Problem A. Graphics Settings

'Do you know what a gamer does first when he starts a new game?'
'Starts training?'
'No. He goes to the graphics settings and sets everything to maximum. Only after that he starts the training and realizes that it works too slowly. So he goes to the settings once again, turns half of the options off, returns to the training, and so on until he has found suitable parameters. Or until he's tired — then he quits the game and deletes it, which we don't want to happen.'
'What do you suggest?'
'Look. When all the graphics enhancements are turned off, a frame with resolution $w \times h$ pixels is generated in $w \cdot h$ GPU cycles. We know the factors by which the generation is slowed down when a certain option is turned on. So we can calculate the number of GPU cycles in which a frame is generated for a given set of options. Dividing the clock rate of the GPU by this number, we obtain the number of frames per second. If it is too small or too large, we will write about it in the graphics settings.'

Input

The first line contains the number of graphics options n ($0 \leqslant n \leqslant 100\,000$). In the i-th of the following n lines you are given a word s_i, which is the name of the i-th option, and an integer k_i ($2 \leqslant k_i \leqslant 100$), which is the factor by which the image generation is slowed down when this option is switched on. The names of the options contain only lowercase English letters, and their lengths are from 1 to 10. The next line contains integers W, H, and p separated with a space ($320 \leqslant W \leqslant 2\,560$; $200 \leqslant H \leqslant 1\,600$; $1 \leqslant p \leqslant 10^9$). They are the initial width and height of the screen in pixels and the clock rate of the GPU in cycles per second. At the beginning all the options are turned on. In the following line you are given the total number m ($1 \leqslant m \leqslant 100\,000$) of the changes in the option settings made by the user. Each of the following m lines has one of the formats:

- "`On s`" — switch on the option with name s;
- "`Off s`" — switch off the option with name s;
- "`Resolution w h`" — set the screen resolution to $w \times h$ pixels ($320 \leqslant w \leqslant 2\,560$; $200 \leqslant h \leqslant 1\,600$).

It is guaranteed that an option s has been turned off when the change "`On s`" is applied and turned on when the change "`Off s`" is applied.

Output

The first line should describe the performance of the game before changes in the settings. Then next m lines should describe the performance of the game after each change in the settings. Output "`Slideshow`" if the image generation speed is less than 10 frames per second, "`Perfect`" if it is 60 frames per second or greater, and "`So-so`" otherwise.

Example

Input	*Output*
1 vsync 10 640 480 10000000 2 Off vsync Resolution 320 240	Slideshow So-so Perfect

Problem B. Game Testing

In the Higgs Pong game, each player can adjust the graphics to the performance of his computer by switching on or off some of the graphics options. There are n graphics options in the game. If they are all turned on simultaneously, the game slows down even on the most powerful computers.

When the development of the game was almost complete, it turned out that the team's PR expert had placed at several popular game sites the information that the game would work perfectly on the Everest personal computer. Then the developers bought such a computer and asked their tester Ivan to find graphics configurations for which the game would work well on it.

Ivan set up a certain configuration of graphics settings, started the game, played it for some time, and wrote the result in his notebook. Then he continued tests changing exactly one setting before each of the following tests (he either switched on an option that was turned off at that moment or switched off an option that was turned on at that moment). Ivan chose the changes in such a way that no configuration had been tested twice. After some time Ivan decided that he had tested enough configurations and went to the developers to present the results.

Given the initial and final configurations of the graphics settings, find the maximum number of configurations that Ivan could test.

Input

The first line contains the number n of graphics options ($1 \leqslant n \leqslant 16$). In the second line you are given the initial graphics configuration in the form of a list of integers $k\ a_1\ a_2\ ...\ a_k$, where k is the number of active options and the integers a_i are the numbers of these options ($0 \leqslant k \leqslant n$; $1 \leqslant a_1 < a_2 < ... < a_k \leqslant n$). In the second line you are given the final graphics configuration in the same format. The initial and final configurations are different.

Output

In the first line output the maximum number m of tested configurations including the initial and final configurations. In each of the following m lines give these configurations in the order in which Ivan tested them. The configurations must be given in the same format in which the initial and final configurations are given in the input data. If the problem has several solutions, output any of them. It is guaranteed that there exists at least one solution.

Example

Input	*Output*
2 0 2 1 2	3 0 1 2 2 1 2

Problem C. Location Generator

'What are you thinking about?'
'I'm trying to invent a location generator for my level. It's one of the first levels, so I need something simple. I've got a map in the form of a convex polygon, and I must place on it two players and some obstacles to separate them. But if there're too many obstacles, the location will be inconvenient for the game.'
'Fix some vertex A of your polygon. Then if you choose random points B and C strictly inside the polygon, you'll get a triangle ABC (it may be degenerate). Make this triangle impassable and place the players near point A on different sides of the triangle.'
'Won't this impassable zone be too big? The players need enough free space for moving.'
'It seems that the area of this zone is on average rather small compared the area of the whole map. Let's estimate it.'

Input

The first line contains the number n ($3 \leqslant n \leqslant 1\,000$) of the vertices of the polygon. The i-th of the following n lines contains integers x_i and y_i ($-10^6 \leqslant x_i, y_i \leqslant 10^6$), which are the coordinates of the i-th vertex in the counterclockwise order. No three vertices of the polygon lie on the same line. The fixed vertex of the polygon is the one given first in the input.

Output

Output the average area of the impassable zone constructed by the proposed method, if two triangle vertices are chosen uniformly from the interior of the polygon. The absolute or relative error of the answer should not exceed 10^{-6}.

Example

Input	*Output*
4 0 0 1 0 1 1 0 1	0.120370370

Problem D. Similar Tunes

'The boss gave me a disk with soundtracks of all six episodes of the *Star Wars*. We must make sure that the tune we want to use in our game isn't similar to any of the tunes on the disk. He says that even though we've written the tune ourselves, copyright holders may file a lawsuit against us if they find a slightest similarity between our tune and any of their tunes. Then we'll have to pay multimillion fines.'
'But it can take us many hours to listen to all the tunes from the disk, and it can be difficult to find similar tunes by listening. We need to automate the process.'
'I think we can write a program for comparing the tunes and then listen to the tunes from the disk for which the computed similarity to our tune is high enough. The comparison algorithm can be the following. If we neglect pauses and the lengths of sounds, we can write each tune as a string of sounds. For strings s and t of equal length, let's call the number of positions i in which $s[i] = t[i]$ the *intersection size* $I(s, t)$ of these strings. The *absolute similarity* of strings s and t will be the maximum of $I(u, v)$ for all u and v, where u is a substring of s and v is a substring of t and the lengths of u and v are the same. For example, the absolute similarity of the strings "`AAAAA`" and "`ABABA`" is three, and the absolute similarity of the strings "`BABAB`" and "`ABABA`" is four. The *relative similarity* between our tune and a tune from the disk is the ratio of their absolute similarity to the length of the tune from the disk. This value will be computed by our program.'

Input

The first line describes the tune used in the game. The description of the tune starts with the length n of the tune, which is followed by the description of n sounds. The second line contains the number m of tunes on the *Star Wars* soundtrack disk ($1 \leqslant m \leqslant 100$). Each of the following m lines describes these tunes in the same format as the tune from the game is described. All tunes have lengths from 1 to 1 000.

The description of sound is a string "`dN`", "`dN+`", or "`dN-`", where `d` is a number of octave, `N` is a note letter name (an uppercase English letter), "`+`" and "`-`" are symbols of raising and lowering a note by a half tone. There are eight octaves numbered from 1 to 8. Each octave consists of 12 sounds: `C`, `C+`, `D`, `D+`, `E`, `F`, `F+`, `G`, `G+`, `A`, `A+`, `B` (sounds follow in this order in the octave). Some of these sounds have alternative notation. In the following pairs both notations denote the same sound of this octave: (`C+`, `D-`), (`D+`, `E-`), (`E`, `F-`), (`F`, `E+`), (`F+`, `G-`), (`G+`, `A-`), and (`A+`, `B-`). Also a notation `B+` denotes a sound `C` of the next octave, and a notation `C-` denotes a sound `B` of the previous octave. There are no sounds `1C-` and `8B+`.

Output

Output m lines, each containing one number. The numbers are the relative similarities between the tune from the game and the tunes from the disk. The answer must have absolute or relative error not exceeding 10^{-6}.

Example

Input	*Output*
4 5C+ 5D 5C 5G 3 2 5D- 5D 2 5D 4B+ 4 5D 5C+ 5D 5F-	1.000000 1.000000 0.500000

Problem E. Model of the Earth

'Is it a model of the Solar system?'
'Yes, I've just made it. Look, all the planets rotate around the Sun, and some of them also rotate around their own axes.'
'And why does the Earth's axis go through Singapore and Ecuador?'
'That's what I'm trying to fix right now. You see, I have two models of the Earth. Both of them are spheres. The first model is physical. I use it to calculate the rotation of the Earth around its axis and around the Sun and the influence of the Moon and other celestial bodies. The second model is graphical. It's shown on the screen with continents and oceans. These two models have been superposed incorrectly, and now the North Pole is in Singapore.'
'How did you superpose the models?'
'Very simply. I chose several key points on the graphical model and specified their places on the physical model. Then the graphical engine found the rotation that should be applied to one model to make the key points coincide with the key points on the other model. But the engine doesn't know the right correspondence between the key points on the models. Now the key points coincide but not in the way I want.'
'Do you know the number of different rotations of the sphere for which all the key points go into key points?'
'That's an interesting question, let's find it.'

Input

The first line contains the number n of key points on the sphere ($3 \leqslant n \leqslant 200$). Each of the following n lines contains real numbers x_i, y_i, and z_i, which are the coordinates of the i-th key point in a Cartesian coordinate system with origin at the center of the Earth. The coordinates are given in Earth radius units with at most nine fractional digits. The distance between any two key points is at least 10^{-5} units.

Output

Output the number of rotations of the sphere that map every key point to a key point, including the identity rotation. You should consider a rotation to map a key point A to a key point B if A is mapped to a point within 10^{-6} units distance from the point B. Rotations having the same mapping should be considered as the same rotation.

Example

Input	*Output*
3 0 1 0 1 0 0 0 0 1	3

Problem F. Game Optimization

'Is anything not working again?'
'It works, but too slowly. I must constantly recalculate the distance from the ship to the nearest base to check if we can teleport the ship there. When there are too many bases on the map, it works very slowly.'
'Is the map discrete?'
'Yes. The map is an $n \times m$ grid made up of square cells. Ships and bases can be located at the centers of the cells only.'
'Then you can precalculate the distances to the nearest base from the centers of each cell.'
'That's true. Let's try it.'

Input

The first line contains integers n and m ($1 \leqslant n, m \leqslant 1\,000$), which are the numbers of rows and columns in the grid that makes up the map. Then you are given an $n \times m$ matrix consisting of zeros and ones. One means that there is a base at the center of the corresponding cell, and zero means that there is no base at that cell. It is guaranteed that there is at least one base and at least one free cell on the map.

Output

Output n lines with m integers in each line. The j-th number in the i-th line must be equal to the squared distance from the center of the cell (i, j) to the nearest base ($1 \leqslant i \leqslant n$; $1 \leqslant j \leqslant m$). The numbers in each line must be separated by a space.

Example

Input	*Output*
3 4 1000 0000 0010	0 1 4 5 1 2 1 2 4 1 0 1

Problem G. Energy Wall

'Assume that a player has built a base on a new planet. Then he needs some protective shelters to guard the base from a possible attack by locals. Let's add watch towers to the game. The player can arrange towers along the perimeter of the base and equip them with laser guns. At the approach of the enemy, the towers will fire guns at them.'
'Watch towers are a commonplace in all games. Let's invent something new. What if we protect the base with a force field?'
'A good idea. The base is located at the shore of a lava ocean, so it won't be attacked from that side. And the other side can be protected by an energy wall consisting of n sections. Each section is an independent unit and constantly accumulates energy.'
'What if the enemy attack comes too early or happens to be too strong for a particular section?'
'Let's build an energy repository at the base and allow the player to transfer energy from it to any sections of the wall in the case of an enemy attack. If enemy units approach the i-th section of the wall, the player can use the energy stored in the repository to enforce the d-area of the i-th section, i.e. sections with numbers from $(i - d + 1)$ to $(i + d - 1)$ (the sections are numbered consecutively by integers from 1 to n). The energy of the i-th section will increase by $d \times x$ units, the energy of the adjacent sections will increase by $(d - 1) \times x$ units, the energy of the sections at distance 2 will increase by $(d - 2) \times x$ units, and so on. The energy of the sections with numbers $i - d + 1$ and $i + d - 1$ will increase by x energy units. The value of x should be selected so as to use all the energy stored in the repository for an enforcement of the wall. After the enforcement, there will be no energy left in the repository.'
'There must be a possibility to refill the repository. Let's make it possible for the player to transfer all the energy from some sections of the wall to the repository if he thinks that these sections won't be attacked in the near future. The player will store energy and use it in the case of attacks, and so the base will be well protected.'

Input

The first line contains integers n and p, which are the number of wall sections and the amount of energy accumulated by each section in a unit of time ($1 \leqslant n \leqslant 10^9$; $1 \leqslant p \leqslant 100$). The second line contains the number q of the player's actions ($1 \leqslant q \leqslant 10^5$). The following q lines describe these actions in the same order they took place. Each description starts with an integer t, which is the time when the action was performed ($0 \leqslant t \leqslant 10^9$). Then you are given the type and parameters of the action. There can be two types of actions: "`enforce i d`" ($1 \leqslant i \leqslant n$; $1 \leqslant i - d + 1 \leqslant i + d - 1 \leqslant n$) means the use of energy from the repository to enforce the d-area of the i-th section of the wall, and "`save l r`" ($1 \leqslant l \leqslant r \leqslant n$) means the transfer of the whole energy of the sections with numbers from l to r to the repository. Each action's duration can be neglected, no two actions have happened at the same moment of time. At the initial time $t = 0$, the energy level of each wall section and the amount of energy stored in the repository are zero.

Output

For each "`save`" action output the current amount of saved energy, including the just saved one. The answer should be given with absolute or relative error at most 10^{-6}.

Example

Input	*Output*
5 1 4 2 save 4 5 3 enforce 2 2 4 save 3 5 5 enforce 3 3	4.000000 9.000000

The energies of the wall sections after each of the player's actions in the example are as follows: (2, 2, 2, 0, 0), (4, 5, 4, 1, 1), (5, 6, 0, 0, 0), (7, 9, 4, 3, 2).

Problem H. E-Lite

The developers of the e-Lite game made it possible for each player to adjust the parameters of his spaceship as he wants. Initially each player has a spaceship with n empty slots on its body. The player can either put up a jet engine in a slot or use this slot for a cargo module. Each slot can be used for an engine of a specific type, i.e., with a certain direction and power.

The spaceship's controlling system is designed so that all the engines can be either simultaneously turned on at full power or simultaneously turned off. If the engines accelerate the spaceship in different directions, it can happen that the installation of all the engines imparts to the spaceship a smaller acceleration than the installation of only a part of them. That is why a player should think where to install engines to get the maximum acceleration.

Input

The first line contains the number n ($1 \leqslant n \leqslant 1\,000$) of empty slots on the spaceship's body. The i-th of the following n lines contains integers x_i and y_i ($-10^6 \leqslant x_i, y_i \leqslant 10^6$), which are the coordinates of the vector of acceleration imparted to the ship by the i-th engine.

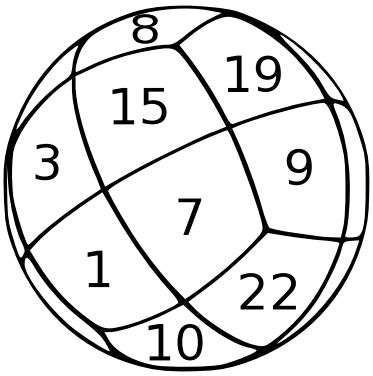
Output

For each i from 1 to n, output in a separate line the value of the maximum acceleration of a ship with exactly i engines. The absolute or relative error of each answer should not exceed 10^{-6}.

Example

Input	*Output*
3 3 -2 -3 -2 0 4	4.000000 4.000000 0.000000

Problem I. Resources Distribution



'Tell me about the bonus level — where a space probe walks over the surface of a planet and collects resources.'
'I needed to decompose the spherical surface into cells. For this, I inscribed a cube in the sphere, and each face of the cube had an $n \times n$ square grid on it. Then I projected the grid from the surface of the cube to the surface of the sphere so that the projection center was the common center of the sphere and the cube. This produced a grid of $6n^2$ cells on the surface of the sphere. Each cell is a spherical quadrangle. The automatic probe can walk over the cells. The probe always leaves a cell through a side opposite to the side through which it entered this cell. It is not allowed to move through the grid nodes.'
'What's next?'
'Now I need to distribute resources among the cells. This problem is more difficult. Each cell should contain some integer amount of resources in the range from 1 to $6n^2$, and these amounts should be different for any two cells. To decrease the luck factor, I want to distribute the resources so that the amount of resources collected by the probe in one complete cycle over the sphere should be independent of the cell where the probe starts the cycle and of the direction of motion. A complete cycle always contains exactly $4n$ cells. The probe walks over them and returns to the cell from which it started the journey.'
'Do you know how to solve this problem?'
'Yes, but it's not easy. You can try and solve it yourself.'

Input

The only line contains the integer n ($1 \leqslant n \leqslant 100$).

Output

Output $6n^2$ different integers in the range from 1 to $6n^2$, which correspond to the amount of resources in each cell of the sphere. The output should have the form of a T-shaped unfolded cube with the integers written on it. Use the format given in the example.

Example

Input	*Output*
2	20 13 11 8 3 15 19 4 12 24 18 5 1 7 9 23 21 17 6 16 10 22 14 2

Problem J. Road to Investor

'How could I forget it? In T hours we must show the alpha-version of our game to a potential investor! This meeting is crucial for us! If we miss it, we possibly won't have a chance to complete the development.'
'Don't panic! Where do they wait for us?'
'In their Tmutarakan office. Let's go right now. I think we'll be there in time, but we'll have to exceed the speed limit in a couple of places.'
'We don't need problems with the police — we'll lose precious time. We'll work out a route to get to the investor in time such that the maximal needed overspeeding along this route is as minimal as possible.'

Input

The first line contains the number n of crossroads and the number m of roads between them ($2 \leqslant n \leqslant 10\,000$; $1 \leqslant m \leqslant 10\,000$). All the roads have two-way traffic. The i-th of the following m lines contains integers a_i, b_i, s_i, and l_i ($1 \leqslant a_i < b_i \leqslant n$; $1 \leqslant s_i \leqslant 300$; $1 \leqslant l_i \leqslant 1\,000$), which mean that there is an l_i-kilometer long road with speed limit s_i kilometers per hour between crossroads a_i and b_i. The game developers' office is at crossroad 1, and the investor's office is at crossroad n. It is guaranteed that there is a way from crossroad 1 to crossroad n. The last line contains the number T of hours left before the meeting ($1 \leqslant T \leqslant 10^6$).

Output

Describe a route by which the developers can get to the investor's office in time with minimum overspeeding. In the first line output numbers S and k, which are the minimum overspeeding in kilometers per hour and the number of roads in the route. In the second line output k integers separated with a space. The integers are the numbers of roads in the order in which they must be traveled. The roads are numbered from 1 to m in the order in which they are given in the input. If there are several ways to get to the investor's office with the overspeeding S, output any of them. The absolute or relative error of S should not exceed 10^{-6}.

Examples

Input	*Output*
3 3 1 3 50 150 1 2 80 100 2 3 80 100 2	20.000000 2 2 3
2 1 1 2 60 60 1	0.000000 1 1