MIT Programming Contest
Team Contest 1 Problems
2015

September 27, 2015

1 Class

Mr. Carrillo has decided to break up the students in his high school chemistry class into groups for team projects. He knows that some students in the class do not get along, so he will not put those students in the same team. Furthermore, Mr. Carrillo noticed a pattern: each student does not get along with the student whose time of birth is closest to theirs in the future (no two students have exactly the same time of birth). Also, Mr. Carillo does not have a restriction on how many teams he can make (each student could potentially be in a team by themselves), and he has no restriction on the size of each team.

From running this team activity in previous years, Mr. Carrillo knows that a team performs as badly as its worst student (that student tends to goof off and cause everyone else in their team to goof off as well). Each student has a *badness factor* which is some non-negative integer, and the badness of a team is the largest badness factor of all students in that team. Mr. Carrillo would like to assign students to teams such that the cumulative badness across all teams is as small as possible. Help him accomplish this goal.

Input

The first line contains an integer N, the number of students ($1 \leq N \leq 1000$). The next N lines give the badness factors of each of the N students, in order from the oldest student to the youngest. Each badness factor is an integer between 0 and 10^8.

Output

On a single line output the minimum possible cumulative badness Mr. Carrillo can achieve.

Example

Sample Input	Sample Output
3 1 2 3	5

2 Explosions

Unlucky scientist Vasya was tired of permanent mockeries of his colleagues. He built a time machine to go away into the future. But his machine had taken him into the past and then exploded!

Vasya needed some way to earn money! He has taken the job of the Court Alchemist of King Arthur. His first task is to create the philosophers' stone. Vasya has found K ingredients and now conducts experiments which consist in mixing these ingredients in various proportions. More formally, each experiment consists in taking a_1 ounces of the first ingredient, a_2 ounces of the second and so on. Then all the ingredients are put into the crucible, carefully mixed and heated. All a_i are non-negative integers, and their sum does not exceed S — the capacity of the crucible. In all Vasya's experiments at least two ingredients are taken, i.e. there are at least two a_i greater than zero.

But all his experiments have been unsuccessful up to now. All mixtures have exploded! The King is unpleased. Maids of honour complain to Queen Guinevere about these awful explosion sounds! The King has decided to give one last chance to Vasya. Tomorrow Vasya is to conduct his next experiment, which could become the last. If the mixture explodes again, Vasya will be executed.

Fortunately, Vasya has just found a way to discard some definitely unsuccessful experiments. He has noted that if for some experiment plan $(a_1, a_2, \ldots, a_K)$ the mixture has exploded, it would explode for any plan $(b_1, b_2, \ldots, b_K)$ such that $b_i \geq a_i$ for all i. Now Vasya wants to count the number of possible experiments which are not definitely unsuccessful according to his unfortunate experience. He has taken out his pocket computer and now he is writing a program which will compute this number. Can you do this?

Input

The first line of input file contains three positive integers $2 \leq K \leq 30$, $2 \leq S \leq 10000$, $0 \leq M \leq 20$ where M is the number of experiments Vasya has already conducted. The next M lines contain K numbers each, one line per experiment.

Output

Output the number of possible experiment plans for which it is not evident that the mixture would explode.

Example

Sample Input	Sample Output
2 4 2 1 3 2 1	2

This page was intentionally left blank.

3 Lamps

Along the rocky road to Dublin, there are N lamps. Irish engineers know the position of each lamp (measured as the distance from Dublin in meters), the distance to which it can throw light on both sides, as well as its power consumption (in watts). They want to illuminate the part of the road from position A to position B by switching on only some lamps, such that the power consumption is minimized. It does not matter if other parts of the road are illuminated. There are actually multiple rocky roads to Dublin, and they want to solve the problem for each of them. As their descendant, they ask you to make a program that will find the solution.

Input

The first line of the input contains integer K, which is the number of roads ($1 \leq K \leq 10$). K road descriptions follow. Description of each road begins with an empty line. The second line of the description contains three integers N, A and B separated by a space character ($1 \leq N \leq 10^5$, $-2 * 10^9 \leq A < B \leq 2 * 10^9$). Each of the next N lines of the description contain three integers x, d, w, separated by a space character that describe one lamp ($-10^9 \leq x \leq 10^9$, $0 \leq d \leq 10^9$, $0 \leq w \leq 20000$). x is the position of the lamp, d is the distance it can illuminate on both sides, and w is its power consumption. Note that positions can be negative (since the road continues on both sides of Dublin, and the position is measured relative to it).

Output

In each of the K lines of the output Write one integer P followed by a newline, which is the minimum power consumption required to illuminate the part of the road from position A to position B, if it is possible to do that, or -1 otherwise. The order of output lines correspond to the order of road descriptions in the input.

Example

Sample Input	Sample Output
2 6 -10 20 -8 4 2 25 4 7 3 10 1 0 15 4 17 3 5 16 4 2 3 4 20 8 5 10 8 4 7 19 4 10	5 -1

4 The (Bayesian) Hound and the Hare

Researchers have discovered that hounds are extremely intelligent—so intelligent, in fact, that they use Bayesian statistics to hunt down their prey.

As an experiement, researchers studying these hounds blindfold one and put it in a maze laid out on a square grid. They put a hare somewhere else in the maze. Once every second, the hare takes a step one unit in a random direction (no diagonals), chosen uniformly at random among the directions that aren't blocked by walls. Sometimes, when the hare takes a step, it makes a little bit of noise at its *new* position. The hound hears the noise with a certain amount of uncertainty σ—that is, if the hare is at position (x, y), then the hound thinks that it heard the hare at position (x', y') with probability $N \exp\left(((x - x')^2 + (y - y')^2)/(2\sigma^2)\right)$, where N is an appropriate normalization factor.

After every step the hare takes, the hound moves one unit in the direction (North, East, South, or West, or not moving at all) that mininizes the expected shortest-path distance to the hare. In case of a tie, the hound favors not moving, then North, then East, then South, then West. (That is, the hound first considers whether to stay go West; then, if the expected distance if it goes South is better or within 10^{-5}, the hound considers going West; then South; then North; then staying still.)

Neither the hound nor the hare can walk into a wall or on a diagonal, and the hound knows this. The hound has a perfect memory and knows the exact layout of the maze. The hound assumes that the hare starts at a uniform random location within the maze. The hound assumes that it could occupy the same space as the hare without realizing it.

Write a program that determines what path the hound takes given a sequence of observations.

Input

The maze has dimension X units West to East by Y units North to South. The first line of input is two numbers, X and Y. You may assume that $3 \leq X, Y \leq 100$. This is followed by a map of the maze. The first line corresponds to the northernmost strip of the maze, and the first column is the western side. A "`#`" indicates a wall, a "`.`" indicates open space, and a "`h`" marks the starting position of the hound. The sides of the maze are always walls, and the interior of the maze is connected.

The map of the maze is followed by σ, an integer, on one line. You may assume that $\sigma \geq 1$.

The remainder of the input is a series of observations, one line per second. A line that contains "`silence`" indicates that the hound hears nothing, and a line that contains two integers gives the coordinates—column, row where northwest corner of the maze is (0,0) in which the hound thinks it heard the hare. Note that the coordinates may be outside the maze.

Output

For each observation, output "`N`", "`S`", "`E`", "`W`", or "`.`" if the hound goes North, South, East, West, or stays still, respectively, after each observation.

Example

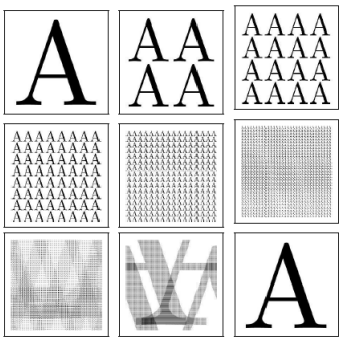
Sample Input

```
20 20
####################
####################
####.....h.....#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.#########.#####
####.....h.....#####
#########.##########
#########.##########
#########.##########
#########.##########
####################
####################
####################
5
silence
silence
silence
10 8
9 8
8 9
```

Sample Output

```
E
E
E
E
E
S
```

5 Pixel



Shuffling the pixels in a bitmap image sometimes yields random looking images. However, by repeating the shuffling enough times, one finally recovers the original images. This should be no surprise, since "shuffling" means applying a one-to-one mapping (or permutation) over the cells of the image, which come in finite number.

Your program should read a number n and a series of elementary transformations that define a "shuffling" ϕ of $n \times n$ images. Then, your program should compute the minimal number m ($m > 0$), such that m applications of ϕ always yield the original $n \times n$ image.

For instance, if ϕ is counter-clockwise 90° rotation then $m = 4$.

Input

Input is made of two lines, the first line is number n ($2 \leq n \leq 2^{10}$, n even). The number n is the size of images, one image is represented internally by a $n \times n$ pixel matrix (a_i^j), where i is the row number and j is the column number. The pixel at the upper left corner is at row 0 and column 0.

The second line is a non-empty list of at most 32 space-separated words. Valid words are the keywords **id**, **rot**, **sym**, **bhsym**, **bvsym**, **div**, **mix**, or a keyword followed by "-". Each keyword **key** designates an elementary transform (as defined by Figure 1), and **key-** designates the inverse of transform **key**. For instance, **rot-** is the inverse of counter-clockwise 90° rotation, i.e. clockwise 90° rotation. The list $k_1, k_2, \ldots, k_p$ designates the compound transform $\phi = k_1 \circ k_2 \circ \cdots \circ k_p$. For instance, "**bvsym rot-**" is the transform that first performs clockwise 90° rotation then vertical symmetry on the lower half of the image.

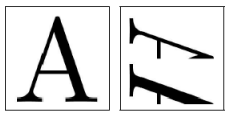


Figure 1: Transformations of image (a_i^j) into image (b_i^j)

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

rot, counter-clockwise 90° rotation

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

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

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

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

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

Output

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

Example

Sample Input	Sample Output
256 rot- div rot div	8

6 Superstitious

Alice lives in Sadtown and would like to get to Awesomeville. There are many cities in the area with major roads connecting some pairs of them, and Alice would like to take some of these roads to get to Awesomeville as quickly as possible. Alice is very superstitious though, and she does not want her total travel time to be divisible by some number K. Tell Alice the minimum amount of time it would take her to get to Awesomeville which isn't divisible by K. If this is not possible, output -1. All roads in the road network are traversable in both directions, but Alice is never allowed to turn around in the middle of a road (she has to wait until arriving at the road's other endpoint).

Input

The first line contains three space-separated integer "N M K" ($1 \leq N \leq 10000$, $1 \leq M \leq 10^6$, $1 \leq K \leq 10^9$). N is the number of cities (including Sadtown and Awesomeville), and M is the number of roads. Sadtown is city number 0, and Awesomeville is city number $N-1$. The next M lines describe the roads. Each road is described by three space-separated integers: u v t. This signifies a bidirectional road between cities u and v which takes t minutes to traverse ($1 \leq t \leq 10^9$). Some cities may have multiple roads between them, but a road will never connect a city to itself.

Output

On a single line output the minimum possible time not divisible by K in which Alice can reach Awesomeville. If this is not possible, output -1. Your answer should be followed by a newline.

Example

Sample Input	Sample Output
4 4 2 0 1 1 0 2 2 1 3 1 2 3 1	3

This page was intentionally left blank.

7 Taxi route

Taxis wait at cab lines in different parts of the city to pick up passengers. At the busiest cab lines, there are always more passengers than cabs, so cabs never need to wait for a passenger. The really obnoxious taxi drivers only take passengers who want to go the same place that the driver wants to go. Of course, the drivers want to make most money over the course of a day, so they only accept passengers who are going to another place with a busy cab line.

Find the greatest profit that an obnoxious taxi driver can make per unit time on any route which begins and ends at the same cab line.

Input

The input begins with an integer N, the number of cab lines in the city. The following N lines, one for each cab line, contain $N-1$ pairs of positive real numbers, giving the amount of time in minutes required and the profit in dollars earned traveling to each *other* cab line in the city.

You may assume that $2 \leq N \leq 200$.

Output

The output is a single number giving the maximum profit in dollars per minute that an obnoxious cab driver can earn. The answer should be given with four digits after the decimal point, rounded in any reasonable manner.

Sample Input	**Sample Output**

`4`
`10 20 10 20 20 100`
`10 20 10 20 20 100`
`10 20 10 20 20 100`
`100 5 100 5 100 5`
	`2.0000`

This page was intentionally left blank.

8 Towns

Along the rocky road to Dublin, there are N towns, indexed with numbers 1 through N. Town 1 is Dublin, its position is 0, and the position of other towns is measured as distance to Dublin in km. Town N is at the other end of the road. Irish engineers don't know the positions of other towns in km. Recently, they found an old archive with distances between each pair of towns. However, there is no information about the pairs themselves, only distances are listed! As their descendant, they ask you to make a program that will recover the positions of N towns.

Input

The first line of the input contains one integer M, which is the number of pairs of N towns ($2 \leq N \leq 2000$). Next M lines contains distances between pairs of towns, one per line. Each distance is an integer greater than 0 and less than or equal to 10^5. Distances are given in non-decreasing order.

Output

In the first line of the output Write one integer N followed by a newline, which is the number of towns. In the next N lines write the positions of N towns in ascending order (starting from 0), one per line. If there are multiple possible solutions, write any of them.

Example

Sample Input	**Sample Output**
6 1 2 3 4 5 6	4 0 2 5 6

This page was intentionally left blank.

9 Fish Catch

Fishermen of New England like to catch so called "SLSS" fish. SLSS stands for "straight line same speed fish", because such fish is very lazy and rarely changes its speed and direction of travel. Fishermen bring a net in a shape of a circle, that has an adjustable radius. They locate the best position for the center of the net, based on their visual observation of high concentration of fish. There are N fish. They also write down the initial position and the direction of travel of each fish. Their plan is to catch K fish in order to make a nice dinner for their families. However, they don not want to catch many more fish, so that there is enough fish left in the Ocean. They need to set a proper radius length of the net, which leads them to a hard computational problem. As a fishermen's friend, please help them to find the smallest radius of the net such that there is a moment when they can dip the net into the Ocean and catch at least K fish.

Input

The first line of the input contains two integers C_x and C_y separated by a space character ($1 \leq C_x, C_y \leq 10^4$), that represent the coordinates of the center of the net. The second line of the input contains two integers N and K separated by a space character ($5 \leq N \leq 1000, 1 \leq K \leq N$). Each of the next N lines contain four integers A_x, A_y, B_x, B_y separated by a space character ($0 \leq A_x, A_y, B_x, B_y \leq 10^4$). A_x and A_y represent the initial coordinates of a fish (at time zero), while B_x and B_y represent the estimated coordinates of a fish after 1 second. Note that fishermen can use the net at any moment starting from time zero.

Output

In the first line of the output write one real number R using exactly 5 decimal digits of precision – which is the desired radius of the net. It should be followed by a newline.

Example

Sample Input	**Sample Output**
6 7 5 4 1 4 1 11 4 12 10 10 4 9 4 7 2 4 10 4 3 1 9 1	5.00000

This page was intentionally left blank.

10 PackIt!

MIT students really like to play with the transformers toys. For that reason, the coaches decided to give a 2D transformer as a prize for the best team on the MIT-ACM competition.

A 2D transformer is made of two dimensional steel parts which are connected by joints. The joints can rotate (around the axis normal to the plane of the transformers) in 90^o increments.

A 2D transformer can be represented by a 2D matrix in the following form:

```
.aa0bb.
```

Where . represents free space, and the letters represent the steel parts, and finally 0 (zero) represents a joint. A joint is connected to all the parts that are either vertically or horizontally (but not diagonally) adjacent to it.

Again, each joint can move in 90 degrees increments. The transformer above can look transform to:

```
.aa0
...b
...b
```

or

```
...b
...b
.aa0
```

However, the following 2D transformer can not transform at all (because of the way the parts are connected)

```
0aa0
b..c
b..c
0dd0
```

The joints are just flexible enough so that you can transform the following 2D transformer:

```
aaaa..
a..0bb
aaaa..
```

into

```
aaaa..
abb0..
aaaa..
```

The coaches want to pack the transformer into a box with the smallest possible area, but are lazy to determine it themselves.

Input

The first line of input contains two integers R and C - the size of the matrix that represents the transformer ($1 \leq R, C \leq 50$). The next R lines contain C characters representing the matrix. Each joint will connect exactly two different parts. The transformer will have at most 10 joints.

Output

Output the area of the smallest box that the transformer can fit in.

Example

Sample Input	Sample Output
5 5 aaaa0 ....b 0e..b d...b 0ccc0	16

Sample Input	Sample Output
5 7 ..aaaa0 ....a.b ee0...b ..d...b ..0ccc0	25