# List of Problems

| Problem | Time limit | Memory limit | Name | Files |
|---|---|---|---|---|
| 1 | 2 seconds | 256 MiB | Heroes 2 (Division 1 Only!!) | input.txt or stdin, output.txt or stdout |
| 2 | 2 seconds | 256 MiB | Gandhi and Dundee | input.txt or stdin, output.txt or stdout |
| 3 | 2 seconds | 256 MiB | Chupacabras vs mothmen | input.txt or stdin, output.txt or stdout |
| 4 | 2 seconds | 256 MiB | A night in the Library | input.txt or stdin, output.txt or stdout |
| 5 | 2 seconds | 256 MiB | Plotter (Division 1 Only!!) | input.txt or stdin, output.txt or stdout |
| 6 | 2 seconds | 256 MiB | Mixing Liquids | input.txt or stdin, output.txt or stdout |
| 7 | 2 seconds | 256 MiB | Christmas Lights Assembly | input.txt or stdin, output.txt or stdout |
| 8 | 2 seconds | 256 MiB | Pattern Matching | input.txt or stdin, output.txt or stdout |
| 9 | 2 seconds | 256 MiB | Tree (Division 1 Only!!) | input.txt or stdin, output.txt or stdout |
| 10 | 2 seconds | 266 MiB | Rally | input.txt or stdin, output.txt or stdout |
| 11 | 2 seconds | 256 MiB | Bug Preprocessor (Division 2 Only!!) | input.txt or stdin, output.txt or stdout |
| 12 | 2 seconds | 256 MiB | Pills (Division 2 Only!!) | input.txt or stdin, output.txt or stdout |
| 13 | 2 seconds | 256 MiB | Flying Slowpoke (Division 2 Only!!) | input.txt or stdin, output.txt or stdout |

Unless explicitly stated in the problem statements, all input elements may be separated by an arbitrary number of whitespace characters. All input data are correct and satisfy the specification given in the problem statement.

The output of your program must exactly satisfy the output specification in the problem statement.

Each line of input is terminated by end-of-line marker (character with code 0A hex).

## **Problem 1.** Heroes 2 (First division only)

Not long ago the game «Heroes of Keyboard and Mouse 2» was released. The 4D-Action essence be the following: A number of cities are placed in a four-dimensional space. Every city is located in a certain point. The player can build new cities. And rob "corovans".

At all times all cities are surrounded by a single connected wall of a finite size. The wall splits the game space into two parts: everything that's within it and all that's outside. The wall is built so that:

1. All the cities are within the wall.
2. There is a straight path between any two points within the wall, not crossing the wall.
3. The set of points within the wall is minimal as long as the conditions 1 and 2 are met.

When a new city is built, the wall has to be rebuilt if the city lies outside it. Your task is to define whether wall reconfiguration is necessary for every newly built town.

It is guaranteed that a new city is always built either outside of the existing wall or within it. Moreover, the distance from the new city to the wall is always greater than $10^{-3}$. No two cities occupy the same point.

### Input

The game begins with five cities with the coordinates $(x_1, y_1, z_1, w_1)$, $(x_2, y_2, z_2, w_2)$, …, $(x_5, y_5, z_5, w_5)$, which are defined in the first five lines of the input file. The initial 4D volume within the wall is strictly positive.

The second line contains an integer N – the number of newly built cities ($1 \le N \le 800$). Each of the following N lines contains four integers – the coordinates of a newly built city. The absolute value of each coordinate does not exceed 5000.

### Output

The output file must contain N lines. The K-th line must contain the word **`Rebuild`**, if wall reconfiguration is necessary after building K-th town, and the word **`Ignore`** otherwise ($1 \le K \le N$).

### Example

| input.txt | output.txt |
|---|---|
| `0 0 0 0`<br>`8 0 0 0`<br>`0 8 0 0`<br>`0 0 8 0`<br>`0 0 0 8`<br>`5`<br>`1 2 2 2`<br>`2 2 3 2`<br>`8 8 8 8`<br>`3 5 3 4`<br>`-1 3 7 2` | **`Ignore`**<br>**`Rebuild`**<br>**`Rebuild`**<br>**`Ignore`**<br>**`Rebuild`** |

### Comments

Initially five cities are placed at the vertices of a coordinate simplex with the length 8 along the axes. The wall is exactly the boundary of the simplex. The equation of the large hyperplane of the simplex is the following: $x + y + z + w = 8$.

As it can be seen from the equation, the first newly built city belongs to the simplex and doesn't require wall reconfiguration, and the second city lies outside the simplex and the wall must be rebuilt. Once the wall has been rebuilt, the set of interior points consists of two adjacent simplexes. When the third city is built, the wall is reconfigured, and after that the set of interior points also consists of two simplexes. The fourth city belongs to this set, and the fifth apparently lies outside of it.

## **Problem 2.** Gandhi and Dundee

Gandhi went to Dundee. Gandhi is a handful, don't mess with Gandhi. Gandhi will eat whatever he can reach on his way. He can reach all points which are within his eating radius from a certain point on his way. He's taking the shortest route from the point A to Dundee, which is, on planet Earth, is a large-diameter arch passing through his start and end points. The arch is not larger than a half of the big circle.

The distance to the edible points is measured along the surface of the Earth. We must find the surface area of the territory where no decent person should go to avoid being eaten by Gandhi.

### Input

The first line of the input file contains six real numbers $\mathbf{x}_0$, $y_0$, $x_1$, $y_1$, R, eps — the latitude and longitude of the beginning point of the interval (in degrees) and, correspondingly, of the ending point, as well as the radius of Earth and Gandhi's eating radius ($-90 \le x_0, x_1 \le 90$, $-180 \le y_0, y_1 \le 180$, $1 \le R \le 10000$, $0 \le eps \le R$).

### Output

The output file must contain a single real number – the surface area of the dangerous territory with absolute or relative error within $10^{-8}$.

### Example

| input.txt | output.txt |
|---|---|
| `-90 5 90 -100 10 5` | `378.1490948518` |

## **Problem 3.** Chupacabras vs mothmen

The arena is a square field with N×N cells. Chupacabras and mothmen are positioned on the grid. Each of the chupacabras and mothmen has "power" defined by a nonnegative integer; the chupacabras have even power numbers and the mothmen have odd numbers. Chupacabras always move from right to left, and mothmen always move from left to right. When a chupacabra collides with a mothman, the stronger one wins, and the weaker one disappears from the arena.

A chupacabra cannot jump over or step on another chupacabra; the same goes for two mothmen. Define how many chupacabras and mothmen will remain on the arena when no further motion is possible.

### Input

The first line of the input file contains a single integer N ($1 \leq N \leq 100$).

The following N lines each contain N numbers separated by spaces. The j-th symbol of the i-th line denotes the "power" of a chupacabra or a mothman situated in the cell with the coordinates `(i, j)`, or equals **`-1`** if the cell is empty. The values of numbers denoting power are equal or lower than 100.

### Output

The output file must contain two numbers separated by a space — the number of the remaining chupacabras and mothmen, correspondingly.

### Example

| input.txt | output.txt |
|---|---|
| 4<br>5 -1 4 -1<br>3 -1 1 -1<br>-1 2 -1 4<br>-1 -1 -1 -1 | 2 3 |

## **Problem 4.** A night in the Library

One summer Jack had “working practice” in the school library: he was fixing old textbooks, helping the librarians with sorting books, etc. Jack always sticks his snout into everything, and math textbooks were no exception.

In one of the books he saw the following:

… denoted using D(M) — a skew-symmetric polylinear normed function of square matrix columns, i.e.:

1. If two columns of the matrix are swapped, its value changes sign:
$$D([C_1, C_2, \dots, C_i, \dots, C_j, \dots, C_n]) = -D([C_1, C_2, \dots, C_j, \dots, C_i, \dots, C_n])$$
2. If a matrix column is a linear combination of two vectors, the linear combination can be taken out:
$$D([C_1, C_2, \dots, C_{i-1}, \alpha A + \beta B, C_{i+1}, \dots, C_n]) \\ = \alpha\, D([C_1, C_2, \dots, C_{i-1}, A, C_{i+1}, \dots, C_n]) + \beta\, D([C_1, C_2, \dots, C_{i-1}, B, C_{i+1}, \dots, C_n])$$
3. On the identity matrix it equals 1:
$$D(E) = D([e_1, e_2, \dots, e_n]) = 1$$

Let us prove that such a function is unique:

…(Jack skips pages and reads in small bits) …

… for example using Gaussian elimination.

… (very small bits) …

… the matrix X is the product of the column u and the row v, i.e.:
$$X_{ij} = u_i v_j$$

Jack is very intrigued by what he’s just read. Jack enjoys competitive programming and spends all his time on the Codeforces website, so he’s not at all afraid of matrices and vectors. But his math could use some improvement.

Jack wants to calculate the value of the mentioned skew-symmetric polylinear normed function D for the matrix X defined in the last bit of text. To make things easier all elements of the vectors u and v are integers. He has a hunch that the answer is an integer too. Since he has a feeling that it can be very, very big, Jack wants to find it modulo prime number P (that’s how things are usually done).

Help Jack solve the task. Otherwise the aforementioned website will be spammed.

### Input

The first line contains two integers N and P ($1 \le N \le 100$, $2 \le P \le 2 \cdot 10^9$). It is guaranteed that P is a prime number. The second line contains N integers $u_1, u_2, \dots, u_n$ — elements of the column vector u, and the third line contains N integers $v_1, v_2, \dots, v_n$ — elements of the row vector v ($0 \le u_i, v_j < P$).

### Output

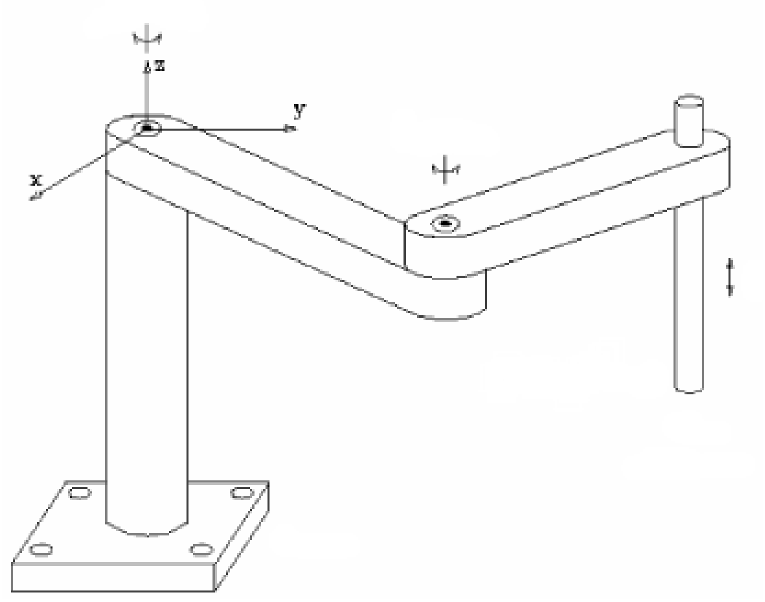
The output file must contain a single integer: the value of D(X) modulo P. The number must be non-negative and strictly less than P.

### Example

| input.txt | output.txt |
|---|---|
| 3 11<br>0 7 8<br>10 5 3 | 0 |

## **Problem 5.** Plotter (First division only)

Pete's parents got him the Mindrack 2.0 kit for building robots. Pete read the manual and decided to build a simple plotter to draw shapes. The plotter is a mechanical arm consisting of two segments with a rotary joint and a slot for a pencil in the distal end. The arm is attached to a base using a joint too. Additionally the arm has servos that can rotate it relative to the base and turn segments relative to each other. The arm can draw various shapes on paper. Below is a schematic illustration:



Pete wrote a program that allows drawing various closed polylines based on a given set of points by means of coordinated actions of the servos. The arm can draw a polyline without taking the pencil off the paper. The only problem Pete has encountered is the limited mobility of the arm segments. The basal joint can turn no more than $\varphi$ degrees left and right relative to its central position and the joint connecting the segments cannot turn more than $\theta$ degrees relative to the codirectional alignment of the segments. For that reason a situation can occur during drawing shapes when the robotic arm is unable to continue drawing without lifting the pencil from paper. In this case Pete has to disconnect the pencil manually, switch the arm into a new configuration and reconnect the pencil. The arm then continues drawing the polyline from the same point where it has ended drawing before manual switching. Naturally, Pete wants to minimize the amount of such manual switches.

Let the arm be positioned in the Oxy reference plane and attached to its origin. Then the central position of the arm is directed along the Oy axis. Assume the arm consists of two segments with the lengths L and l, corresponding to the basal segment attached to the base and distal segment holding the pencil. Given the parameters of the arm – the numbers L, l, $\varphi$ and $\theta$, and coordinates of polyline points on the plane you have to calculate the minimal number of manual arm switches necessary to draw the polyline or determine that it is impossible. The arm can begin drawing from any point of the polyline and in any of the two directions. However, once the start point and the drawing direction have been selected the latter cannot be changed. In case of the switch, the arm must continue drawing from the same point where switching has occurred and in the same direction as before. If the polyline has self-intersections or self-contacts, then the arm is not allowed to switch to another polyline part at such points, it still must draw polyline segments in the prescribed order taking into account the chosen direction and starting point.

## Input

The first line of the input file contains four nonnegative numbers L, l, $\varphi$ and $\theta$ ($1 \le L + l \le 10^4$, $\varphi + \theta \le 180$). The next line contains an integer N — the number of points in the closed polyline ($3 \le N \le 10^5$). The next N lines contain pairs of integers $x_i$ and $y_i$ — Cartesian coordinates of the polyline points. The absolute values of coordinates do not exceed $10^4$. No two points coincide. It is assumed that the polyline consists of the segments $(x_1, y_1)$–$(x_2, y_2)$, $(x_2, y_2)$–$(x_3, y_3)$,…, $(x_{N-1}, y_{N-1})$–$(x_N, y_N)$ and $(x_N, y_N)$–$(x_1, y_1)$.

## Output

The single line of the output file must contain the minimal number of manual switches of the arm necessary to draw the given polyline, or the number **`-1`** if it is impossible to draw the polyline using the method described above.

## Example

| input.txt | output.txt |
|---|---|
| `4 4 90 90`<br>`6`<br>`-6 2`<br>`-3 6`<br>`3 6`<br>`6 2`<br>`3 7`<br>`-3 7` | 1 |
| `1 1 45 45`<br>`3`<br>`1 1`<br>`5 1`<br>`1 5` | -1 |

## **Problem 6**. Mixing liquids

Drinking tea and other liquids (purely for the sake of conspiracy) is a vital part of the sittings of the Secret Committee. Hot tea is diluted with cold milk. Of course, everyone wants his tea to reach the perfect temperature as fast as possible. The temperature of the mixture is measured according to the exponential law:

$$T(t) = T_0 + (T_1 - T_0)e^{-kt},$$

where T(t) is the temperature at the moment of time t,

$T_1$ — temperature of the mixture at the initial moment,

$T_0$ — ambient temperature,

k — a certain static coefficient.

When two liquids are mixed with the masses $m_1$ and $m_2$ and temperatures $T_1$ and $T_2$ respectively the result is a mixture with the temperature

$$T_1' = \frac{T_1 \cdot m_1 + T_2 \cdot m_2}{m_1 + m_2}$$

Let us consider that all liquids at the tea party have the same specific thermal capacity. The temperature of the first liquid begins to change according to the exponential law straight away, and at any moment of time we can mix it instantly with the second liquid, which has a stable temperature $T_2$, after which the temperature of the resulting liquid begins to change according to the same law with the same coefficient k and new temperature $T_1'$ calculated from the formula of temperatures of mixed liquids. We must calculate the minimal time necessary to achieve the desired temperature of the resulting mixture.

### Input

The first line of the input file must contain an integer N — the number of tests. Each line of the following N lines contains seven integers $T_0$, $T_1$, $T_2$, $m_1$, $m_2$, $T_{opt}$, k. $T_0$, $T_1$, $T_2$, $T_{opt}$ are the temperatures of the ambience, of the first and second liquids and the desired temperature, ($-273 < T_0, T_1, T_2, T_{opt} \le 1000$), $m_1$ and $m_2$ are the masses of the first and second liquids ($0 \le m_1, m_2 \le 1000$, $m_1+m_2 > 0$), and k is the coefficient for the speed of temperature change ($1 \le k \le 1000$).

### Output

Each of N lines according to the sequence of data in the input file must contain a single real number — the minimal time necessary to achieve the desired temperature calculated with a relative or absolute error margin not exceeding $10^{-8}$ or the message `Impossible`, if the desired temperature cannot be achieved.

### Example

| input.txt | output.txt |
|---|---|
| 2<br>0 1 1 10 10 1 1<br>0 1 1 10 10 0 1 | 0<br>Impossible |

## **Problem 7.** Christmas Lights Assemby

Peter bought Christmas lights in a do-it-yourself assembly kit.

The box contains:

- 2·N differently colored light bulbs, no two bulbs are the same color;
- 2·N sockets for the bulbs;
- enough wires to connect the sockets and bulbs to the electric network.

Each socket has a number of slots for connecting wires. The box comes with N types of sockets. The first type has one slot, the second type has two slots, the third has three, … , and so on. The N type has N slots. There are two sockets of each type.

The Christmas lights works when bulbs are screwed into sockets and sockets are wired so that the following conditions are met:

- all slots of all sockets must be wired;
- one slot may contain only one wire;
- each wire must be connected to two different sockets;
- two sockets may not be connected by more than one wire.

This task is too difficult for Peter. Help Peter make his Christmas lights work.

### Input

The input file contains an integer N — the number of socket types ($1 \le N \le 500$).

### Output

The first line of the output file must contain an integer M — the number of wires necessary to assemble the kit. The following M lines must each contain two integers separated by spaces — the numbers of light bulbs screwed into sockets connected by a single wire. The bulbs are numbered from 1 to 2·N. If more than one solution is possible, output any.

### Example

| input.txt | output.txt |
|---|---|
| `3` | `6`<br>`1 2`<br>`1 3`<br>`2 4`<br>`1 5`<br>`2 6`<br>`5 6` |

## Problem 8. Pattern matching

String contains characters of two disjoint alphabets $A_1$ and $A_2$.

Two strings are considered equal if there is a one-to-one mapping of the alphabet A2 to itself which, when applied to the first string, produces the second string.

Under this definition of string equality we say that for a given pattern there is its occurrence in a string, if there is a substring in a string equal to the pattern.

For a given string and a pattern count the number of occurrences of the pattern in the string.

### Input

The first line of the input file contains two integers N and M — the number of characters in the first and second alphabets, respectively ($1 \le N, M < 52$).

The following two lines contain characters from the alphabets $A_1$ and $A_2$ without spaces. Characters from the alphabets are Latin, lowercase or capitalized. The fourth line contains the string, and the fifth contains pattern. The lengths of the string and pattern do not exceed $10^5$.

### Output

The output file must contain a single integer — the number of times the pattern appears in the string.

### Examples

| input.txt | output.txt |
|---|---|
| `8 4`<br>`aIbctlyd`<br>`eouh`<br>`Itellyeh`<br>`you` | `1` |
| `2 6`<br>`cq`<br>`xyztab`<br>`qxycxycztc`<br>`abc` | `3` |

## **Problem 9.** Tree (First division only)

Given: an ordered rooted tree T and a list of its modifications. The modifications are the following: cut off node with the given number along with the subtree from its current parent node and add it as a rightmost child node to the other node.

You must determine the modification after which the tree becomes isomorphic to one of the previous trees.

In a rooted ordered tree each non-leaf node has an established order of child nodes. Two trees are considered isomorphic if the following is retained in bijective mapping:

− the parent-child relation, as in «the node s is a child of the node f»,
− the order of children for every node.

### Input

The first line of the input file contains two integers: N is the number of nodes in the tree and M is the number of modifications ($2 \le N \le 100000$, $1 \le M \le 100000$). The tree nodes are numbered by integer numbers from 1 to N. The root node has the number 1.

The following N lines contain ordered lists of child nodes for the nodes of the initial tree. Each i-th list is described by the number $K_i$ of children of the i-th node and by the sequence of its child node numbers ($0 \le K_i \le N - 1$).

The remaining M lines contain the description of the tree modifications. Each modification is defined by two integers $S_j$ and $F_j$ – the number of the node being cut from the parent along with its subtree and the number of the node to which it is added as its rightmost child ($2 \le S_j \le N$, $1 \le F_j \le N$). It is guaranteed that the node $F_j$ is not a descendant of the node $S_j$ and does not coincide with it.

### Output

If all M + 1 resulting trees are different, the output file must contain the number `-1`.

Otherwise it must contain two integers A and B separated by a space — the numbers of two isomorphic trees ($0 \le A < B \le M$). If there are several pairs of isomorphic trees, the output file must contain the pair with the lowest number B.

Trees are numbered in the following way: the initial tree has the number 0, and j-th modification results in j-th tree ($1 \le j \le M$).

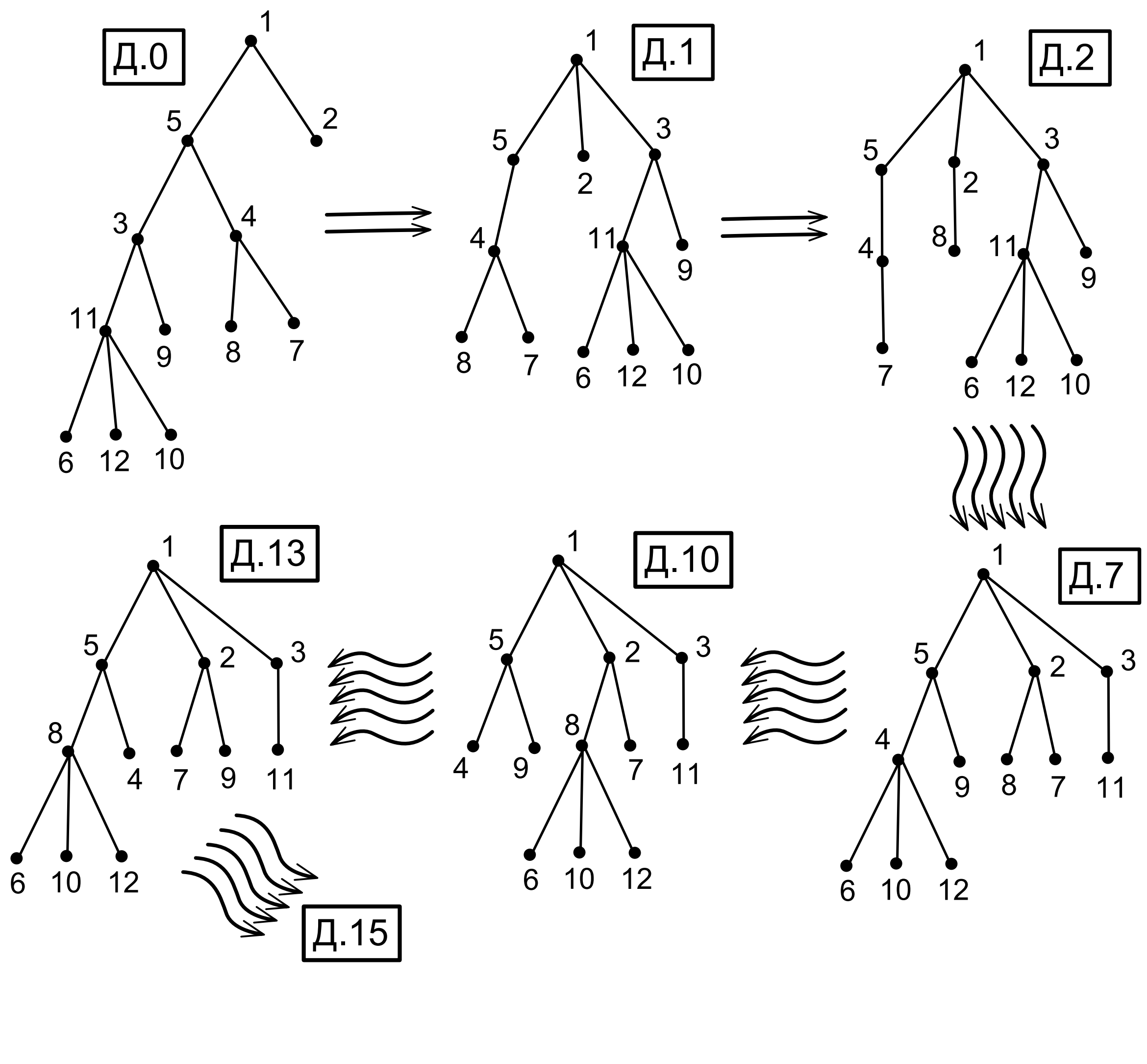
### Examples

| input.txt | output.txt |
|---|---|
| `12 15`<br>`2 5 2`<br>`0`<br>`2 11 9`<br>`2 8 7`<br>`2 3 4`<br>`0`<br>`0`<br>`0`<br>`0`<br>`0`<br>`3 6 12 10`<br>`0`<br>`3 1`<br>`8 2`<br>`7 2`<br>`6 4`<br>`10 4` | `7 13` |

| | |
|---|---|
| 12 4<br>9 5<br>6 8<br>10 8<br>12 8<br>8 5<br>9 2<br>4 5<br>6 9<br>6 8 | |
| 4 3<br>3 2 3 4<br>0<br>0<br>0<br>2 3<br>3 1<br>4 2 | -1 |

**Comment**

Below are some of the trees for the first test from the given data.



## **Problem 10** . Rally

The route of Antelope Gnu lies through hospitable and generous places. There is a certain chance that an important telegram arrives to any of the towns on their route, or that Panikovskiy lapses into his old ways, meaning the crew will have to spend an extra day in that town. Naturally, the time necessary to cover the same part of the route is never the same and depends on pure luck. However, the team must arrive to the glorious town of Chernomorsk as soon as possible, well, perhaps not 100% on time, but with a pre-defined probability of it. At the same time the chance of delays in any of the towns on their route is the same, and the delays don't depend on each other. Let the route duration be T such that the chance of covering the route in time not exceeding T is not smaller than the given probability P. The time of covering the route includes possible delays in the first and last towns lying on the route.

Help the team find the route with the smallest duration.

### Input

The first line of the input file contains two integers N and M — the number of towns and the number of roads between them, and two real numbers P – the predefined probability used to determine the route duration and $P_1$ — the probability of 24-hour delay in every town ($2 \le N \le 1000$, $1 \le M \le 10000$, $0 \le P, P_1 \le 1$). P and $P_1$ are given with five digits after the decimal point.

The following M lines contain the descriptions of roads with three integers $A_i$, $B_i$, $L_i$ each: the numbers of towns linked by a given road and time necessary to cover that distance in hours ($1 \le L_i \le 1000$). The roads are bidirectional. Towns are numbered from 1 through N. The route is set from the town with the number 1 to the town with the number N. No pair of towns is connected with more than one road. No road connects a town to itself. It is guaranteed that there is a path between any pair of towns and that the duration of any route doesn't change if P is changed by not more than $10^{-9}$ in any direction.

### Output

The first line of the output file must contain a single integer — the number of towns on the optimal route. The second line must contain the numbers of these towns in the order of passing, separated by spaces.

### Example

| input.txt | output.txt |
|---|---|
| `4 4 0.99000 0.50000`<br>`1 2 1`<br>`2 3 1`<br>`3 4 1`<br>`1 4 4` | `2`<br>`1 4` |

# Problem 11. Bug Preprocessor (Division 2 Only!)

Recently, there appeared a promising open-source initiative called the Bug Preprocessor. The preprocessor is a program able to find all bugs in your source code and mark them, so they are relatively easy to remove. Your task is to write a program that will remove all marked bugs from the preprocessed source code.

## Input

The input contains several (10 or less) test cases. Each test case starts with a line containing one integer $T$ ($0 \leqslant T \leqslant 1000$), one space and a string $B$ used by the preprocessor to mark all bugs. The next $T$ lines then contain the preprocessed source code. All bugs are represented by a case-sensitive string $B$.

Each line of the input will be between 0 and 200 characters long. The bug marker $B$ consists of at least 1 and at most 10 uppercase letters (‘`A`’ through ‘`Z`’).

## Output

Your program must remove all of the bugs from the input and print a text that does not contain any occurrence of $B$. Nothing else than bugs may be removed, not even spaces.

## Example

| input.txt | output.txt |
|---|---|
| `7 BUG` | `print "No bugs here..."` |
| `print "No bugs here..."` | |
| | `void hello() {` |
| `void hello() {` | |
| `BUGBUG` | `printf("Hello, world!` |
| `printfBUG("Hello, world!` | `n);` |
| `n);` | `}` |
| `}` | |
| | `writeln("Hello E-R-R");` |
| `1 ERR` | |
| `wriEERRRRtelERRn("Hello E-R-R");` | |

# Problem 12. Pills (Division 2 Only!)

Aunt Polly takes half a pill of a certain medicine every day. She starts with a bottle that contains $N$ pills.

On the first day, she removes a random pill, breaks it in two halves, takes one half and puts the other half back into the bottle.

On subsequent days, she removes a random piece (which can be either a whole pill or half a pill) from the bottle. If it is half a pill, she takes it. If it is a whole pill, she takes one half and puts the other half back into the bottle.

In how many ways can she empty the bottle? We represent the sequence of pills removed from the bottle in the course of $2N$ days as a string, where the $i$-th character is $W$ if a whole pill was chosen on the $i$-th day, and $H$ if a half pill was chosen ($1 \leqslant i \leqslant 2N$). How many different valid strings are there that empty the bottle?

## Input

The input will contain data for at most 1000 problem instances. For each problem instance there will be one line of input: a positive integer $N \leqslant 30$, the number of pills initially in the bottle. End of input will be indicated by 0.

## Output

For each problem instance, the output will be a single number, displayed at the beginning of a new line. It will be the number of different ways the bottle can be emptied

## Example

| input.txt | output.txt |
|---|---|
| 6<br>1<br>4<br>2<br>3<br>30<br>0 | 132<br>1<br>14<br>2<br>5<br>3814986502092304 |

# Problem 13. Flying Slowpoke (Division 2 Only!)

Slowie knows how to fly. Despite this, whenever Slowie wants to move from one point to another, it becomes a tedious task for him. The main trouble is that Slowie is an flying slowpoke.

And it is a well-known fact that all flying slowpokes are clumsy and slow. Not only they need some time to fly along a straight line, they also spend more time making turns. Knowing their limitations, will you help Slowie to find the time to fly over its route?

At the beginning, Slowie is facing north, which is the positive direction of Y-axis. So, he first needs to turn by $D$ degrees ($0 \leqslant D \leqslant 180$) to an appropriate direction, then he can fly over a straight path to the final point.

## Input

The input consists of several (150 or less) instances. The first line of each instance contains integers $S$ and $T$ ($1 \leqslant S, T \leqslant 1000$), where $S$ is SlowieвЂ™s speed in meters per second, and $T$ is the speed of him turning in degrees per second. The second line contains four integers ($0 \leqslant X_f, Y_f, X_t, Y_t \leqslant 10^4$) indicating the starting point $(X_f, Y_f)$ and the destination $(X_t, Y_t)$. All coordinates are given in meters.

## Output

For each input instance, print a single line containing one real number $R$, giving the shortest time Slowie needs to get from the initial to the final point.

The answer will be accepted as correct if the difference between $R$ and the answer computed by the judges is at most 0.001.

## Example

| input.txt | output.txt |
|---|---|
| 3 10<br>0 0 0 10<br>3 10<br>0 0 10 10 | 3.3333<br>9.2140 |