Problem A. A Simple Geometry Problem

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Rotation plays an important in computational geometry. In this problem, we are interested in the following 2-dimensional rotation problem:

Given a 2-d point (x, y), what's the position after rotating counterclockwise by an angle θ?

It is perfect if you already know how to solve this problem from other classes like linear algebra. Otherwise, we will derive the solution as follows.

My favorite derivation is based on the complex numbers. A complex number is a number that can be expressed in the form $a + bi$, where a and b are real numbers, and i is a solution of the equation $z^2 = -1$, which is called an imaginary number because there is no real number that satisfies this equation. The rotation is actually equivalent to $(x+yi)\cdot(\cos\theta+\sin\theta i) = (x\cos\theta-y\sin\theta)+(x\sin\theta+y\cos\theta)i$. Therefore, the new x coordinate is $x\cos\theta - y\sin\theta$ and the new y coordinate is $x\sin\theta + y\cos\theta$. Now, you should be able to solve this problem using the built-in math functions :). Enjoy!

Input

The first line of input contains a positive integer T ($1 \le T \le 100$), which represents the number of test cases in total. In the follow T lines, every line has three floating numbers, x, y, and θ ($0 \le |x|, |y| \le 100, 0 \le \theta \le 360$), as described before. θ is in degrees. There are exactly 2 digits after the decimal point for all floating numbers.

Output

For each test case, print a line of the rotated position. All floating numbers are **rounded up to 2 digits after decimal points**.

Examples

stdin	stdout
3 1 0 0.00 2 0 90.00 2 1 30.00	1.00 0.00 0.00 2.00 1.23 1.87

Problem B. Barfoosia Tax Return

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

It's tax return season in the United States of Barfoosia! Because their IRS employees are lazy, U.S.B's tax system is much simpler: when payroll is delivered, a 50% tax is withheld, regardless of your pre-tax income or any other personal factors. In the tax return process, a final tax amount will be calculated and a balance will be refunded or collected. The final tax amount will be calculated by a staggered rate based on one's pre-tax annual income:

The part of income falling within \$0-\$10,000 will be tax-free
The part of income falling within \$10,000-\$20,000 will be taxed 10%
The part of income falling within \$20,000-\$40,000 will be taxed 20%
The part of income falling within \$40,000-\$80,000 will be taxed 40%
The part of income above \$80,000 will be taxed 80%

For example, if Alice made \$5,000 last year, all her income will fall into the tax-free column, and she should have been taxed \$0. Since \$2,500 was withheld from her payroll, she should be refunded \$2,500. If Bob made \$35,000 last year, his first \$10,000 income is tax-free, his next \$10,000 income is taxed by 10%, and his remaining \$15,000 income is taxed by 20%. He should have been taxed

$$\$10,000 * 0\% + (\$20,000 - \$10,000) * 10\% + (\$35,000 - \$20,000) * 20\% = \$4,000$$

Since \$17,500 was withheld from his payroll, he should be refunded \$13,500.

Given the pre-tax annual income (always a multiple of 100) of U.S.B residents, compute amount refunded to or owed by each person.

Input

The first line of input contains a positive integer N ($1 \leq N \leq 100$), which represents the number of residents in U.S.B.
In the following N lines, the ith line has one integer X_i ($0 \leq X_i \leq 1,000,000$, X_i is guaranteed to be a multiple of 100), the pre-tax annual income of the ith resident.

Output

For each resident, print a line containing tax refunded or owed. If the person should be refunded tax, print "Refund "followed by the amount to be refunded; if the person owes tax, print "Owe "followed by the amount owed; if neither, print "Clear". All printed amounts should be integers.

Examples

stdin	stdout
4 5000 35000 0 150000	Refund 2500 Refund 13500 Clear Owe 2000

Problem C. Common Mistake

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

In the era of dreadroughts and battleships, people love to stack as many guns as possible onto ships to increase their fire power.

However, this is a common mistake, because adding too many guns may decrease maneuverability and mobility, and ends up decreasing the overall battle efficiency of the ships.

Given an array of N guns available, your are in charge of making the optimal selection so the battle efficiency of a ship is maximized.

You are given the following simplified model:

Each gun will have a weight w_i and power p_i.

The ship has K load ranges. While the total weight of guns selected is in load range $[l_i, r_i]$, the battle efficiency of the ship is calculated as $m_i \sum_j p_j$ where m_i is the modifier associated with that load range, and j denotes the indices of all selected guns.

Yamato with her 9x46cm guns

Note m_i will not necessarily decrease as l_i goes up, because having more guns will also bring crew members pride and increase their morale, and so bring up the ship battle efficiency.

Input

The first line of input contains one integer $N(1 \leq 500)$, the number of guns available.

In the next N lines, each line will contain two nonnegative integers, $w_i, p_i(1 \leq w_i, p_i \leq 1000)$.

The next line contains one integer $K(2 \leq K \leq 1000)$, the number of load ranges.

In the next K lines, each line contains two numbers l_i, m_i, the first integer is the starting weight $l_i(0 \leq l_i \leq 5000)$ of this load range. The second number is the power modifier $m_i(0 \leq m_i \leq 10)$.

It is guaranteed the load ranges will be given in the increasing order of l_i, so l_0 will always be 0.

It is also guaranteed the last load range will always have power modifier 0, which can be considered as right past the maximum load capacity of this ship.

The total weight of all guns will not exceed 5000.

Output

A single line with an number giving the maximum battle efficiency of the ship. The floating number is **rounded up to 3 digits after decimal points**.

Examples

stdin	stdout
3 10 30 20 40 20 40 4 0 1.0 20 2.5 50 1.8 60 0	200.000
2 100 100 200 200 2 0 2.5 10 0	0

Problem D. Dull Game

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Summer break just started and you just arrived home yesterday. Even better, you just got yourself the newly released game of Ubihard. The game tells the story about the creation of the world and you will become the Creator. As we know, at the beginning of time, before creating any creatures, the first thing the Creator should do is to create raindrops. On the other hand, obviously, there is a Destroyer that will prevent the world from forming. (I know it's a lame game, but what do you expect from Ubihard?). The game takes place on a $2D$ plane and is limited by the rectangle with lower-left corner $(0, 0)$ and top-right corner (m, n). The x-axis represents the ground. The rules are:

- The Creator can select a point (x, y) inside the give frame with integer coordinates a $(x, y \in \mathbb{Z}, 0 \leq x \leq m, y \leq n)$ and create a raindrop at (x, y). Then, the Creator will receive y points.
- The Destroyer can create horizontal bars inside the given frame to prevent the raindrops from touching the ground. A bar can be represented as a segment with 2 endpoints (x_1, y) and (x_2, y) with integer coordinates $(0 \leq x_1 < x_2 \leq n, 0 \leq y \leq m, x_1, x_2, y \in \mathbb{Z})$. After created, the bar will stay there for the rest of the game. A raindrop will not reach the ground if it is created above a bar. That is, if there is a bar with end points (x_1, y), (x_2, y) and a raindrop created at (x', y'), the drop will not reach the ground if and only if $x_1 \leq x' \leq x_2$ and $y \leq y'$. Because there is a bug in the game, the newly created bars can overlap with previously created bars (Why? Cause Ubihard created it!).
- At each turn, the Destroyer will first add a bar. Then, the Creator will create 1 raindrop.

Given the moves of the Destroyer, find the maximum amount of points you can get at each turn.

Input

The first line contains two integers m, n $(1 \leq m, n \leq 10^5)$ represents the top-right corner (m, n) of the frame. The second line contains q, the number of turns you and your brother played in total. Each of the next q lines contain 3 integers x_1, x_2, y $(0 \leq x_1 < x_2 \leq m, 1 \leq y \leq n)$ that represent a move of your brother. That is, creating a bar with the 2 end points (x_1, y) and (x_2, y).

Output

For each turn, print a single line with the maximum a mount of point you can get in that turn.

Examples

stdin	stdout
5 5 3 1 2 4 3 5 3 1 4 2	5 3 2

Problem E. Enclose These Cows

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Farmer John has a herd of cows, and he plans to build a fence to enclose them so that no cow can escape (a cow on the boarder of the fence is not considered enclosed in the fence). There are already some poles on the field, and Farmer John can only build fence segments between these poles. Help Farmer John minimize the cost by computing the minimal length of a valid fence.

Input

The first line of input contains two integers N, M ($1 \leq N, M \leq 100$). N is the number of poles and M is the number of cows.
In the following N lines, each line contains two integers x, y ($-10^6 \leq x, y \leq 10^6$), indicating that the coordinate of a pole is (x, y) in 2-D Cartesian coordinate system.
In the following M lines, each line contains two integers x, y ($-10^6 \leq x, y \leq 10^6$), indicating that the coordinate of a cow is (x, y) in 2-D Cartesian coordinate system.
All positions are guaranteed to be distinct. (i.e. no two poles, or two cows, or one pole and one cow, can take the same position)

Output

Print a line containing one real number, the minimal length of fence, **rounded to 2 digits after decimal point**. If it's impossible to build such a fence, output -1.

Examples

stdin	stdout
6 4 2 0 1 2 -1 2 -2 1 -1 0 0 -2 0 0 1 1 -1 1 0 -1	11.83
3 1 0 0 1 1 -1 1 0 1	-1

Problem F. Greedy String Match

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Given a long string and a set of keywords, you are asked to find non-overlapped matches of these keywords in the given string. Although there are several dynamic programming algorithms can maximize the total number of matched characters, you are NOT going to do that in this problem, considering the efficiency.

Alternatively, you are going to implement a greedy algorithm: from the left to the right, try to match the longest keyword starting from every non-matched character. In the end, print the total number of matched characters.

Input

The first line of input contains the long string S ($1 \leq |S| \leq 10^5$). The second line contains an integer N, i.e., the number of keywords ($1 \leq N \leq 10^4$). In next N lines, there is a single keyword per line. The lengths of these keywords are no longer than 100.

Output

For each test case, print a line of the total number of matched characters.

Examples

stdin	stdout
This ICPC contest is sooo cooooool! 3 ICPC so cool	4
thisisanoverlapcase 4 this isis is san	6

Problem G. Hard Combinatorics

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

In this problem, you are asked to calculate the following formula efficiently.

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\binom{A_i + B_j + C_i + D_j}{A_i + B_j} \tag{1}$$

where N, A_i, B_i, C_i, and D_i are all given positive integers.

Input

The first line of input contains a positive integer N ($1 \le N \le 10^5$). The following four lines describe A_i, B_i, C_i, and D_i, respectively ($1 \le A_i, B_i, C_i, D_i \le 10^3$).

Output

For each test case, print a line of the desired answer.

Examples

stdin	stdout
2 1 2 3 4 5 6 7 8	

Problem H. Just A Hike!

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

The summer just started and you just spent all night last night playing the famous Creator-Destructor game of Ubihard. Thus, you decide to do something more helpful this morning - cleaning the attic. Cleaning the attic, although seems to be a boring job to do, is in fact your favourite chore at home. While doing the task, you always have a chance to go through the old photos of your family and spend time recalling all the good memories you had during your childhood. Suddenly, you come accross the old photo of you and your family in your first hike. At this very moment, you decide that your second mission of the summer (the first one is to finish the very interesting game mentioned above) is to get your family to go on a hike together one more time.

You start to plan for the hike using a that has the estimated height of the countryside area of your hometown. It can be represented as a rectangle grid with m rows and n columns. The cell at the i_{th} row and j_{th} column is denoted as cell (i, j) and has the height $h[i, j]$. You call a cell (i, j) a peak if its height is greater than or equal all the cell adjacent to it. Two cells are considered adjacent if the share a common edge. Thus, a cell has at most 4 adjacent cells. You believe that any peak in the map represents a real mountain and can be used for hiking. You want to find any peak here as the destination for your hike. However, since the map is rather large, it is simply impossible to find a peak manually. As a Computer Science student, you know this is your moment to shine!

Input

The first line contains two integers m, n ($1 \leq m, n \leq 10^5$) which represent the size of the map. The second line contains n integers $h[1, 1], h[1, 2], \cdots, h[1, n]$ represent the height of all the cells in the first row of the map. Since the map is rather large, this is the only thing you get. The rest of the map can be calculated with the formula

$$h[i, j] = h[1, j]^j \bmod 1000000007 \tag{2}$$

where mod denotes the remainder of the division for $10^9 + 7$.

Output

Print 2 numbers i, j denotes the coordinate of any peak (i, j).

Examples

stdin	stdout
2 3 2 3 1	2 2

Explanation: The full map is

$$\begin{matrix} 2 & 3 & 1 \\ 4 & 9 & 1 \end{matrix} \tag{3}$$

Then, clearly the peak is cell $(2, 2)$ with height 9.

Problem I. Kindle the Bonfire

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second

Being the last guardian of the lost land of Dregoheap, you are in charge of protecting the inhabitants in this area.

The land of Gregoheap consists of N cities and M roads, where each road is bidirectional and connects exactly 2 cities. During the day, you will patrol around Dregoheap. When you are at some city i, you may receive a distress call from some other city j about an emergency event. Of course, as the sole guardian, you need to go to city j as soon as possible and save the citizens there.

Luckily, at Gregoheap, there are K cities which contain an unkindled bonfire. Once a bonfire is kindled, you can teleport from that bonfire to any other *kindled* bonfire. However, due to limited resources, you are only able to kindle at most L bonfires.

After kindling bonfires in the optimal way, you would like to know in the worst case, what is the longest distance you need to travel in order to respond to one emergency event.

Note your patrol route is random, and the emergency events also happen randomly. You can assume traveling within a city is cost-free. Also teleporting between bonfires is also costfree. You will also only receive distress call when you are in a city, and no two emergencies events will happen at the same time.

Input

The first line of input contains two integers $N, M (1 \leq N \leq 20, 0 \leq M \leq 500)$.

In the next M lines, each line will contain three integers, u_i, v_i, w_i, $1 \leq u_i, v_i \leq N$, $1 \leq w_i \leq 10^6$, which means this road connects cities u_i, v_i and has length w_i.

The next line contains two integers $K, L (0 \leq L \leq K \leq 10)$.

The final line of input contains L integers, denoting the indices of cities which contain an unkindled bonfire.

Output

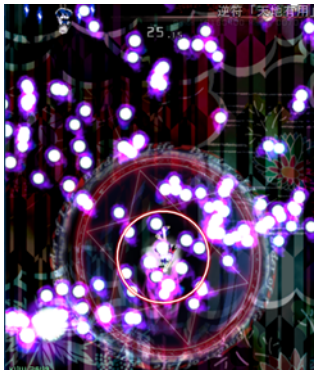
A single line with an integer giving the longest distance you need to travel in case of emergency.

Examples

stdin	stdout
3 3 1 2 1024 1 3 1024 2 3 1024 3 3 1 3 2	0
3 2 1 2 5 1 3 1024 3 2 1 3 2	5

Problem J. Land of Fantasy

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second



Our heroine, once again, is on her way to defeat the evil mastermind threating our beloved land of fantasy.

Before reaching her final target, however, our heroine is blocked by one of the minions of the great evil. Having no interest in dealing with these valueless targets, our heroine decides to use her time manipulation super power to stop the time, and quickly move to the other side of the rectangular battlefield.

Unfortunately, the evil minion has already released a large amount of circular bullets. Even though the heroine can stop the time, she will still die if she touches any of the bullets.

Starting from the upper boundary of the battlefield, our heroine wants to know whether she can safely move to the lower boundary of the battlefield without touching any of the bullets released by the evil minion.

Input

The first line of input contains two numbers $H, W (1 \leq H, W \leq 10^6)$, denoting the height and width of the rectangular battlefield.

The second line of input contains one integer $N (0 \leq N \leq 2000)$, which is the number of circular bullets on the battlefield.

The following N lines each contain three numbers x_i, y_i, r_i, which are the x coordinate, y coordinate, and the radius of the ith bullet.

The lower-left corner has coordinate $(0, 0)$ and the upper-right corner has coordinate (W, H).

The heroine can start from any point on the upper boundary, and can end up on any point on the lower boundary. It is guaranteed no bullet will touch upper and lower boundaries. It is also guaranteed bullets will never be tangent with each other or any of the boundaries. You can assume our heorine is so small compared to the bullets, and can be treated as a point.

Output

A single line either be "YES YES YES" if she can safely move to the other side of the battlefield, or "NO NO NO" otherwise.

Examples

stdin	stdout
6.0 4.0 1 2.0 3.0 1.5	YES YES YES
10.0 6.0 3 2.0 4.0 3.0 3.0 4.0 3.0 4.0 4.0 3.0	NO NO NO