Problem A. Where is the Boundary

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

An island country JAGAN in a certain planet is very long and narrow, and extends east and west. This long country is said to consist of two major cultural areas — the eastern and the western. Regions in the east tend to have the eastern cultural features and regions in the west tend to have the western cultural features, but, of course, the boundary between the two cultural areas is not clear, which has been an issue.

You are given an assignment estimating the boundary using a given data set.

More precise specification of the assignment is as follows:

1. JAGAN is divided into n prefectures forming a line in the east-west direction. Each prefecture is numbered $1, 2, \ldots, n$ from WEST to EAST.
2. Each data set consists of m features, which has 'E' (east) or 'W' (west) for each prefecture. These data indicate that each prefecture has eastern or western features from m different point of views, for example, food, clothing, and so on.
3. In the estimation, you have to choose a cultural boundary achieving the minimal errors. That is, you have to minimize the sum of 'W'-s in the eastern side and 'E'-s in the western side.
4. In the estimation, you can choose a cultural boundary only from the boundaries between two prefectures.

Sometimes all prefectures may be estimated to be the eastern or the western cultural area. In these cases, to simplify, you must virtually consider that the boundary is placed between prefectures number 0 and number 1 or between prefecture number n and number $n + 1$. When you get multiple minimums, you must output the most western (least-numbered) result.

Write a program to solve the assignment.

Input

The first line consists of two integers n ($1 \le n \le 10^4$) and m ($1 \le m \le 100$), which indicate the number of prefectures and the number of features in the assignment. The following m lines are the given data set in the assignment. Each line contains exactly n characters. The j-th character in the i-th line d_{ij} is 'E' (east) or 'W' (west), which indicates j-th prefecture has the eastern or the western feature from the i-th point of view.

Output

Print the estimated result in a line. The output consists of two integers sorted in the ascending order which indicate two prefectures touching the boundary.

Examples

standard input	standard output
2 1 WE	1 2
3 2 WWE WEE	1 2
3 1 WWW	3 4
3 1 WEW	1 2

Note

In second sample, both estimates "1 2" and "2 3" achieve 1 error as the minimum. From the restriction that you must adopt the most western estimate, you must output "1 2".

In third sample, all the prefectures are western. As described in the problem statement, you must virtually consider that the boundary is placed between third and fourth prefectures.

In fourth sample, you cannot assume that 'E's and 'W's are separated.

Problem B. Vector Field

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

In 20015, JAG (Jagan Acceleration Group) has succeeded in inventing a new accelerator named "Force Point" for an experiment of proton collision on the two-dimensional xy-plane. If a proton touches a Force Point, it is accelerated to twice its speed and its movement direction is veered. A proton may be veered by a Force Field in four ways: the positive or negative directions parallel to the x- or the y-axis. The direction in which a proton veers is determined by the type of the Force Point. A Force Point can accelerate a proton only once because the Force Point disappears immediately after the acceleration. Generating many Force Points on the two-dimensional plane, which is called a 2D Force Point Field, allows us to accelerate a proton up to a target speed by sequentially accelerating the proton with the Force Points in the 2D Force Point Filed.

The Force Point generation method is still under experiment and JAG has the following technical limitations:

- JAG cannot generate a Force Point with a specified position and a type.
- JAG cannot generate a Force Point after putting a proton into a 2D Force Point Field.
- JAG cannot move Force Points.
- JAG cannot change a proton's direction except by the effect of Force Points.
- JAG can use only one proton for a 2D Force Point Field.
- JAG can put a proton moving in any direction with its speed 1 at any position in a 2D Force Point Field.

In order to achieve the maximum speed of a proton, the engineers at JAG have to choose the optimal initial position and the optimal initial direction of the proton so that the proton is accelerated by as many Force Points as possible, after carefully observing the generated 2D Force Point Field.

By observing a generated 2D Force Point Field, the number of the generated Force Points is known to be n. The position (x_i, y_i) and the direction veering type d_i of the i-th point are also known. Your task is to write a program to calculate the maximum speed of a proton by acceleration on a given 2D Force Point Field when JAG puts a proton optimally.

Input

The first line contains one integer n ($1 \le n \le 3000$) which is the number of the Force Points on the 2D Force Point Field. Each of the next n lines contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^9$) and one character d_i (d_i is one of '`>`', '`v`', '`<`' or '`^`'). x_i and y_i represent a coordinate of the i-th Force Point, and d_i is the direction veering type of the i-th force point. A force point with a type '`>`' changes proton's direction to the positive direction of the x-axis, '`v`' represents the positive direction of the y-axis, '`<`' represents the negative direction of the x-axis, and '`^`' represents the negative direction of the y-axis. You can assume that any two Force Points are not generated on the same coordinates.

Output

Display a line containing the integer $log_2 v_{max}$, where v_{max} is the proton's possible fastest speed.

Example

standard input	standard output
9 0 0 v 1 0 > 2 0 < 0 1 > 1 1 v 2 1 v 0 2 ^ 1 2 ^ 2 2 <	9
9 0 0 ^ 1 0 ^ 2 0 ^ 0 1 < 1 1 ^ 2 1 > 0 2 v 1 2 v 2 2 v	2

Note

In first sample input looks like the following diagram.

```
v><
>vv
^^<
```

All the Force Points will disappear if you put a proton at (1,1).

Problem D. Identity Function

Input file: *standard input*
Output file: *standard output*
Time limit: 4 seconds
Memory limit: 256 mebibytes

You are given an integer N, which is greater than 1.

Consider the following functions:

- $f(a) = a^N \mod N$
- $F_1(a) = f(a)$
- $F_{k+1}(a) = F_k(f(a))(k = 1, 2, 3, \ldots)$

Note that we use $\mod$ to represent the integer modulo operation. For a non-negative integer x and a positive integer y, $x \mod y$ is the remainder of x divided by y.

Output the minimum positive integer k such that $F_k(a) = a$ for all positive integers a less than N. If no such k exists, output -1.

Input

The input consists of a single line that contains an integer N ($2 \leq N \leq 10^9$), whose meaning is described in the problem statement.

Output

Output the minimum positive integer k such that $F_k(a) = a$ for all positive integers a less than N, or -1 if no such k exists.

Examples

standard input	standard output
3	1
4	-1
15	2

Problem F. Marching Course

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Since members of Kitafuji High School Brass Band Club succeeded in convincing their stern coach of their playing skills, they will be able to participate in Moon Light Festival as a marching band. This festival is a prelude in terms of appealing their presence for the coming domestic contest. Hence, they want to attract a festival audience by their performance.

Although this festival restricts performance time up to P minutes, each team can freely determine their performance course from a provided area. The provided area consists of N checkpoints, numbered 1 through N, and M bidirectional roads connecting two checkpoints. Kitafuji Brass Band already has the information about each road: its length and the expected number of people on its roadside. Each team must start at the checkpoint 1, and return back to the checkpoint 1 in P minutes. In order to show the performance ability of Kitafuji Brass Band to a festival audience, their stern coach would like to determine their performance course so that many people listen their march as long as possible.

The coach uses *impression degree* to determine their best course. If they play m minutes on the road with length d and the expected number v of people, then the impression degree will be $m \times v/d$. The impression degree of a course is the sum of impression degree of their performance on the course. Marching bands must move at a constant speed during marching: 1 unit length per 1 minute. On the other hand, they can turn in the opposite direction at any time, any place including a point on a road. The impression degree is accumulated even if they pass the same interval two or more times.

Your task is to write a program to determine a course with the maximum impression degree in order to show the performance ability of Kitafuji Brass Band to an audience as much as possible.

Input

The first line contains three integers N, M, and P: the number of checkpoints N ($2 \leq N \leq 200$), the number of roads M ($N-1 \leq M \leq N(N-1)/2$), and the performance time P ($1 \leq P \leq 1000$). The following M lines represent the information about roads. The i-th line of them contains four integers s_i, t_i, d_i, and v_i: the i-th road bidirectionally connects between checkpoints s_i and t_i ($1 \leq s_i, t_i \leq N$, $s_i \neq t_i$) with length d_i ($1 \leq d_i \leq 1000$) and the expected number v_i ($1 \leq v_i \leq 1000$) of people.

You can assume that any two checkpoints are directly or indirectly connected with one or more roads. You can also assume that there are no pair of roads having the same pair of endpoints.

Output

Output the maximum impression degree of a course for a P-minute performance. The absolute error should be less than 10^{-4}.

Examples

standard input	standard output
3 3 4 1 2 1 1 2 3 2 4 3 1 1 1	6
4 3 9 1 2 2 1 1 3 2 2 1 4 2 3	13.5
4 3 5 1 2 10 1 2 3 2 100 1 4 3 10	16.6666666667
3 3 10 1 2 3 1 1 3 4 5 2 3 2 10	22

Problem G. Surface Area of Cubes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Taro likes a signle player game “Surface Area of Cubes”.

In this game, he initially has an $A \times B \times C$ rectangular solid formed by $A \times B \times C$ unit cubes (which are all 1 by 1 by 1 cubes). The center of each unit cube is placed at 3-dimensional coordinates (x, y, z) where x, y, z are all integers ($0 \leq x \leq A-1$, $0 \leq y \leq B-1$, $0 \leq z \leq C-1$). Then, N distinct unit cubes are removed from the rectangular solid by the game master. After the N cubes are removed, he must precisely tell the total surface area of this object in order to win the game.

Note that the removing operation does not change the positions of the cubes which are not removed. Also, not only the cubes on the surface of the rectangular solid but also the cubes at the inside can be removed. Moreover, the object can be divided into multiple parts by the removal of the cubes. Notice that a player of this game also has to count up the surface areas which are not accessible from the outside.

Taro knows how many and which cubes were removed but this game is very difficult for him, so he wants to win this game by cheating! You are Taro’s friend, and your job is to make a program to calculate the total surface area of the object on behalf of Taro when you are given the size of the rectangular solid and the coordinates of the removed cubes.

Input

The first line of a test case contains four integers A, B, C, and N ($1 \leq A, B, C \leq 10^8$, $0 \leq N \leq \min\{1,000, A \cdot B \cdot C - 1\}$).

Each of the next N lines contains non-negative integers x,y, and z which represent the coordinate of a removed cube. You may assume that these coordinates are distinct.

Output

Output the total surface area of the object from which the N cubes were removed.

Examples

standard input	standard output
2 2 2 1 0 0 0	24
1 1 5 2 0 0 1 0 0 3	18
3 3 3 1 1 1 1	60

Problem H. Laser Cutter

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

Ciel is going to do woodworking. Ciel wants to make a cut in a wooden board using a laser cutter.

To make it simple, we assume that the board is a two-dimensional plane. There are several segments on the board along which Ciel wants to cut the board. Each segment has a direction and Ciel must cut those segments along their directions. Those segments are connected when you ignore the directions, that is, any two points on the segments are directly or indirectly connected by the segments.

While the laser cutter is powered on, it emits a laser which hits the board at a point and cuts the board along its trace. The laser initially points to (x, y). Ciel can conduct the following two operations:

- Move the laser cutter with its power on and cut (a part of) a segment along its direction, or
- Move the laser cutter to any position with its power off. Ciel should not necessarily cut the whole segment at a time; she can start or stop cutting a segment at any point on the segments.

Ciel likes to be efficient, so she wants to know the shortest route such that the laser cutter cuts the whole parts of all the segments and then move back to the initial point. Your task is to write a program that calculates the minimum total moving distance of the laser cutter.

Input

The first line of the input contains an integer n ($1 \le n \le 300$), the number of segments. The next line contains two integers x and y ($-1000 \le x, y \le 1000$), which is the initial position (x, y) of the laser. The i-th of the following n lines contains four integers sx_i, sy_i, tx_i and ty_i ($-1000 \le sx_i, sy_i, tx_i, ty_i \le 1000$), which indicate that they are the end points of the i-th segment, and that the laser cutter can cut the board in the direction from (sx_i, sy_i) to (tx_i, ty_i). The input satisfies the following conditions:

For all i ($1 \le i \le n$), $(sx_i, sy_i) \ne (t_x i, t_y i)$. The initial point (x, y) lies on at least one of the given segments. For all distinct i, j ($1 \le i, j \le n$), the i-th segment and the j-th segment share at most one point.

Output

Output a line containing the minimum total moving distance the laser cutter needs to travel to cut all the segments and move back to the initial point. The absolute error or the relative error should be less than 10^{-6}.

Examples

standard input	standard output
3 0 1 0 0 0 1 0 1 0 2 0 2 0 3	6.0
2 0 1 0 0 0 2 -1 1 1 1	6.8284271247461900
5 0 0 0 0 1 0 1 1 -1 1 -1 1 -1 -1 -1 -1 1 -1 1 -1 1 1	10.00

Problem J. Black Company

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

JAG Company is a sweatshop (sweatshop is called "burakku kigyo" in Japanese), and you are the CEO for the company.

You are planning to determine N employees' salary as low as possible (employees are numbered from 1 to N). Each employee's salary amount must be a positive integer greater than zero. At that time, you should pay attention to the employees' contribution degree. If the employee i's contribution degree c_i is greater than the employee j's contribution degree c_j, the employee i must get higher salary than the employee j's salary. If the condition is not satisfied, employees may complain about their salary amount.

However, it is not necessarily satisfied in all pairs of the employees, because each employee can only know his/her close employees' contribution degree and salary amount. Therefore, as long as the following two conditions are satisfied, employees do not complain to you about their salary amount.

- If the employees i and j are close to each other, $c_i < c_j \Leftrightarrow p_i < p_j$ must be satisfied, where p_i is the employee i's salary amount.
- If the employee i is close to the employees j and k, $c_j < c_k \Leftrightarrow p_j < p_k$ must be satisfied.

Write a program that computes the minimum sum of all employees' salary amount such that no employee complains about their salary.

Input

The first line contains an integer N ($1 \leq N \leq 10^5$), which indicates the number of employees. The second line contains N integers c_i ($1 \leq c_i \leq 10^5$) representing the contribution degree of employee i.

The third line contains an integer M ($0 \leq M \leq 2 \cdot 10^5$), which specifies the number of relationships. Each of the following M lines contains two integers a_i and b_i ($a_i \neq b_i$, $1 \leq a_i, b_i \leq N$). It represents that the employees a_i and b_i are close to each other. There is no more than one relationship between each pair of the employees.

Output

Print the minimum sum of all employees' salary amounts in a line.

Examples

standard input	standard output
3 1 3 3 2 1 2 1 3	5
3 1 2 3 2 1 2 1 3	6
4 1 1 2 2 2 1 2 3 4	4
5 1 2 5 5 1 6 1 2 4 1 2 3 5 2 4 3 4 5	10
6 4 3 2 1 5 3 7 4 2 1 5 2 6 6 5 4 1 1 6 6 3	13

Problem K. Emoticon Counter

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The three consecutive characters "`:-)`" indicate a happy smile and the three consecutive characters "`:-(`" indicate a sad smile.

The overall mood of a text message is defined by next way: if message contains more happy emoticons, than sad ones, mood is "`happy`", otherwise if it contains more sad emoticons, than happy ones, mood is "`sad`", otherwise the mood is "`neutral`".

Write a program to determine the overall mood of a message.

Input

There will be one line of input that contains between 1 and 255 characters.

Output

If overall mood is "happy", print "`happy`". If overall mood is "sad", print "`sad`". If overall mood is neutral, print "`neutral`".

Examples

standard input	standard output
Solved the problem :-) got TL :-(then Accepted :-)	happy
Happy? :)	neutral
Emo:-(ticons are go:-(:-(ing out of co:-)ntrol	sad

Problem L. Rogue Language

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In the so-called Rogue Language English words are encrypted by the next way:

- Vowels (‘a’, ‘e’,‘i’,‘o’ and ‘u’ remains unchanged.
- Consonants are replaced by three letters: the consonant itself, the vowel closest to the consonant in the alphabet (e.g., if the consonant is ‘b’, then the closest vowel is ‘a’), with the rule that if the consonant falls exactly between two vowels, then the vowel closer to the start of the alphabet will be chosen (e.g., if the consonant is ‘c’, then the closest vowel is ‘a’) and the next consonant in the alphabet following the original consonant; if the original consonant is ‘z’, then another ‘z’ is used.

Write a program that encrypts a given word.

Input

The input consists of one non-empty word composed of no more than 30 lower-case English letters.

Output

Print the encrypted word.

Examples

standard input	standard output
opencup	opoqenopcadupoq
xyz	xuyyuzzuz

Problem M. RPS

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Alice and Bob are playing rock-paper-scissors game. The game rules are simple: rock crushes scissors; paper covers rock; and scissors cut paper. If both players throw the same shape, the game is tied. Your task is to determine the number of games won by Alice and Bob, respectively.

Input

The first line contains one integer N ($1 \le N \le 100$) that represents the number of games.

The second line is Alice's turns sequence; it contains N words. The i–th word in the sequence represents the move of Alice in the i–th game. There are only three words possible: "`rock`", "`paper`" and "`scissors`".

The third line is Bob's turns sequence in the same form.

Output

Print two integers: number of games won by Alice and the number of games won by Bob.

Examples

standard input	standard output
4 paper scissors rock paper rock rock scissors paper	2 1