For all problems:

Input file:	***input.txt***
Output file:	***output.txt***
Memory limit:	***256 megabytes***

Problem 1. Measuring the Universe

Time limit: ***1 second***

Ostap Ibragimovich and Ippolit Matveyevich each bought an astrolabe and decided to measure something. And why not to measure – the astrolabe does all the measuring, if only there was something to measure… They've both pointed their astrolabes at the North Pole of the Universe and are measuring all they can. The astrolabe is such a contraption that measures everything, but only within an area whose points are no farther from the observer than R parsecs. Besides that, from the observer's point of view the angle between the positive direction of the OZ axis and the point to be measured can't be more than α degrees. Of course, our friends would like to know the volume of the area reachable by at least one of their astrolabes.

Input data

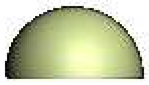
The input file contains two lines. Each line contains five real numbers x, y, z, α, and R – the observer coordinates, the angle within which an astrolabe can make measurements, and the radius of the observable area, respectively ($0 \le x, y, z, R \le 30$, $0 \le \alpha \le 90$), where x, y, z and R are measured in parsecs and α is measured in degrees.

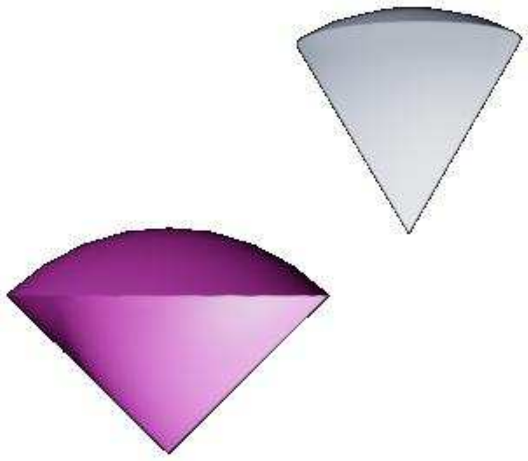
Output data

The output file must contain a single real number – the volume of the measurable area expressed in cubic parsecs with the precision of 0.1 cubic parsec.

Samples

input.txt	output.txt	The schematic representation of the measured area

`0.0 0.0 0.0 90.0 10.0`
`0.0 0.0 5.0 45.0 5.0`
	`2094.4`	

`5.1 5.4 5.6 45.5 15.7`
`20.8 20.1 20.3 30.3 13.1`
	`3067.3`	

Problem 2. Heron in a swamp

Time limit: ***4 seconds***

A heron lives in a moorland. It likes to fly around the swamps nearby and hop from tussock to tussock. When the heron lands in a swamp, it chooses a tussock, stands on it on its left leg and contemplates for a while, then chooses another tussock and hops there using tussocks in between. It makes the first step with its right leg, and uses only one leg to stand on a tussock. It chooses its route in such a way that it ends up on the desired tussock on its left leg; it never steps on the same tussock twice.

The heron is pretty lazy. It would like to make as few steps as possible. Please help the heron.

Input data

The input file contains descriptions of several swamps. The first line of a swamp description contains the numbers n, m, s, t separated by spaces which are the number of tussocks at that particular swamp, the number of transitions between the tussocks, the number of the initial tussock and the number of the final tussock. All tussocks in a swamp have numbers from 1 to n, $s \neq t$..

The following m lines describe the transitions between tussocks. Each line contains two different integers – the numbers of tussocks between which the heron can take a step. The sum of all n*'s* in the input file is smaller than or equals 20. Each swamp has $n \geq 2$.

The input file ends with the following line: **`0 0 0 0`**.

Output data

For each swamp the output file must contain a separate line with the sequence of the numbers of tussocks on which the heron should step in order to get from the initial point to the end point. If there are several sequence options the output should contain any of them. If there is no such sequence the output is to contain the phrase **`No path`**.

Sample

input.txt	output.txt

`3 3 1 2`
`3 1`
`3 2`
`2 1`
`4 4 1 2`
`1 2`
`1 3`
`3 4`
`4 1`
`0 0 0 0`
	`1 3 2`
	`No path`

Problem 3. Trumpet (Division 1 Only!)

Time limit: `1 second`

The trumpet is the musical instrument with the highest register in the brass family. Trumpets are among the oldest musical instruments, dating back to at least 1500 BCE. They are constructed of brass tubing bent twice into an oblong shape, and are played by blowing air through closed lips, producing a "buzzing" sound which starts a standing wave vibration in the air column inside the trumpet. There are several types of trumpet; the most common is a transposing instrument pitched in Bb. The tube length of the most common B-trumpet is about 134 cm. The predecessors to trumpets did not have valves, but modern trumpets have either three piston valves or three rotary valves. Each valve increases the length of tubing when engaged, thereby lowering the pitch. The trumpet is used in many forms of music, including classical music and jazz.

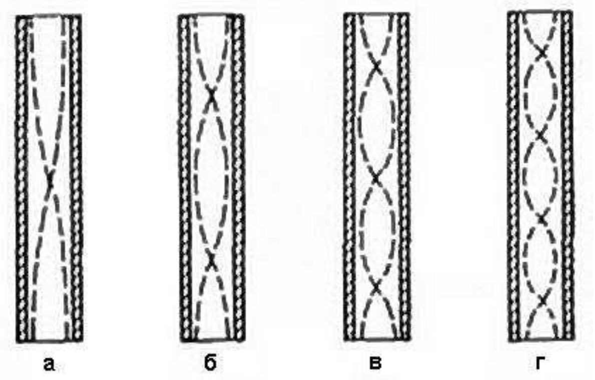
From Wikipedia, the Free Encyclopedia

Let's have a look at how a natural trumpet works. In essence the trumpet is a pipe which is bent solely to save space. It has a small narrowing near the mouthpiece and widens near the mouth, and is otherwise cylindrical. For the purposes of this task let's consider the trumpet to be a cylinder with the length of L, without curves, narrowings, or widenings.

The sounding body of the trumpet is the column of air filling its inner space and separated from the exterior environment by the body of the instrument. This column of air serves as the resonator responding to vibrations of air of a certain frequency depending on the diameter of the resonator.

A special vibrating mechanism in the main part of the trumpet generates static waves whose antinodes are at the ends of the trumpet; when the keynote (the first harmonic) is generated one of the nodes is located in the middle of the trumpet (fig. a).

The narrow and long shape of the air column provides for the ease of the so called overblowing, i.e. the separation of the air column into a number of regions vibrating in opposite phases; this generates new nodes and antinodes, with obligatory antinodes on the edges. This phenomenon can be seen on fig. b, c, and d, which show the generation of the second, third, and fourth harmonic, respectively.



This means that a natural trumpet can only generate sounds with wavelengths that divide the doubled length of the pipe into an integer. The set of all such sounds is called its harmonical sound scale, or natural sound scale. The k-th sound of the scale is called the k-th harmonic and has a wavelength of

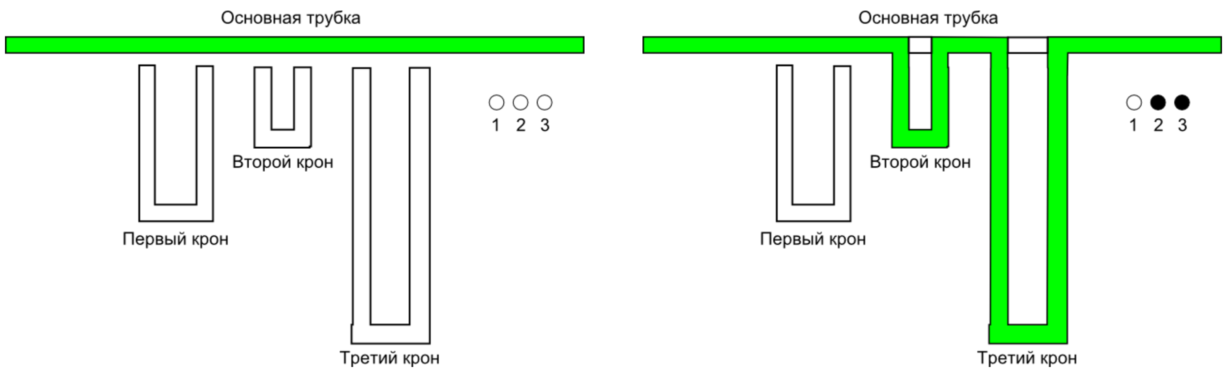
$$\lambda_k = \frac{2L}{k}$$

Due to the technical features of the trumpet the keynote, i.e. the first harmonic, cannot be generated. The second and the consequent harmonics are generated in perfect accordance to the theory described above. Hence we can only generate λ_k wavelength where $k \geq 2$.

The inability of the natural trumpet to produce sounds outside the natural sounds scale seriously limits its repertoire. For example, a particular trumpet can only be used to play tunes of a certain tonality which matches the trumpet's keynote. Moreover, most of modern music is written using the chromatic sound scale which is much richer than the natural scale.

To expand the sound scale of the trumpet to the full chromatic scale the ventil system has been devised. The ventil system allows to expand the length of the air column, thus lowering the natural sound scale.

Apart from the main pipe the ventil trumpet also has N valve tubes. A valve tube is an additional pipe of a certain length. Each valve tube is linked to the main pipe through an individual ventil. When a ventil is pushed, the corresponding valve tube is instantly linked to the main pipe, thus expanding the length of the vibrating air column by the length of the connected tube. The ventils can be pushed independently of each other. When several ventils are pushed the air column length is expanded by the sum of lengths of all corresponding tubes.



At the picture the green color shows the vibrating air column.
No ventils are pushed on the left; on the right the second and the third ventils are pushed.

The ventil mechanism of the modern trumpet allows to produce all sounds of the chromatic scale of European music with high fidelity.

Considering that the speed of sound in air is static, c = 340 meter per second, we can unambiguously define the frequency by its wavelength and vice versa. Let's recall that the wavelength multiplied by frequency equals the speed of sound:

$$F\lambda = c$$

In the era of classical music piano became the main keyboard instrument. It's a well-known fact that the sequence of all keys of the piano from left to right forms a chromatic sound scale. In theory, a perfectly tuned piano must correspond to the equitempered scale. In such a scale the interval between any two adjascent sounds of the chromatic scale is the same and equals a half-tone. If two sounds have the frequencies F_1 and F_2, the interval between them in half-tones is calculated using the formula:

$$I(F_1, F_2) = 12\left|\log_2 \frac{F_2}{F_1}\right|$$

For example, an interval of a half-tone means the proportion of frequencies equaling $\sqrt[12]{2}$, and an interval of twelve half-tones (an octave) means a frequency proportion of 2.

All sounds (notes) of a chromatic gamma have their own symbols. The symbol consists of the note name and octave number. The names of the notes are repeated in every 12 sounds. The octave number raises by 1 every 12 sounds. Below is a table showing a part of the chromatic scale.

Number	−9	−8	−7	−6	−5	−4	−3	−2	−1	0	1	2
Symbol	**C1**	**C#1 Db1**	**D1**	**D#1 Eb1**	**E1**	**F1**	**F#1 Gb1**	**G1**	**G#1 Ab1**	**A1**	**A#1 Bb1**	**B1**
Number	3	4	5	6	7	8	9	10	11	12	13	14
Symbol	**C2**	**C#2 Db2**	**D2**	**D#2 Eb2**	**E2**	**F2**	**F#2 Gb2**	**G2**	**G#2 Ab2**	**A2**	**A#2 Bb2**	**B2**

Note that some notes can be defined in several ways. This table can be continued indefinitely in both directions – both by increasing and decreasing the note number. In this task all notes will be between the DO of the small octave (C0 = −21) through the SI of the fifth octave (B5 = 50) inclusively.

It is a standard that the LA of the first octave (A1 = 0) is 440 Hz. Since every consecutive note has a $\sqrt[12]{2}$ times higher frequency, and each preceding note has a $\sqrt[12]{2}$ times lower frequency, it is possible to define frequencies of all notes.

It's obvious that a ventil trumpet with few ventils is incapable of perfectly producing all notes of the equitempered chromatic scale. The playing technique presumes pressing such a combination of ventils that the error is minimal. Than means choosing one of all 2^N variants of pressing the ventils. It also means choosing the natural number of the harmonic. Together these two parameters unequivocally define the frequency of the sound. We must minimize the interval between this frequency and the perfect frequency of the note the musician is trying to produce.

In this task you'll be given a trumpet configuration: the number of its ventils, length of the main pipe and valve tubes as well as the sequence of notes to be played. You will have to define the optimal configuration to play this notes and the extent of unavoidable error.

Input data

The first line of the input file contains two integers. N is the number of ventils on a trumpet ($1 \le N \le 10$) and L_0 is the length of the main pipe in millimeters ($100 \le L_0 \le 10000$). The second line contains N integers — the lengths of all valve tubes in millimeters. The length of each valve tube is within 20 to 2000 millimeters. This sums up the description of the trumpet.

The next line contains the number M – the number of notes to be played by the musician ($1 \le M \le 100$). The following M lines each contain 2 or 3 symbols defining the notes. The # symbol has the ASCII code 35. The note pitch limits are described above in the task text.

Output data

The output file must contain M lines. Each line must contain a description of the optimal way to perform a specific corresponding note from the input file.

In the beginning of the description there should be a word containing N symbols – a combination of pressing the ventils. Each symbol denotes the state of the corresponding ventil — **'.'**(ASCII 46) denotes the unpressed state, **'o'** (ASCII 111) denotes the pressed state. The second thing to put out is the number of the harmonic used. The number is an integer equal or higher than 2 (the reason is explained in the task text above). The last thing the output must contain is the length of interval between the necessary frequency and the best achievable frequency. The interval should be measured in half-tones as a real number with an absolute error below 10^{-6}. It is guaranteed that the optimal performance scheme is unique.

Samples

input.txt	output.txt

`3 1300`
`159 77 261`
`1`
`C0`
	`ooo 2 6.389290`

`3 1300`
`159 77 261`
`11`
`G1`
`C2`
`G1`
`A1`
`B1`
`E1`
`E1`
`A1`
`G1`
`F1`
`G1`
	`... 3 0.013785`
	`... 4 0.005765`
	`... 3 0.013785`
	`oo. 4 0.106236`
	`.o. 4 0.001968`
	`oo. 3 0.125786`
	`oo. 3 0.125786`
	`oo. 4 0.106236`
	`... 3 0.013785`
	`o.. 3 0.016166`
	`... 3 0.013785`

```
3 1300
159 77 261
31
F#0
G0
G#0
A0
A#0
B0
C1
C#1
D1
D#1
E1
F1
F#1
G1
G#1
A1
A#1
B1
C2
C#2
D2
D#2
E2
F2
F#2
G2
G#2
A2
A#2
B2
C3
```

```
ooo 2 0.389290
o.o 2 0.147472
.oo 2 0.006850
oo. 2 0.106236
o.. 2 0.003384
.o. 2 0.001968
... 2 0.005765
ooo 3 0.408840
o.o 3 0.167022
.oo 3 0.012700
oo. 3 0.125786
o.. 3 0.016166
.o. 3 0.017582
... 3 0.013785
.oo 4 0.006850
oo. 4 0.106236
o.. 4 0.003384
.o. 4 0.001968
... 4 0.005765
oo. 5 0.030627
o.. 5 0.140247
.oo 6 0.012700
ooo 7 0.077549
o.. 6 0.016166
.o. 6 0.017582
... 6 0.013785
.oo 8 0.006850
oo. 8 0.106236
o.. 8 0.003384
.o. 8 0.001968
... 8 0.005765
```

Comments

All tests in the examples show a standard ventil trumpet configuration.

In this configuration the keynote is the DO of the small octave (C0). Hence **C1**, **G1**, **C2**, **E2**, **G2**, **Bb2**, **C3** and so forth can be played with relatively high fidelity without pressing the ventils.

The valve tube lengths are such that pressing the first ventil the note pitch falls 2 half-tones, the second – 1 half-tone, the third – 3 half-tones. When several ventils are pressed the tone decrease is combined. For example since G1 can be played without pressing the ventils, it is possible to play F1 by pressing the first ventil, or play Eb1 by pressing the second and third ventil. All this is true to some degree of fidelity.

It the first test you must generate they keynote of the trumpet. Since it's impossible the best solution is to press all ventils and choose the minimal harmonic. This way the error is very substantial – about 6 half-tones.

The second test contains the first 11 notes of the Russian anthem.

The third test contains the full chromatic scale in the classic trumpet range: from FA-sharp of the small octave (F#0) to DO of the third octave (C3). The answer to this test is a finger notation every trumpeter knows. It's exactly the way the ventils are pressed when playing the trumpet. The only difference between the solution for this test and the classic finger notation is the MI-sharp of the second octave (E2) which is usually played without pressing the ventils.

Problem 4. Restaurant (Division 1 Only!)

Time limit: `3 seconds`

It's been a hot year in the resort town of X. So hot that people who get out in the sun get a heat stroke in an instant.

Uncle Nick, a born entrepreneur, decided to open a restaurant in the town. At the moment he's thoroughly planning the location of the restaurant. He's an optimist, so he's absolutely sure that his restaurant will be the best in town: he doesn't take the competition into account. What he really has to worry about is the blazing sun.

In Nick's model the town is a graph: the nodes are the town's districts, and the edges are the shaded passages between the districts. At any moment of time depending on the position of the sun in the sky a district is either completely sunlit or completely shaded. The passages are always in shade. Uncle Nick must choose the best district for the restaurant taking all this into account.

He's sure that not a single person will go to the restaurant if he or she has to pass through a sunlit patch, even if only the restaurant is in the sun, or the sun is above the district where that person is initially. We can imagine that the guests of the resort town can move along shaded areas instantly.

Uncle Nick made a "crowdedness" map of the town. He measured the "crowdedness" for each district – that is, the number of people wishing to go to the restaurant per unit of time. When a tourist decides to go to the restaurant, he or she checks whether it's possible to reach the restaurant while staying in shade. If it's possible, the tourists goes there, and Uncle Nick profits 1 ruble. If it's impossible at the moment, the tourists does something else.

Uncle Nick has already decided at what hours his restaurant will be open. There'll be no breaks in the restaurant hours. The hot sun which is terrorizing the city changes its position during the day. Some districts of the town become sunlit and some become shaded. The change is instant in each district.

At daybreak all districts are shaded. During sunrise some districts become sunlit. Some time later the sun starts to go down, and the sunlit district become shaded again. At dusk all districts are shaded again. Please note that a district can't become shaded during sunrise, nor can a district become sunlit during sunset.

Uncle Nick wants to maximize his income. Use his data to find the best district for the restaurant in terms of profit.

Input data

The first line of the input file contains four integers: N is the number of districts in the town, M is the number of passages between them, To is the opening time and Tc is the closing time of the restaurant ($1 \le N, M \le 100000$, $0 \le T_O < T_C \le 10^7$).

The next line contains the "crowdedness" values for the town districts – N integers between 0 and 10^6 inclusive.

The following M lines describe the passages between the districts. Each line contains two integers A_i and B_i – the numbers of districts linked by that passage($1 \le A_i \ne B_i \le N$). No passage is repeated in the input data. The tourists can move along any passage in both directions.

The following lines contain the description of the lighting schedule of the districts. It starts with a line containing a single integer Q – the number of lighting changes per day ($0 \le Q \le 2N$). The following Q lines describe the changes of lighting status of the city districts in strict chronological order.

Each line contains a symbol (a plus or a minus) and two integers. The first integer T_j is the moment in time when the lighting changes ($0 \le T_j \le 10^7$). The second integer V_j is the number of the district where the lighting changes ($1 \le V_j \le N$). The plus symbol means the district becomes sunlit and the minus symbol means the district becomes shaded. A district is either not mentioned at all or is mentioned twice in the schedule – with a plus and with a minus. All changes with a plus happen earlier than all changes with a minus.

Output data

The output file must contain two integers: the number of the district which would be optimal for the restaurant and the daily income in rubles for that location. If there are several optimal solutions the output can contain any of them.

Sample

input.txt	output.txt

`3 2 0 10`
`1 1 1`
`1 2`
`2 3`
`6`
`+ 1 3`
`+ 2 2`
`+ 3 1`
`- 4 1`
`- 5 2`
`- 6 3`
	`1 21`

Comments

The example shows three districts linked by passages linearly. A single hungry tourist appears in each of the districts in a unit of time. We can draw the status of the districts throughout the day.

Time gap	District status
0-1	###
1-2	##.
2-3	#..
3-4	...
4-5	#..
5-6	##.
6-7	###
7-8	###
8-9	###
9-10	###

It's obvious that if Nick opens the restaurant in district 1 then all tourists from the first district will be able to reach it at any moment of time except the period 3-4, the tourists from district 2 will be able to reach the restaurant in periods 0-2 and 5-10; the tourists from the third district will be able to reach the restaurant at 0-1 and 6-10. If he opens the restaurant in district 3, the area will be lit between 1 and 6, hence there won't be any customers and the profit will be only 15 rubles.

Problem 5. Supercomputer (Division 1 Only!)

Time limit: **4 seconds**

The International Brotherhood of Magicians (IBM) created a brand new supercomputer. Its special feature is that it has an infinite number of processors and unlimited memory. Moreover, thanks to the hypercube architecture and the tachyon bus the memory access time for the processors is zero. Nevertheless, floating point calculations usually require certain amounts of time. In particular the addition and subtraction time equals t_a, the multiplication time is t_m and division time is t_d, calculating a square root is t_s. The negation operation takes no time to complete. However, the operations of floating point number comparisons are still inefficient and thus are blocked. Moreover, due to the hardware implementation of the libastral library implementation all integer arithmetic operations, including comparisons, are instant. Unfortunately, the implementation is incomplete and doesn't produce the answer to a task right away.

One of the first customers to buy this supercomputer was an organization called MJ12. The task they need to solve has the following conditions. Let A be a real symmetric matrix of the size of N×N with known locations for its non-zero entries. The Cholesky decomposition of the matrix A is such a lower triangular matrix L that $A=LL^T$ where L^T is the transposed matrix L. For L to exist coefficients of the matrix A must comply with specific conditions. We assume that these conditions are met. In that case the numbers of L can be calculated using the following recurrent formulae:

$$L_{jj} = \sqrt{A_{jj} - \sum_{k<j} L_{jk}^2}$$

$$L_{ij} = \frac{A_{ij} - \sum_{k<j} L_{ik} L_{jk}}{L_{jj}}, i > j$$

MJ12 needs to calculate the L matrix which corresponds to the given A matrix. It is clear that since some of the matrix A entries are zero, some of the matrix L entries will be zero as well. All other entries of the L matrix must be calculated. Before the purchase of the supercomputer MJ12 wants to know how quickly it will be able to solve the task using their super-secret matrices. In order to avoid disclosing them to the IBM they demanded a program which being given non-zero elements of the matrix A can calculate the minimal time for the supercomputer to compute the entries of the matrix L. Since the IBM engineers are busy tuning up the libastral, they delegated this task to you.

MJ12 requires that the calculation of L to be performed in complete accordance with the defined formulae. However that might prove inefficient, and you've managed to persuade them to allow using the following identity laws to optimize the calculations:

1. $a + b = b + a, \quad a - b = -(b - a)$
2. $(a + b) + c = a + (b + c), \quad (a - b) + c = a - (b - c), \quad (a - b) - c = a - (b + c)$
3. $0 + a = a + 0 = a \quad a - 0 = a, \quad 0 - a = -a$
4. $0 \cdot a = a \cdot 0 = 0$

The identities (1) and (2) in essence allow the addition of the components under the root and in the term of fraction to be performed in any order; the identities (3) and (4) allow to take into account the structure of L and A matrices and avoid additions with and multiplications by numbers known to be zero.

Please keep in mind that the supercomputer has an unlimited number of processors. The model of parallel computation using an infinite number of processors presumes that if there is an arithmetic operation to be performed, it will start as soon as all of its arguments are available.

Input data

The first line of the input file contains two integers, N and M, ($1 \le N \le 50000$, $0 \le M \le 200000$) which are the size of the symmetric matrix A and the number of its non-diagonal entries in the lower triangular part of it. The following M lines contain pairs of numbers r_j and c_j which are the row and column where the non-diagonal entry j ($1 \le c_j < r_j \le N$) is located. It is guaranteed that each non-zero entry is specified exactly once. Non-zero entries of the matrix A diagonal are not specified in the input file since it's assumed that the whole diagonal is non-zero. The last line of the input file contains four nonnegative integers t_a, t_m, t_d and t_s which are the time it takes to calculate the corresponding operations. These numbers are not greater than 10^9.

Output data

The only line of the output file must contain a single integer – the minimal time necessary to calculate the Cholesky decomposition of the matrix defined in the input file. It is guaranteed that the number of non-zero elements of the L matrix including the elements on its diagonal is not greater than 300000.

Samples

input.txt	output.txt

`2 0`
`1 1 1 1`
	`1`

`3 1`
`3 2`
`1 2 3 4`
	`14`

Problem 6. Cloning

Time limit: `2 seconds`

The Magic Copypaste Institute (MCI) has been living a serene life; the magician scientists studied the science of copypaste, wrote articles, received grants and lived a happy life until the inspectors came. The inspectors evaluated the state of affairs in the institute as pitiful and made a decision: the MCI has so few worthy employees and its popularity is so low that its funding must be cut. However, the director of the institute, I.V. Copyraiko, managed to persuade the inspectors that the situation is just a temporary crisis due to demographic problems, and that soon the MCI would become popular again. He talked the inspectors into repeating their check in some time. When the inspectors left, the employees of the MCI began to think of ways to solve the problem.

A young specialist in cloning living things, A.V. Manclone, proposed his variant which everyone liked instantly. He proposed to use his brand new spell for cloning cattle on humans to produce as many employees as necessary. However it wasn't all that easy. The director contacted the Institute of Perfect Numbers (IPN) which made some calculations and stated that in order to retain the current funding levels they'd have to have Y employees, where Y is a new magic number invented in the IPN. Let's keep in mind that the whole thing takes place in the year 30 000 and it's normal for an institute to have 10^8 employees. The Manclone's spell works in the following way: a certain number of the currently available employees are assembled together, and the spell is cast. Each time the spell can be cast by a different number of employees. As the result, each employee of the institute produces as many clones as there were people in the group upon which the spell was cast. The employees now must find the fastest way to achieve the target amount Y of employees, because casting the spell takes a week, and the inspectors aren't that patient. The employees asked us to help them solve the problem.

Input data

The only line of the input file contain two integers X and Y where X is the real number of the institute employees and Y is the required number ($0 < X < Y \le 10^8$).

Output data

The only line of the output line must contain a single number – the minimal number of spell casts necessary to produce Y employees from the available X employees. In case it's impossible the output line must contain the word **`Impossible`**.

Sample

input.txt	output.txt
`3 18`	`2`

Comments

In the example the spell is first cast on three people producing 9 employees. Then the spell is cast on 2 people multiplying the number of employees two-fold.

Problem 7. A day at the woods

Time limit: ***1 second***

People from the city really liked to spend their weekends at the forester's range. A bus would take them to the forester's hut and they'd hike to forest clearings nearby to picnic and bathe in the sun. But it often happened so that they lost their way and roamed from one clearing to another, taking different trails. The forester decided to help the silly citizens. He put numbers on all clearings and put arrows on trails so that a citizen walking from the forester's hut along the arrows, passing on to other clearings no more than once, would find himself on clearings with increasing numbers. If that citizen decided he wanted to return he'd have to go against the direction of the arrows. The decreasing numbers of the clearings would mean he'd be going in the right direction. After renumbering all the clearings should have different numbers from 1 to n.

The forester has a map of his forest showing the clearings and the trails linking the clearings. Any two clearings can be linked with one or more trails.

The forester put the arrows all right, but he's having trouble with the clearing numbers. Help the forester put numbers on the clearings according to his plan.

Input data

The input file contains several test data sets.

The first line of each test set contains the integers n, m, and s, separated by spaces – the number of clearings, the number of trails, and the number of the clearing where the forester's hut is located. All clearings are marked with numbers from 1 to n.

The next m lines describe the trails – they each contain two integers denoting the numbers of the clearings a trail links in the direction of the arrow put by the forester. In some cases these numbers can be the same.

The sum of n in all tests is smaller than or equals 1000. The sum of m in all tests is smaller than or equals 1000.

The input file ends with the line `0 0 0`.

Output data

For each test the output must contain the numbering of the clearings in a separate line. If no acceptable numbering exists, the output must contain the phrase **`No solution`**.

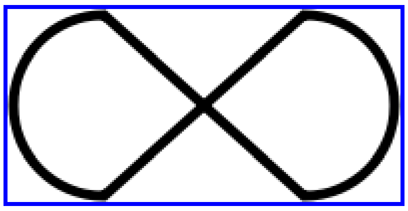
Sample

input.txt	output.txt

`3 3 1`
`1 2`
`2 3`
`1 3`
`4 4 4`
`4 1`
`1 2`
`2 3`
`3 1`
`4 6 1`
`1 2`
`1 3`
`1 4`
`2 3`
`3 4`
`4 2`
`0 0 0`
	`1 2 3`
	`2 3 4 1`
	`No solution`

Problem 8. Running in eights

Time limit: **`1 second`**



Usually when an internet user is agitated, he starts worrying and running around in funny eights. In doing so he's using up all the space available, filling the largest possible volume with his worries. The more eights he makes, the calmer he becomes.

Peter the LJ user is being anxious in an orthogonal room. The eight-track he's taking has the following configuration: there are two semi-circles each adjacent to a short side of the room and to both long sides; two criss-crossed segments link the ends of the semi-circles.

Peter has enough free space to run, he's running at a constant speed, so if we know the length of the 8-track we can easily calculate the number of full eights he must run.

Input data

The first line of the input file contains two integers A and B – the size of the room in meter ($1 \le A, B \le 10000$). The room is orthogonal but not square.

Output data

The output file must contain one natural number – the length of the eight-track in meters to the precision of centimeters.

Sample

input.txt	output.txt
`1 2`	`5.97`

Problem 9. Flight of the rook

Time limit: **1 second**

Moments before the epic throw of the rook into a one-eyed chess player Ostap Benter is solving a tricky problem in mind – how to throw the rook so that it hits the aim for sure?

Let's have a look at the flight of the rook which Ostap has hidden up his sleeve.

The rook passes the first X meters with a constant acceleration a. At this period its speed (a fact well known to us, not to Ostap) is changing according to the law of $v = at$, and the traveled distance – according to $l = \frac{at^2}{2}$, где t is the time lapse from the moment of the throw.

The rook passes the following Y meters in the dense layers of the atmosphere, the resistance of which is directly proportional to the speed of the rook with the index k. Here its speed changes according to the following law: $v = v_0 \cdot e^{-kt}$; the traveled distance is $l = \frac{v_0}{k} - \frac{v_0 e^{-kt}}{k}$ where v_0 is the speed at which the rook enters the dense layers of the atmosphere, and it coincides with the final speed given by the Great Combinator to the rook; k is the resistance index and t is the time which has lapsed since the moment of entrance into the atmosphere.

If the time that lapses while the rook is traveling the distance X+Y is greater than the predefined value T_0, the rook won't reach its aim – the one-eyed chess player will manage to dodge. Naturally, Ostap cannot allow that to happen. It's necessary that the total time of the rook flight towards the aim be below T_0 while casting the minimal possible acceleration to the rook to save powers – which Ostap will surely need to escape from the raging crowd of chess players.

Ostap may have played chess at some time – once, perhaps – but he's surely ignorant when it comes to complicated formulae such as these. He won't cope with this task alone. Help him.

Input data

The first line of the input line contains four real numbers X, Y, k, and T_0 separated by spaces, where X and Y are the lengths of the first and second parts of the rook flight, k is the resistance index and T_0 is the reaction time of the one-eyed amateur chess player ($0.1 \le X, Y \le 100$, $0.1 \le k \le 1$, $0.1 \le T_0 \le 10$).

Output data

The output file must contain a real number – the necessary minimal acceleration a to a precision of 10^{-6}. It's guaranteed that this number is not greater than 100.

Sample

input.txt	output.txt
`2.0 3.0 1.0 4.0`	`2.567201`

Problem 10. Rogue-like (Division 1 Only!)

Time limit: **8 seconds**

Rogue-like is a genre of computer games. These games always work in text mode, when the game field is an orthogonal field of cells filled with ASCII-symbols. A symbol in the field cell denotes the contents of the cell. In this task we'll use only two types of cells – '#' for wall and '.' for free space.

The text mode of the game presumes certain interesting limitations to the gaming process. To illustrate this let's have a look at how a spell called Death Ray used in one of the rogue-like games works. The spell creates a ray which goes straight and reflects from walls. However, only 1 of 8 directions can be chosen to cast the ray, denoted by digits 1 to 8: 1 (down left), 2 (down), 3 (down right), 6 (right), 9 (up right), 8 (up), 7 (up left), 4 (left).

7	8	9
4	@	6
1	2	3

Let's now look at the process of casting the spell in more detail. The player is located in a free cell and casts the ray in any of the 8 directions. The movement of the ray is modeled step-by-step. At the first step the ray is situated in the same cell as the player. With each consequent step the ray shifts one cell along its direction. Upon hitting the wall the ray is reflected according to the following rules:

1. If the edge of the wall in the area of the point of tapping is a straight line then the reflection occurs according to the "reflection angle equals the tap angle" law. Examples:

####	####	####	####
####	####	####	####
....	.↑..	.↓..	....
.↑..	....	....	.↓..
Step 1	Step 2	Step 3	Step 4

####	####	####	####
####	####	####	####
....	.↗..	..↘.	....
↗...	....	....	...↘
Step 1	Step 2	Step 3	Step 4

2. Upon touching a wall edge the ray moves further. For example:

##..	##..	##..	##.↗
##..	##..	##↗.	##..
....	.↗..	....	....
↗...	....	....	....
Step 1	Step 2	Step 3	Step 4

##..	##..	##..	##.↗
##..	##..	##↗.	##..
..##	.↗##	..##	..##
↗.##	..##	..##	..##
Step 1	Step 2	Step 3	Step 4

3. If a ray hits an inner angle it's not considered a touch and the ray direction is reversed. Examples:

####	####	####	####
####	####	####	####
..##	.↗##	.↙##	..##
↗.##	..##	..##	↙.##
Step 1	Step 2	Step 3	Step 4

..##	..##	..##	..##
..##	..##	..##	..##
....	.↗..	.↙..	....
↗...	....	....	↙...
Step 1	Step 2	Step 3	Step 4

After K+1 steps the ray disappears. In this case K is the spell range. If the ray passes through a cell where a player or an opponent is located, the poor thing dies. This does not affect the further advance of the ray.

In our task you're given a game field the size of M by N cells, and Q queries for it. Each query defines the position of the player, the position of the opponent, and the range of Death Ray. For each query you must define the result of casting the Death Ray spell in each of the 8 possible directions. The result is defined by a single letter:

`M` (iss) – the ray doesn't hit anyone;
`W` (in) – the ray kills the opponent without hitting the player;
`L` (oss) – the ray kills the player without hitting the opponent;
`D` (eath) – the ray kills both the player and the opponent

Input data

The first line of the input file contains the field size: M lines and N columns, ($1 \le M, N \le 800$). Each of the consequent M lines contains N symbols – the game field map. The first symbol of the first line has the coordinates (1, 1). The next line contains a number Q – the number of queries ($1 \le Q \le 100000$).

The description of each query takes up one of the following Q lines. Each description contains five integers separated by spaces: R_P and C_P – the player's position, R_E and C_E – the opponent's position and D – the Death Ray range ($1 \le R_P, R_E \le M$; $1 \le C_P, C_E \le N$; $1 \le D \le 10^9$). For both the player and the opponent R defines the line number and C defines the column number. It's guaranteed that the player and the opponent are located in different empty cells. It is considered that all cells outside the game field are filled with walls (**`‘#’`**).

Output data

Each of the Q lines of the output file must contain precisely 8 symbols – the answer for the corresponding query. For each of the directions the output should contain a symbol (**`M`**, **`W`**, **`L`**, or **`D`**) in the following order: **`12369874`**.

Sample

input.txt	output.txt

`3 3`
`...`
`.#.`
`...`
`3`
`1 1 1 3 2`
`1 1 1 3 1`
`1 1 3 3 10`
	`MMLWMLLL`
	`MMLMMLLL`
	`DLLLDLLL`

Problem 11. Servers (Division 1 Only!)

Time limit: ***6 seconds***

A computer network in an apartment house has been built by linking a new computer to the last of the connected computers. So the network consists of N consecutively linked computers. Neighbors have been exchanging information with each other but they've come to realize that they need some proxy-servers. They decided to turn some of the computers into them. The computer community of the house can install only K server software onto computers. They now have to decide which computers will become proxy servers. The main criterion is the monthly maintenance cost for all computers.

Each computer has been assigned its own server fee in rubles per wire length in meters. The maintenance cost for a computer is the server fee for this computer multiplied by the total length of wire linking the computer to its server.

Your task is to write a program which will position the K servers in such a way that the total maintenance costs for all computers are minimal.

Input data

The first line of the input file contains two integers N and K – the number of computers in the network and the number of servers to be installed ($1 \le K \le N \le 2000$).

All computers in the network are numbered from 1 to N in order of their connection to the network.

The second line of the input file contains an integer T_1 – the server fee for the first computer.

The following N – 1 lines each contain two non-negative integers L_i, T_i – the information about the rest of the computers in the network according to their numbers, separated by spaces. L_i is the length of the wire connecting the i-th computer with the adjacent one with a lower number, T_i is the server fee for this computer ($2 \le i \le N$). All L_i and T_i don't exceed 10^6.

Output data

The first line of the output file must contain an integer – the minimal maintenance cost for all computers by all servers. The second line must contain K numbers of computers which are to be turned into servers, separated by spaces. When several solutions are possible any of them will be accepted.

Samples

input.txt	output.txt

`3 1`
`10`
`2 2`
`3 3`
	`19`
	`1`

`3 2`
`10`
`2 2`
`3 3`
	`4`
	`1 3`

Problem 12. Anagrams (Division 2 Only!)

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Two words are called «anagrams» if they contain the same letters but in a different order. For example `word` and `drow` are anagrams, but `word` and `worm` are not. In this problem you will be supplied with a set of words. All you have to do is to pick out the word with the most anagrams and report how many anagrams it has.

Input

Input begins with a number n, which is the number of words in the set ($0 \leqslant n \leqslant 1000$). Set contains n words, each on a single line. All words consist of lower case letters only. No word will contain more than 6 letters. It is guaranteed that set will contain at least one word that has an anagram in the list.

Output

Output consists of one word — the word with the most anagrams within the set, followed by a space, followed by the number of anagrams in set (excluding word itself). The word displayed will be the first occurrence in the input file of the anagram. If more than one word has the same highest number of anagrams, display only the one that comes first in the input file.

Example

input.txt	output.txt
6 nat cat act out tac ant	cat 2
6 worm word galo drown goal drow	word 1

Problem 13. Banking (Division 2 Only!)

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

The bankers of Somebank have collected day-to-day data of the daily profits (and losses) of the shares they hold. Based on these numbers they decide to calculate which days would have been the most profitable to buy and sell their shares, so they can compare that with their actual gain.

Your task is to write a program that computes the optimal «run» in the sequence of numbers, the subsequence that maximizes the profit. The subsequence you compute is represented by the 1-based indexes of the first and last numbers in the subsequence. We are asking for exactly one date to buy and one date to sell shares since otherwise the solution would be simple: keep your shares on dates that the profit is non-negative.

Input

Input file has the following format:

- One line with a single integer N, $(1 \leqslant N \leqslant 10^6)$ — the length of the sequence to follow.
- One line with N integers p_i $(-1000 \leqslant p_i \leqslant 1000)$ — the profit (or loss) on date i.

The integers are separated by single spaces. At least one of the integers is positive.

Output

Output file contains a single line with two integers k and l $(1 \leqslant k \leqslant l \leqslant N)$, such that the sum of the k-th until the k-th integer is maximized, boundaries included. When k and l can be selected by different ways, choose minimal k and minimal l.

Example

input.txt	output.txt
11 -3 1 -1 2 3 1 -1 2 -3 -5 7	2 8
9 1 -2 3 -1 -1 3 -2 2 -4	3 6

Problem 14. Dimensions (Division 2 Only!)

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Teacher has given his pupils a table of widths, heights, lengths and volumes of some cuboid. Each row on the table has one of the 4 values missing; the pupils — and your program — have to work out the missing value and write it in the table such that the values on each line represent the width, height, length and volume of one cuboid.

Input

Input is a series of lines, each containing 4 integers: w, h, l and v representing the width, height, length and volume of a cuboid in that order. The integers are separated by a single space. $0 < l, w, h < 100, 0 < v < 10^5$. In each row, one of the values has been replaced by a zero. The final row contains 0 0 0 0 and should not be processed.

Output

Output is the same series of lines but with the zero in each line replaced by the correct value for length, width, height or volume as appropriate. It is guaranteed that the new value is always an integer.

Example

input.txt	output.txt

1 0 2 6
5 5 5 0
0 2 2 80
8 0 9 576
0 0 0 0
	1 3 2 6
	5 5 5 125
	20 2 2 80
	8 9 9 576

Problem 15. ePad (Division 2 Only!)

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

«ePad» is new gadget from Elppa Inc. Every ePad has a unique serial number which is an integer between 1 and 10^5. Elppa inc. can sell several ePad's with discount, if that the sum of all the serial numbers of those ePad's is a multiple of some positive integer M. Given all serial numbers of ePad's on a stock and the number M, calculate the maximum number of ePad's that could have been sold from this stock with discount.

Input

Input file has the following format:

- One line with two integers N and M ($1 \leqslant N \leqslant 500,\ 1 \leqslant M \leqslant 10^5$) — the total number of ePad's on the stock and the «magic» number M.
- One line with N different integers $S_1, \ldots, S_N$ ($0 \leqslant S_i \leqslant 10^5$) — the serial numbers of ePad's on the stock.

Integers on the same line are separated by single spaces, and there will always be at least one subset of ePad's such that the sum of their serial numbers is a multiple of M .

Output

The output file should contain a single number, on a single line: the maximum number of ePad's from described stock, such that the sum of their serial numbers is a multiple of M.

Example

input.txt	output.txt
3 5 1 8 6	3
6 9 8 6 4 1 2 3	5

Problem 16. Farmer (Division 2 Only!)

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 Mebibytes

Farmer owns several horses that graze in the fields and walk around happily. However, one horse is too lazy and always takes the shortest route to the barn to get food. Farmer wants to give her some more walking exercises, so he placed some extra fences to make sure that lazy horse cannot always take the shortest route and has to walk around the fences.

Given the current location of lazy horse, the location of the barn with the food, and all locations of the fences (modelled as line segments), farmer wants you to compute the minimum distance that lazy horse has to walk. Horses are not allowed to cross any fence on her route, but they are allowed to touch the fences.

Input

Input file has the following format:

- One line with four integers B_x, B_y, F_x and F_y ($-10^4 \leqslant B_x, B_y, F_x, F_y \leqslant 10^4$) — the location of lazy horse and the location of the food.
- One line with one integer N ($0 \leqslant N \leqslant 100$): the number of fences.
- N lines, one for each fence, with four integers x_1, y_1, x_2, y_2 ($-10^4 \leqslant x_1, y_1, x_2, y_2 \leqslant 10^4$) and $x_1 = x_2$ or $y_1 = y_2$ — the x and y coordinates of the begin and end points of this fence.

Integers on the same line are separated by single spaces. The current location of lazy horse and the location of the food will not lie on a fence, and fences will not touch or overlap.

Output

Output file should contain a single real number, rounded to six digits after the decimal point, on a single line: the minimum walking distance for the lazy horse.

Example

input.txt	output.txt

`0 0 10 0`
`1`
`-1 -100 -1 100`
	`10.0`

`0 0 10 0`
`2`
`-1 -100 -1 100`
`5 4 5 -6`
	`12.806248`

`0 0 2 1`
`1`
`1 1 1 -1`
	`2.414214`