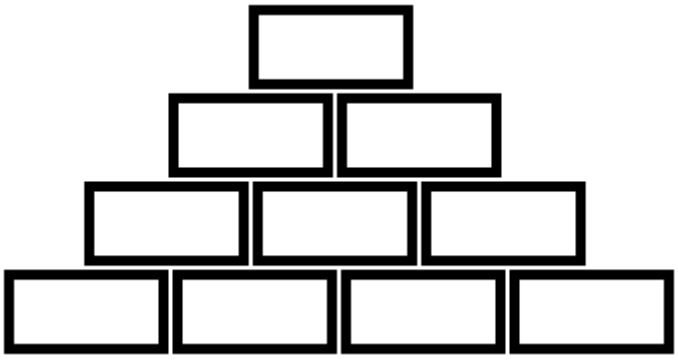
Problem A. Bricks

Input file:	`bricks.in`
Output file:	`bricks.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The annual 32nd of March sale of bricks is in full swing. Hundreds of people are standing in queue waiting for brick stores all over the world to open. As always, the longest queue has formed in front of the BrickStore supermarket. This store has become popular thanks to its product — unique bricks of ideal shape of a rectangular parallelepiped with center of gravity coinciding precisely with the brick's geometric center.

Traditionally, bricks for the sale are arranged in the shape of a large symmetric wall, the lower level of which consists of n bricks lying on the floor. The bricks are arranged as shown in the illustration below:



Each brick is given by two coordinates — the number of the row (rows are numbered downward) and the position of the brick in this row from left to right. Each of k customers in the queue knows for sure which specific brick he/she wants and is ready to buy nothing but that. At the same time conflicts of interest in the queue are excluded — the customers have enough time to talk to each other, which means there is impossible to find two customers who want the same brick. When being purchased, a brick is taken out of the wall so carefully that positions of other bricks are not changed at all.

The supermarket's health and safety advisor found out that if at a certain moment the gravity center of one of the bricks was not supported on the boundary of any other brick or on the floor, the former one would fall down. Such situations must not be allowed, that is why a responsible employee asked all people in the whole queue and each customer told him two numbers (x, y) — the coordinates of the brick he/she is going to buy.

All that remains is to find out the number of the first customer standing in the queue whose purchase will cause a fall of at least one brick.

Input

In the first line there are two integers $1 \le n \le 10^9$, $1 \le k \le 2 \cdot 10^5$. Each of the following k lines contains a pair of integer numbers (x_i, y_i) — the coordinates of the brick, which the i-th customer is about to buy.

Output

The only number — the position of the first customer standing in the queue whose purchase will cause a fall of at least one brick. If the sale takes place without falls, output `-1`.

Examples

bricks.in	bricks.out
10 9 1 1 2 1 2 2 4 1 4 3 5 3 5 4 4 2 5 5	8

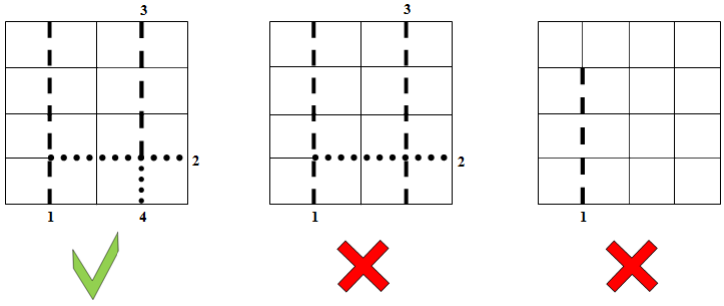
Problem B. Cuts

Input file:	cuts.in
Output file:	cuts.out
Time limit:	3 seconds
Memory limit:	256 megabytes

Train passengers are offered many ways of killing time. Outdated crosswords are replaced by new sophisticated puzzles. One of such puzzles has been created specially for passengers of the Anchorage-Kazan train.

The puzzle is a checkered sheet of paper of size $n \times m$. One number from 1 to $n \cdot m$ is written in each square. Each number is used exactly once.

The task is to cut this sheet of paper into 1×1 squares. Each cut must begin from one boundary of the sheet and finish on the other boundary, and be parallel to one of the coordinate axes. See the examples of cuts below.



Let's define the square's time of release as the number of cuts made before the selected square is released from its neighbors and all squares with a less number are released too.

We are interested in such ways of cutting when a square containing 1 is released by minimal number of cuts. Out of all such ways we are interested in the ways of the earliest release of a square containing 2, and so on. The square containing $n \cdot m$ has the lowest priority.

The task is to find $n \cdot m$ numbers — the number of cuts after which a square containing 1 is released, a square containing 2 is released, ..., a square containing $n \cdot m$ is released when the optimal order of making cuts is applied.

Input

The first line contains two integers n and m ($1 \le n, m \le 30$). The following n lines contain m numbers each describing the puzzle.

Output

Output $n \cdot m$ unknown numbers.

Examples

cuts.in	cuts.out
2 3 1 6 4 3 5 2	2 3 4 5 5 5

Problem C. Icicles (Div-1 only!)

Input file:	`ice.in`
Output file:	`ice.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

There are n square icicles with side of 1 hanging from the roof located very high. Centers of the icicles are located on a line, which is parallel to the X-axis and the i-th icicle is located on the interval $[i-1, i]$ of the X-axis.

There are m balloons in the air. Centers of the balloons lie in one plane with all the icicles, that's why each balloon has three parameters: position of the center on the X-axis and Y-axis, and a radius. All the balloons are moving up along the Y-axis at equal speed. They are passing one unit of length per one unit of time. Any pair of balloons has no more than one common point, and the balloons cannot be located inside each other.

When at least one of the balloons touches an icicle, the Doomsday will occur. Let's consider this point of time to be a reference point.

Girl Valya bought a laser and decided to melt some icicles to postpone the Doomsday, because melted icicles are not a threat for the balloons.

Valya does not know how many shots the laser energy can provide, but it is known for certain that there will be not enough shots to save all the balloons. Therefore, we can assume that all nonnegative numbers of shots, which do not prevent the Doomsday, are equally probable. Valya is ready to do by optimal way, that is she to choose such sequence of icicles destruction which postpones Doomsday for maximal possible time.

The task is to estimate the average time of the Doomsday delay.

Valya is a markswoman and never misses a target, and the laser always shoots. Each shot of laser melts exactly one icicle. It is guaranteed that Valya has enough time to fire all the shots before the balloons approach the icicles at a critical distance.

Input

The first line contains of the number of icicles n and the number of balloons m, $1 \le n \le 10^7$, $1 \le m \le 40\,000$.

The following m lines contain triplets of integers x, y, r, where x, y are the coordinates of the balloon's center, and r is its radius, $-10^9 \le x, y \le 10^9$, $1 \le r \le 10^9$.

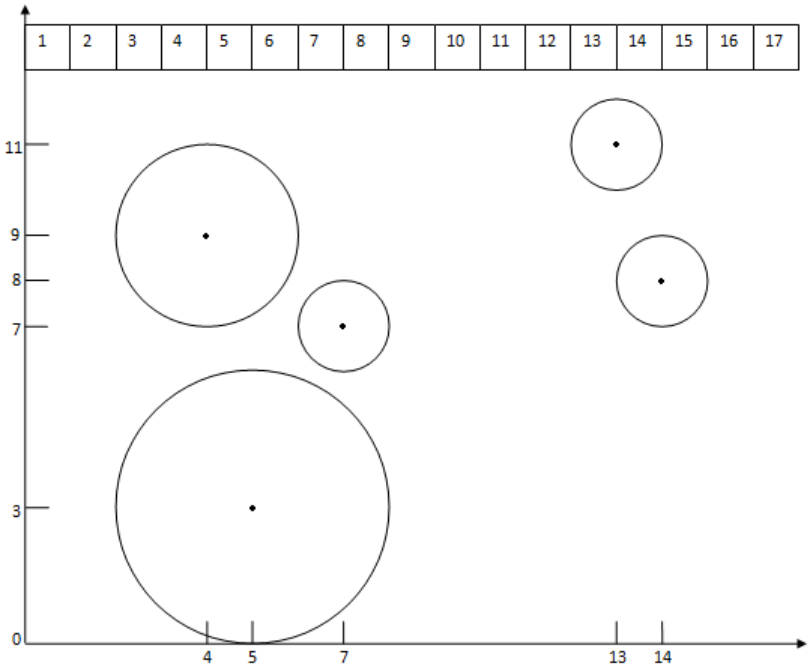
Output

Output the only number — the expected time for which Valya can postpone the Doomsday. Absolute or relative error must not exceed 10^{-6}.

Examples

ice.in	ice.out
17 5 5 3 3 4 9 2 7 7 1 13 11 1 14 8 1	1.8239927417758746723

Note



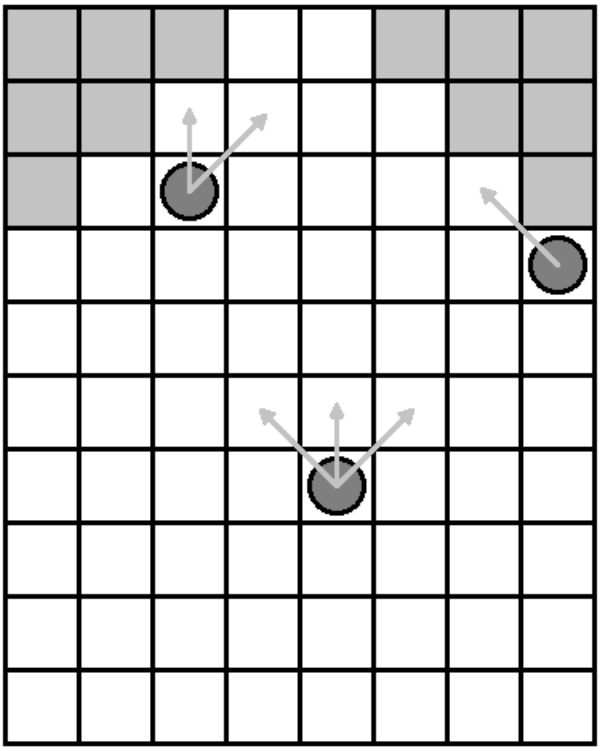
Problem D. The Millenium

Input file:	`millenium.in`
Output file:	`millenium.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Traffic jams occur due to different reasons. They often occur at the entrance point to the bridge due to the simple fact that there are fewer lanes for transport vehicles on the bridge than on the passage of approaching the bridge. To illustrate this effect, a bridge called "Millenium", which is translated from Old-Kazanian as "an endless traffic jam", was built in Kazan.

There are N cars in Kazan and all of them are in the traffic jam.

The road section at the entrance point of the bridge can be represented as a matrix of the height of H lines and the width of $2 \cdot A + B$. There is an example of such road with $H = 10$, $A = 3$, $B = 2$ in the illustration.



Each of the cars in this matrix initially occupies one of the squares. The shaded squares are occupied by bump stops before the section where the road narrows. Thus, the width of the road in the first row equals to B columns, in the second row — $B+2$, etc. Beginning from the row $A+1$, the width of the road equals to $2 \cdot A + B$.

The cars move from the bottom upwards and must enter the bridge through a narrow part of the road, the width of which is B columns. In one unit of time each car can advance one square forward in one of the three directions: up to the left, up, or up to the right, under the condition that the square is vacant. If necessary, a car can stop moving for one or several units of time. The cars move rhythmically and synchronously. It means that if any car's spot becomes vacant, then one of the following cars can take the vacant spot at the same unit of time.

Drivers require information about how fast all cars can leave the section of the narrowing road and enter the bridge, if all drivers act in optimal concert with each other.

Input

The first line contains three integers $1 \le N \le 10^5$, $1 \le A \le 10^9$ и $1 \le B \le 10^9$.

Each of the following N lines contain two integers x and y — number of row and number of column of the i-th car, $1 \le x \le 2 \cdot A + B$ and $1 \le y \le 10^9$. It is guaranteed that there is no car located in area occupied by bump stops.

Output

Print the time all the cars need to enter the bridge.

Examples

millenium.in	millenium.out
5 2 2 1 3 1 4 2 2 2 3 2 4	3

Problem E. Awticulation disowdews

Input file:	changes.in
Output file:	changes.out
Time limit:	2 seconds
Memory limit:	256 megabytes

The world's only group of speech therapist-ethnographers has found a lost island. To the great joy of the discoverers, the island was populated with people, each of whom had exactly one articulation disorder. The scholars found out m types of different articulation disorders which occur on the island. Moreover, an unusual feature of the island was discovered: when $(m-1)$ islanders with different articulation disorders meet, they all simultaneously get rid of their articulation disorder, but get a new disorder which is not represented at the meeting. For example, if $m = 3$, islanders who lisp and burr meet and start stammering. In all other cases the islanders do not change their defect.

At present, specialists are studying the exact population of the islanders, trying to tackle a vitally important question: is it possible that at a certain moment of time there will be exactly l_1 people with the first articulation disorder, l_2 — with the second one, etc., l_m people who have the m-th articulation disorder on the island; and, if it is possible, then under what conditions?

Input

The first line contains one integer m ($3 \le m \le 10\,000$) — the number of articulation disorders. The second line contains m integers $k_1, k_2, ..., k_m$, where k_i is the number of people with the i-th disorder living on the island ($0 \le k_i \le 10\,000$). The next line contains m integers $l_1, l_2, ..., l_m$ — the number of people for each disorder — which are required to obtain using a matching sequence of meetings ($0 \le l_i \le 10\,000$). It is guaranteed that sequences are different.

Output

In the first line output one number — the least number of meetings after which the required distribution of disorders is achieved. In the second line output a sequence of space-separated numbers, each of them is the number of the defect not represented at the current meeting. If there are several possible solutions, output any of them. If there is no solution, print `-1` in the only line.

Examples

changes.in	changes.out
3 3 0 0 0 0 3	-1
3 1 2 3 2 3 1	2 1 2

Note

Clarification to the second example. The least number of meetings is equal 2. After the first meeting in which people with the first defect do not take part, the number of people with the second and third defects will decrease by one, while the number of people with the first defect will increase by two, which will be described by the triplet $(3, 1, 2)$. After the second meeting, in which people with the second defect didn't participate, we will receive the required triplet $(2, 3, 1)$.

Problem F. Oranges (Div-1 only!)

Input file:	`oranges.in`
Output file:	`oranges.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Scientists finally managed to breed a variety of oranges that does not have seeds. However, their joy did not last long. Very soon it became clear that if oranges don't have seeds, then there is nowhere new trees with fruit of the same variety would come from.

Shortly thereafter scientists carried out several experiments and slightly modified the variety — now each tree will give several seedless oranges and exactly one particular red orange, which will ripe only once — at the first harvest, and there will be exactly two seeds in it.

Fruit company «O'Range» bought one seed of the new variety from scientists and started planting orange trees in plantations. After a while, the company's employees noticed some interesting features. Firstly, every season a tree yields the same number of seedless oranges. For instance, the first tree planted in the plantation gives only one seedless orange each harvest. Secondly, if one picks the red orange off the tree, on which x seedless oranges grow, and plant two other trees from its seeds, then one of them will have $(p \cdot x)$ seedless oranges, and the other one — $(p \cdot x - 1)$. Thirdly, red oranges from the trees which give more than $(2 \cdot n)$ oranges, contain abortive seeds, which means that trees cannot grow from them.

Plantations have spread and now all trees that could be grown from one seed grow there.

The company ships oranges in barrels n items in a lot. Besides, the company's trademark feature is that each lot contains harvest from two trees. Now the company's management needs to find out how many ways there are to choose such a pair of orange trees that in one season they would yield exactly n seedless oranges in total.

Input

The only line contains two integers p и n ($2 < p < 10^9$, $0 < n < 10^{2000}$).

Output

Output one number — answer of the problem modulo $10^9 + 7$.

Examples

oranges.in	oranges.out
3 7	2
5 20	1

Problem G. Tag on a graph

Input file:	`runaway.in`
Output file:	`runaway.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The world's first reality show for programmers — "Tag" is being filmed on a graph with n vertices and m edges of equal length. Three players participate in the show: Alice is in vertex $1 \le A \le n$, Bob is in vertex $1 \le B \le n$, and Caesar is in vertex $1 \le C \le n$. The players start to move from their initial position vertices simultaneously and move along the edges at equal constant speed.

Alice's goal is to run down Bob and not to be caught by Caesar; Bob's goal is to run down Caesar and not to be caught by Alice; Caesar's goal is to run down Alice and not to be caught by Bob. The catching act takes place when two or more players meet at one point. This point may be not only vertex but may be a point at some edge. The game finishes then catching act has occured. The player who has achieved his/her goal is announced the winner.

The players do not have to move and can stop and continue at arbitrary points of time.

All players are selfish, asocial, and fanatic. They think independently of each other, identically, trying to use the best tactics. Selfishness means that the only thing each player wants is victory (technically, the situations of being caught and not catching the opponent are both considered a loss). Asociality means that there are no two players who are able to unite against the third one. Fanatism means that even in case of inevitable loss the player will resist as long as possible.

Therefore, the result of the highest priority is victory, it is followed by a draw, and the least desirable result is loss due to any of the two reasons.

The show producers need to find out who will win, if the participants play by the optimal way. If the game lasts endlessly long, or if all players meet at one point, the result will be a draw.

Input

The first line contains one number T — the number of test cases that does not exceed 10 000. Each of the test cases is as follows. The first line of each test contains a pair of integers n and m ($3 \le n \le 10^5$, $0 \le m \le 10^5$). The following m lines contain a pair of integers x, y ($1 \le x, y \le n$) each and describe the graph edge. The last line of each test contains three different integers A, B and C — numbers of vertices where are Alice, Bob, and Caesar located, respectively.

It is guaranteed that the total number of vertices in the input file does not exceed 10^5 and the total number of edges in the input file does not exceed 10^5.

Output

Print T lines. The i-th line contains of name of winner in the i-th test case (`ALICE`, if Alice wins, `BOB`, if Bob wins, `CAESAR`, if Caesar wins) or `DRAW` in case of a draw.

Examples

runaway.in	runaway.out
3 5 4 1 2 2 3 3 4 4 5 1 3 5 4 3 1 2 1 3 1 4 2 3 4 3 3 1 2 1 3 2 3 1 2 3	BOB DRAW DRAW

Problem H. Poplars

Input file:	`trees.in`
Output file:	`trees.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Mayor of Poplartown likes poplars very much and plants them all over the town. He also likes order, that's why all streets in the town comply with one rule — the distance between two adjacent trees on the street is equal for all trees. This rule is followed rigorously, that's why Poplartown's Tree Planting Services have already built large nurseries, which can provide the town with any number of poplar saplings.

Design engineers of a new n kilometers long street divided it into 1 kilometer long sections between side streets. Keeping in mind eccentricity of their Mayor, at the center of each section between side streets they left an area of grass plot where one poplar can be planted. However, by the time municipal services were about to set to work, poplars had grown in k areas!

Unfortunately, it is forbidden to cut any poplars which have been grown. Now municipal services providers want to find out how many ways of planting poplars in the remaining areas there are, provided that "the Mayor's rule" is observed.

Input

The first line contains two numbers: $1 \le n \le 2 \cdot 10^9$, $1 \le k \le 10^5$. The second line contains k numbers sorted in ascending order: positions of the trees which have been already grown.

Output

Output the only number — the number of ways of planting poplars "properly".

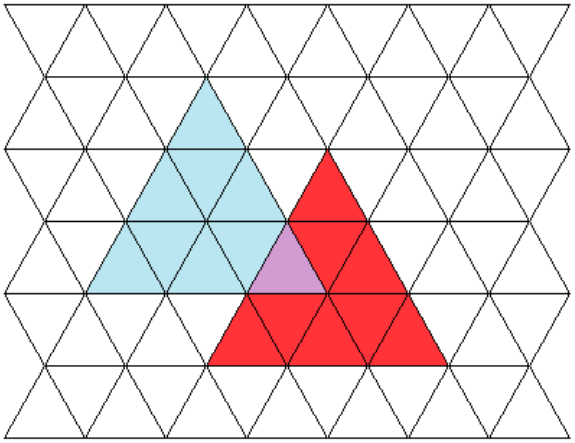
Examples

trees.in	trees.out
4 2 1 4	2
3 1 2	4

Problem I. Triangoogle (Div-1 only!)

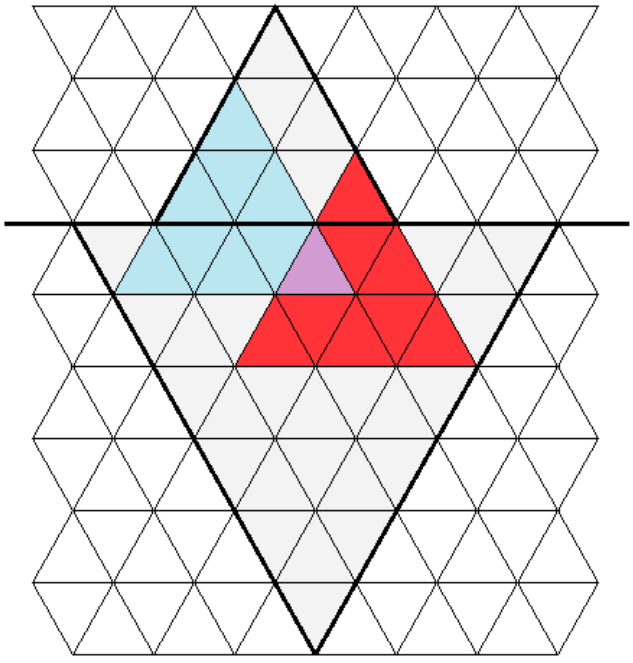
Input file:	`queries.in`
Output file:	`queries.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Search engine Triangoogle, which you need to implement in this problem, processes search queries within a limitless triangle field with n triangles that might have intersections.



You task is to process k queries. Each query is a straight line which cuts the original plane into two half-planes. This straight line is called a search line. Some triangles may be cut by the search line. Your task is to find out how to construct two triangles (probably degenerate) in such a way that each of them lies entirely in its half-plane, one of the sides lies on the search line, and all cut parts of triangles are entirely covered by these two triangles. It is important to minimize the total area of constructed triangles. Each query is applied to the original state of the plane.

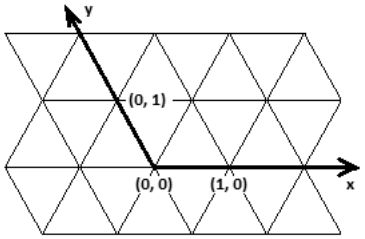
The illustration below shows how to construct triangles on either side of the plane.



Input

The first line contains two integers $1 \le n \le 40\,000$, $1 \le k \le 10^5$.

The following n lines contain six numbers each: the coordinates of triangles' vertices. The coordinates are defined as shown in the illustration below.



It is guaranteed that all triangles are nondegenerate, regular, their sides are parallel to the three axes.

The last k lines contain four numbers each: the coordinates of two different points which are passed through by the search line. It is guaranteed that the search line is parallel to one of the three axes.

All coordinates are integer and do not exceed 10^8 by absolute value.

Output

For each of k queries, in a single line, output one number — the minimal total area of constructed triangles that cover all cut triangles.

Examples

queries.in	queries.out
2 4 1 1 1 -2 -2 -2 0 -1 -3 -1 0 2 -2 0 0 0 -1 1 0 1 1 1 2 2 1 1 1 0	45 26 26 0

Problem J. Islands

Input file:	islands.in
Output file:	islands.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Archipelago «The Green Rat Islands» is a grid $n \times m$, in the nodes of which there are small islands. Two islands are considered neighboring if Euclidean distance between them is no more than 1.

At the center of each island there is an organizer containing k bridges ($0 \le k \le 9$). Number k can be different on different islands and is called the bridge number of an island. An automatic system, operated by means of a control panel, can either connect any of the bridges from the organizer with the neighboring island, or return it back to the organizer. Any bridge can be returned irrespective of the other, and only back to the organizer where it was initially taken from. The bridges can connect only neighboring islands. It is possible to move between two islands only if they are connected by a bridge. A pair of neighboring islands can be connected by more than one bridge.

The green rat (the archipelago's only resident) travels between the islands operating the bridges by means of a control panel. The green rat is on island A right now and wants to find out whether it can reach island B. It would be an easy question but for the islands' legislation, according to which it is strictly forbidden to use the same bridge twice in a row.

Input

The first line contains two integers $1 \le n, m \le 10^5$, $n \cdot m \le 10^5$.

The following n lines contain m numbers each — the bridge numbers of the relevant islands.

The last two lines contain pairs of coordinates of islands A and B respectively, where the first number is the row of the grid, the second one — the columns of the grid.

Output

Output `YES` if it is possible to get from island A to island B, and `NO` if that is not the case.

Examples

islands.in	islands.out
4 6 011110 100011 110001 011112 1 1 2 2	YES
3 3 201 010 030 2 1 2 3	YES
3 3 200 000 002 1 1 3 3	NO

Note

Clarification to the second example — the path from island A to island B looks the following way:

$$(2,1) \to (1,1) \to (1,2) \to (2,2) \to (3,2) \to (2,2) \to (2,3).$$

Problem K. Subpalindromes (Div-1 only!)

Input file:	palindromes.in
Output file:	palindromes.out
Time limit:	4 seconds
Memory limit:	256 megabytes

A palindrome is a character string that may be read the same way both from left to right and from right to left. Palindromology is a science that searches for palindromes wherever they can be found. Today the given string S consisting of lowercase Latin letters is being searched.

Let $S[i, j], i \le j$ denote a substring from character i to j. The task is to calculate the number of substring-palindromes for substring $S[a, b]$. In other words, you need to find out the amount of $S[x, y]$, such that:

- $a \le x \le y \le b$,
- $S[x, y]$ is a palindrome.

Input

The first line contains input string S, the length of which does not exceed 10^5 symbols. The next line contains the number of requests m ($1 \le m \le 3 \cdot 10^5$). Each of the following m lines consists of a pair of integers a and b ($1 \le a \le b \le n$, where n is the length of string S), which denote a request for searching for substring-palindromes in substring $S[a, b]$.

Output

For each request output the answer on a single line.

Examples

palindromes.in	palindromes.out
abaa 4 1 2 1 3 1 4 3 4	2 4 6 3

Problem L. Races

Input file:	`race.in`
Output file:	`race.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Far away in the lost space on a flat planet unusual races are held. They are unusual because the cars taking part in them move at strictly equal speed and cannot turn. One would think, then how is the winner determined? It's quite simple.

There are n drivers consequentially numerated from 1 to n starting race from n starting points. This points are located on one start line. Distance between any neighbour points are equal. Initially, each driver applies his car at a certain angle with the start line. At the start moment all racing cars begin moving at the applied angle simultaneously and they are never change their direction. Each car leaves a deep rut after itself. If another car runs into that rut, it gets stuck there forever. The winners are all those drivers who reach the finish without getting stuck in any ruts. You may assume that the finish line is the further among theoretically possible intersections of the cars' motion paths.

Leader of the extraordinary races federation assumes that the race winners can be determined just knowing the angles of each participant's start.

Input

The first line contains one integer number n ($2 \le n \le 10^5$) — the number of cars. The second line contains n space-separated real numbers α_1, α_2, ..., α_n ($0 < \alpha_i < \pi$), where α_i is the angle of the i-th car in radians, given with no more than five digits after the decimal point. The angles are measured from the start line direction which corresponds to the ascending order of participants' numbers. It is guaranteed that there are no two cars which can reach any point simultaneously.

Output

In the first line output one number — the number of the race winners. In the second line output the numbers of all winners in any order, separated by spaces.

Examples

race.in	race.out
3 1.00 1.57 2.00	1 2
3 2.00 1.57 1.00	3 3 2 1

Problem M. Pharmacies (Div-2 only!)

Input file:	`weights.in`
Output file:	`weights.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Absolutely healthy people live in the Anginsk city. There is only one citizen who is constantly unhappy about this fact. The reason is that he works as a pharmacist and has to sit idly the whole time. Trying to make himself busy, the pharmacist makes up mathematical problems using pharmacy weights and then solves them himself. But one problem turned out to be extremely difficult.

The pharmacist uses a set of N weights of $1, 2^1, 2^2, \ldots, 2^{N-1}$ grams and a two-pan weighing scale. Weights can be put on both scale pans. Two ways of weighing that differ only in arrangement of the scale pans are considered to be the same. For example, there are two ways of weighing a 5 gram weight using three $1, 2^1$ and 2^2 gram weights: $5 = 1 + 4$ and $5 + 1 = 2 + 4$.

The task of the pharmacist is to find out the number of ways of scaling each item of goods in the pharmacy using different sets of weights.

Input

The first line contains the number of goods T ($1 \le T \le 100$).

Each of the following T lines contain two positive integers N_i and M_i — the number of weights that can be used for weighing an item of goods and the weight of this item of goods respectively($1 \le N_i < 63$, $1 \le M_i \le 5 \cdot 10^{18}$).

Output

For each item of goods output the number of possible ways of weighing it, one per line.

Examples

weights.in	weights.out
4 1 1 3 5 5 10 7 17	1 2 5 14

Problem N. Numbers (Div-2 only!)

Input file:	`numbers.in`
Output file:	`numbers.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The most challenging exam for the students of the Recreational Mathematics Department is the exam in numeric shifts. Every year problems in this subject become more and more difficult. For the latest exam Professor Numberski has prepared something unbelievable.

The decimal notation of a certain number begins from a group of digits which will be denoted as B. In other words, the initial number is BX. If we multiply this number by A, the result of multiplication will be a number obtained by rewriting the group of digits B from the beginning to the end of the initial number, which means the final result of multiplication is XB.

The task of students is to find the minimal possible initial number BX based on given A and B.

Input

The only line contains two integers A and B ($2 \le A \le 9$, $1 \le B \le 10^6$).

Output

The only number — solution of the task, or `-1` if there is no solution.

Examples

numbers.in	numbers.out
5 1	-1
3 1	142857

Problem O. Rectadarts (Div-2 only!)

Input file:	darts.in
Output file:	darts.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Everyone likes playing darts, but no one likes calculating the score. This process has been automated for regular darts, but not for rectangular darts.

The game of rectadarts is played on a $n \times m$ board with a Cartesian coordinate system. The boundaries of the board filled with black lines. The player throws each of k darts and definitely hits a point with integer coordinates. After each throw two black lines are drawn from the hitting point parallel to the board sides. Summarizing the game results, the player scores one point for each square drawn with black sides.

The Russian Federation of Rectadarts has announced a reward for an algorithm which calculates the score based on the coordinates of darts.

Input

In the first line — three integers n, m and k ($1 \le n, m \le 10^9$, $0 \le k \le 2000$).

The following k lines contain pairs of integers x and y — coordinates of darts on the board ($0 \le x \le n$, $0 \le y \le m$).

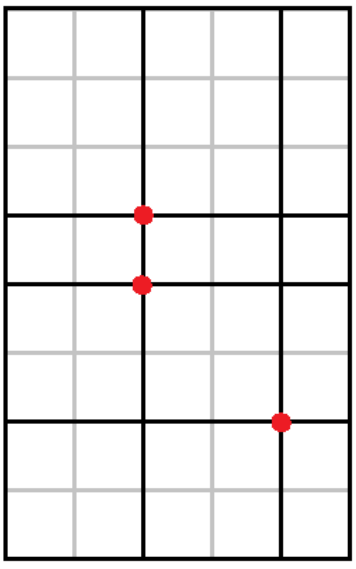
Output

Output one number — the score.

Examples

darts.in	darts.out
5 8 4 4 2 2 4 2 5 4 2	10

Note



Problem P. Wires (Div-2 only!)

Input file:	`wire.in`
Output file:	`wire.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The village of Wires, located not far from Kazan, is well-known for its traditional metal handicrafts — squiggles, which from the earliest times have been made by craftsmen of Wires. Each such article is created on a checkered table which appears to be a Cartesian coordinate system. The squiggler master lays a long metallic rod on the table, starting from the point with integer coordinates. At each stage the master pulls the rod parallel to one of the coordinate axes towards another point with integer coordinates. The squiggler repeats this routine n times, then he anchors the rod at the next point and cuts the excess metal.

It is commonly known that squiggles are extremely valuable due to their authenticity. However, value of each squiggle is determined in a very particular way. Each elbow bend of a squiggle is worth one ruble. A rod that is bent at an angle of 90 degrees is considered an elbow bend.

The union of squigglers is waiting for development of a new software product which will automate calculation of a squiggle cost based on its configuration.

Input

In the first line: $2 \le n \le 100$. The following n lines contain pairs of integers x and y which do not exceed 100 by absolute value — coordinates of an article bends.

Output

Output one number — the price of a squiggle in rubles.

Examples

wire.in	wire.out
9 0 0 0 2 0 3 3 3 3 2 3 1 2 1 1 1 1 4	4
6 1 1 1 5 1 2 1 4 1 3 5 3	1