, each group is related to a dierent face i of the cube ordered as shown in the above gure. In each group, the first integer is the number of rays (the amount of light) that go out of the face i and the second one is the number of times the corresponding rays have been deviated.

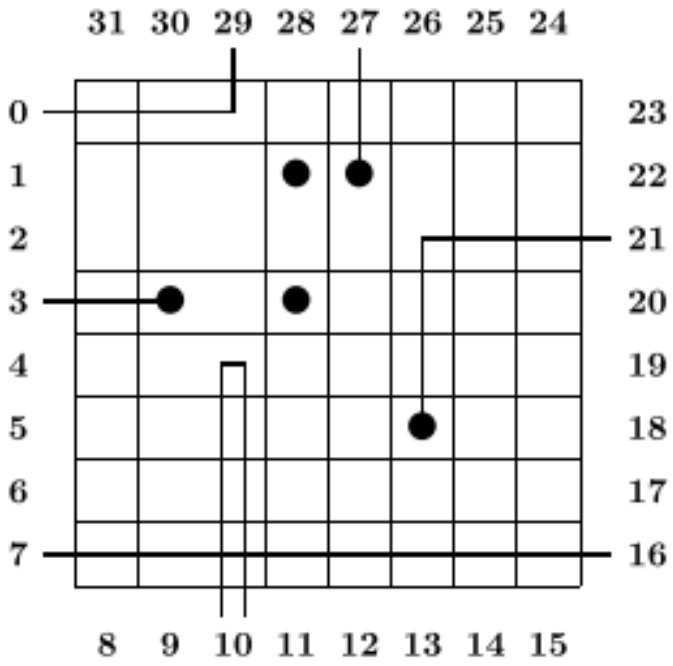
Sample Input

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

Sample Output

```
0
0
0
0
1
1
1
1
34
0
0
0
```

H. Black box



Physicists study atoms hidden in a black box. So as to get information on the position of atoms in the box, they cast a laser beam through gates and look at where light gets out from the box. As a computer scientist you are (kindly) requested to interpret the physicists experiments. By weighting the box, the physicists already managed to count how many atoms there are ($K = 5$). Besides, they adopt a grid model. First, the box is quite simple: this is a flat, N x N box ($N = 8$), with $32 = 4 * N$ gates, which are numbered as shown above. Additionally, following the famous no border principle and a loose application of Pauli exclusion principle, the physicists restrict the available positions to the central $36 = (N-2)^2$ positions, and they assume that no two atoms occupy the same position. Besides, in the grid model, light is also quite simple:

- Light travels at infinite speed in either of the fourth directions, east, north, west or south. For instance, if the beam enters the box from the west, then it travels eastward.
- In the absence of obstacles, light goes straight ahead. See the beam entering at gate 7.
- In case it enters a position occupied by an atom, light is absorbed. Then, there is no output gate. See the beam entering at gate 3.
- Light is deviated by atoms. Before entering a position whose left (resp. right) neighbor contains an atom, light turns right (resp. left). See the beam entering at gate 0 for an example of a left deviation, and the beam entering at gate 29 for for an example of a right deviation.
- Absorption takes precedence over deviation. See the beam entering at position 27.
- When a beam is deviated both left and right at the same time, it turns back. See the beam entering at gate 10 and leaving at the same gate 10, because of such a double deviation.

- Laws of light combine. See the beam entering at gate 21, which is absorbed after a left deviation.

Input

In each test case, the input is a log of experiments performed over a given box. The rst line is an integer e ($0 < e \leq 32$). Integer e is the number of experiments performed. Then, come e lines, each line being made of two integers. The first integer i is a gate number expressing that the beam enters the box at gate i. The second integer o is either a gate number, expressing that the beam leaves the box at gate o, or the integer -1, expressing that the beam is absorbed. The file ends with a 0 line. Filename: h.in

Output

If the atom positions can be deduced from the experiments in each test case, then your program should output an ascii representation of the box, as N lines of N characters, with atoms being shown as "+" and empty positions as "-". Otherwise, your program should output "NO" on a single line. Notice that "NO" is the correct answer in several situations. More specically, the experiments may be contradictory (there does not exist a repartition of atoms compatible with the experiments) or non-concluding (there exist several repartitions of atoms compatible with the experiments).

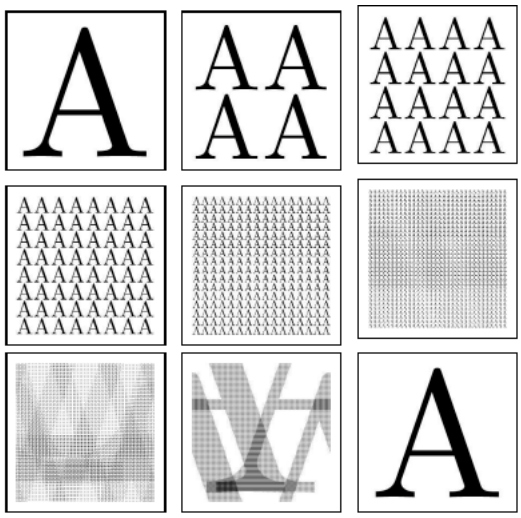
Sample Input

```
7
0  29
27  -1
21  -1
10  10
3  -1
16  7
6  12
10
0  23
1  28
7  8
20  -1
19  25
10  16
2  31
4  5
12  -1
29  -1
0
```

Sample Output

```
--------
---++---
--------
-+-+----
--------
-----+--
--------
--------
NO
```

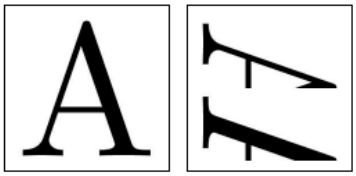
I. Pixel Shuffle



Shuffling the pixels in a bitmap image sometimes yields random looking images. However, by repeating the shuffling enough times, one finally recovers the original images. This should be no surprise, since shuffling means applying a one-to-one mapping (or permutation) over the cells of the image, which come in finite number. Your program should read a number n, and a series of elementary transformations that define a shuffling ϕ of n x n images. Then, your program should compute the minimal number m ($m > 0$), such that m applications of ϕ always yield the original n x n image. For instance if ϕ is counter-clockwise 90^o rotation then $m = 4$.

Input

For each test case, input is made of two lines, the first line is number n ($2 \leq n \leq 2^{10}$, n even). The number n is the size of images, one image is represented internally by a n x n pixel matrix (a_i^j), where i is the row number and j is the column number. The pixel at the upper left corner is at row 0 and column 0. The second line is a non-empty list of at most 32 words, separated by spaces. Valid words are the keywords id, rot, sym, bhsym, bvsym, div and mix, or a keyword followed by "-". Each keyword key designates an elementary transform (as dened by Figure 1), and key- designates the inverse of transform key. For instance, rot- is the inverse of counter-clockwise 90 rotation, that is clockwise 90^o rotation. Finally, the list $k_1, k_2, ..., k_p$ designates the compound transform $\phi = k_1 \circ k_2 \circ ... \circ k_p$. For instance, bvsym rot- is the transform that first performs clockwise 90^o rotation and then vertical symmetry on the lower half of the image.



id, identity. Nothing changes : $b_i^j = a_i^j$.

rot, counter-clockwise 90° rotation

sym, horizontal symmetry : $b_i^j = a_i^{n-1-j}$

bhsym, horizontal symmetry applied to the lower half of image : when $i \geq n/2$, then $b_i^j = a_i^{n-1-j}$. Otherwise $b_i^j = a_i^j$.

bvsym, vertical symmetry applied to the lower half of image ($i \geq n/2$)

div, division. Rows 0, 2, ..., $n-2$ become rows 0, 1, ... $n/2-1$, while rows 1, 3, ... $n-1$ become rows $n/2$, $n/2+1$, ... $n-1$.

mix, row mix. Rows $2k$ and $2k+1$ are interleaved. The pixels of row $2k$ in the new image are $a_{2k}^0, a_{2k+1}^0, a_{2k}^1, a_{2k+1}^1, \ldots a_{2k}^{n/2-1}, a_{2k+1}^{n/2-1}$, while the pixels of row $2k+1$ in the new image are $a_{2k}^{n/2}, a_{2k+1}^{n/2}, a_{2k}^{n/2+1}, a_{2k+1}^{n/2+1}, \ldots a_{2k}^{n-1}, a_{2k+1}^{n-1}$

The file ends with a 0 line. Filename: i.in

Output

For each test case, your program should output a single line whose contents is the minimal number m ($m > 0$) such that ϕ^m is the identity. You may assume that, for all test input, you have $m < 2^{31}$.

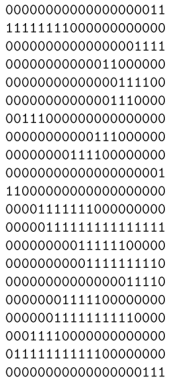
Sample Input

```
256
rot-  div  rot  div
256
bvsym  div  mix
0
```

Sample Output

```
8
63457
```

J. Consecutive ones



A time schedule is represented by a 0-1 matrix with n lines and m columns. Each line represents a person and each column an event. All the persons participating to an event have a one in the corresponding entry of their line. Persons not attending the event have a zero entry in that column. Events occur consecutively. Write a program that finds a *smart permutation* of the events where each person attends all its events in a row. In other words, permute the columns of the matrix so that all ones are consecutive in each line.

Input

For all the test cases, the first line of the input consists in the number $n \leq 400$ of lines. The second line contains $m \leq 400$, the number of columns. Then comes the n lines of the matrix. Each line consists in m characters 0 or 1. The input matrix is chosen so that there exists only one smart permutation which preserves column 0 in position 0. To make things easier, any two columns share few common one entries.The file ends with a 0 line. Filename: j.in

Output

For each test case, the output consists of m numbers indicating the smart permutation of the columns. The first number must be 0 as column 0 does not move. The second number indicate the index (in the input matrix) of the second column, and so on.

Sample Input

```
3
4
0110
0001
1101
6
5
01010
01000
10101
10100
00011
00101
0
```

Sample Output

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