Problem A. Nanoassembly

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2.5 seconds
Memory limit:	256 MiB

One of the problems of creating elements of nanostructures is the colossal time necessary for the construction of nano-parts from separate atoms. Transporting each of the vast number of atoms requires the use of a manipulator. While the manipulator is busy moving an atom, it can't move other atoms. Using a larger number of manipulators simultaneously is impossible, since manipulators are relatively large and don't fit into the space where the nano-parts are assembled. Experts at the Nanotech State University (NSU) lab proposed a method aimed at solving the problem described above.

The essence of the method consists in abandoning manipulators and substituting them with a semiplanar reflector developed in the laboratory. The device performs manipulations with atoms located on one of the sides of a graphene sheet, which can be viewed as a plane surface. The principle of the semiplanar reflector consists in folding the graphene sheet along a specified straight line, which brings all atoms located on a (specific) side relative to the line to the opposite side, occupying mirrored positions of their initial positions relative to the line. All atoms that were located on the other side of the folding line retain their initial positions. Once all atoms have been transferred, the graphene sheet is unfolded. As the result of several subsequent folding operations, a large number of atoms can be transferred.

Define the positions of the atoms after the manipulations based on their initial position on the graphene sheet surface and the description of the sequence of operations performed by the semiplanar reflector.

Input

The first line of the input file contains two integers N and M — the number of atoms and the number of operations ($1 \le N \le 10^5$, $1 \le M \le 10$).

The following N lines define the initial positions of atoms. Each position is described by two space-separated integer coordinates x and y, which do not exceed 10^4 in absolute value.

The following M lines describe operations performed by the semiplanar reflector. Four space-separated integers x_1, y_1 and x_2, y_2 are used to describe each operation — the coordinates of the start and end points of the vector lying on the folding line (the absolute value of each of the coordinates also does not exceed 10^4). The start and end points are different. The operation transfers atoms located in the semiplane to the right of the defined vector.

Output

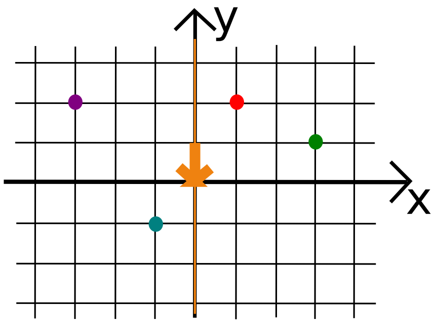
The output file must contain N lines containing the coordinates of the atoms in the same order as in the input data, with absolute error not exceeding 10^{-5}.

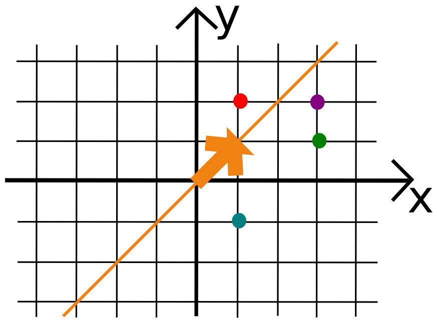
Example

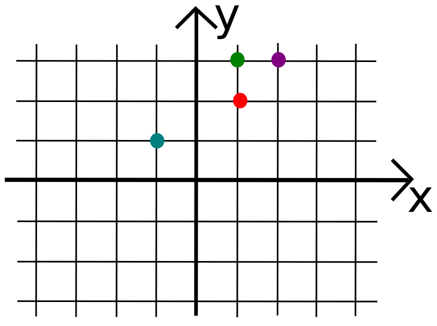
input.txt	output.txt

4 2
1 2
3 1
-1 -1
-3 2
0 1 0 0
0 0 1 1
	1 2
	1 3
	-1.0 1
	2.0 3.0

Note


y
x


y
x


y
x

Problem B. Playoff

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB

Let's consider a tournament based on the Olympic system. There are 2^N participants. They are split into pairs. The winner of each pair continues to the next round. The number of participants of that round is 2^{N-1}. They are split into pairs again, the winner goes to the next round and so on, until the absolute winner is established.

Let's introduce *stronger*, a relationship of teams. If the team A wins in the game of A versus B, then the team A is *stronger* than the team B. Let's consider *stronger* a transitive relationship: if the team A is *stronger* than the team B, and the team B is *stronger* than the team C, then the team A is *stronger* than the team C.

For instance, if the team A beats the team B in semifinals, and C beats D, and then A beats C, we can tell that A is *stronger* than D, however, we can't tell which of the teams B and C is *stronger*.

You are given the results of the tournament based on the Olympic system. Also given are several pairs of teams; define whether one team from each pair is *stronger* than the other.

Input

The first line of the input file contains an integer N ($1 \le N \le 18$).

The following 2^N lines contain the names of the participating teams. The names are unique and consist of no more than 20 small latin letters.

In the first round, in the first pair the first team plays with the second team, in the second pair — the third and fourth and so on. In the second round, the winner of the first pair plays with the winner of the second pair, the winner of the third pair plays with the winner of the fourth pair and so on.

Pairs for the next rounds are matched in the same manner.

All games in the tournament are numbered according to the numeration of pairs in rounds: the numbers from 1 to 2^{N-1} are assigned in order to the pairs from the first round, the numbers of $2^{N-1} + 1$ to $2^{N-1} + 2^{N-2}$ are assigned in order to the pairs of the second round, etc.

The following line consists of $2^N - 1$ symbols '`W`' or '`L`' and describes the results of games in the given order, where the symbol '`W`' denotes the victory of the first team in a pair, and the symbol '`L`' — the victory of the second team.

The next line contains an integer Q — the number of queries ($1 \le Q \le 10^5$).

The following Q lines contain query descriptions. Each query consists of two different space-separated words — the names of the teams.

It is guaranteed that the size of the input file is equal to or less than three megabytes.

Output

The output file must contain one of the following words for each query:

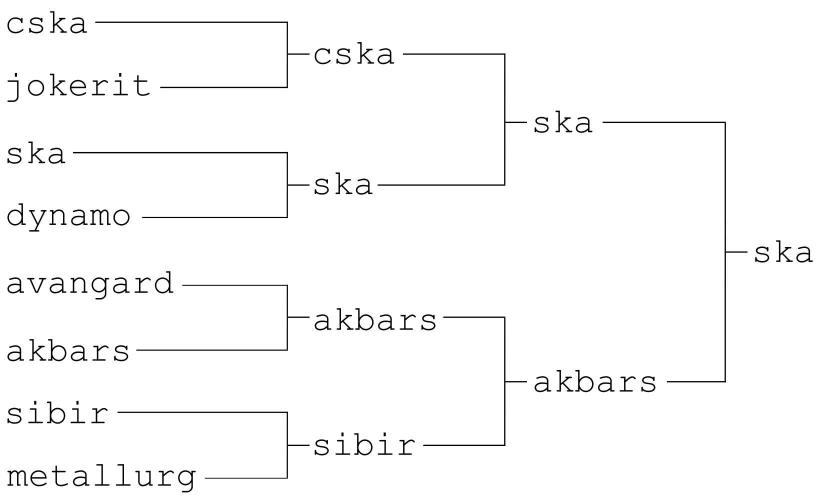
- `Win`, if the first team *stronger* than the second team;
- `Lose`, if the second team is *stronger* than the first team;
- `Unknown`, if it is impossible to tell that one of the teams is *stronger* than the other.

Example

input.txt	output.txt
3 cska jokerit ska dynamo avangard akbars sibir metallurg WWLWLWW 3 jokerit avangard ska metallurg metallurg akbars	Unknown Win Lose

Note

Tournament rounds in the example looks like this:



Problem C. Inequalities

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
	5 seconds (for Java)
Memory limit:	256 MiB

A system of inequalities is given. The inequalities are in the form of $s < t$ and $s \le t$, where each s and t can be an integer constant or an integer variable x_i.

Find an integer solution to this problem, or determine that there is no such solution.

Input

The first line of the input file contains two integers n and k — the number of inequalities and the number of variables respectively ($0 \le n \le 10^5$, $1 \le k \le 10^5$).

The following n lines describe the inequalities, one per line. A description of an inequality consists of five integers. The first integer is the inequality type: `0` — slack ($\le$), `1` — strict ($<$). The second and third numbers encode the left part of an inequality: «`0` i» defines the variable x_i ($1 \le i \le k$), «`1` a» — the integer a ($-10^9 \le a \le 10^9$). The right part of the inequality is identically encoded with the fourth and fifth numbers.

Output

If the inequality system had no integer solution, the output file must contain the word `NO`.

Otherwise the first line must contain the word `YES`, and the following k lines must contain any solution to the system, namely, the i-th line must contain an integer a_i — the value of x_i variable ($1 \le i \le k$, $-2 \cdot 10^9 \le a_i \le 2 \cdot 10^9$).

It is guaranteed that if the system is solvable, there exists a solution which satisfies the constraints.

Example

input.txt	output.txt

2 2
1 0 1 0 2
0 0 2 0 1
	NO

3 2
1 0 1 0 2
0 1 5 0 1
0 0 2 1 7
	YES
	5
	7

Note

The second example describes the following inequality system:

$$\begin{cases} x_1 < x_2 \\ 5 \le x_1 \\ x_2 \le 7 \end{cases}$$

One of its solution is the solution $x_1 = 5$, $x_2 = 7$.

Problem D. How to measure the Ocean?

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 MiB

Cosma Prutkov once wondered what can be used to measure the depth of the Eastern Ocean. For Ostap, the answer is obvious — the astrolabe, naturally. It can do all the measuring as long as you've got something to measure. There is one problem, though: how do you get an astrolabe to the bottom of the ocean. For this purpose, Ostap has got rolls of wire of different diameter made from the different material; the stock of wire of each type is unlimited. All wire in each roll is of the same diameter. You can link wire from different rolls using a very strong glue, i.e. the wire can break anywhere but not at the merging point.

You must chop the pieces of the wires, then arrange them in some order, and glue together, and attach the astrolabe to one end. In particular, you can't glue more than two pieces of wire in one place or attach two wires to the astrolabe. The whole contraption must reach the ocean bottom, and the wire must not break. Wire with a diameter of S made from the material with tensile strength equal to σ can break in any point if the total weight of the astrolabe and the wire below that point exceeds $\sigma \cdot S$.

Determine what the maximum weight may have the astrolabe, so as not to break the wire.

Input

The first line of the input file contains two integers d and n, where d — the ocean depth, n — the number of wire varieties ($1 \le d \le 20000$, $1 \le n \le 10^5$).

Next in n lines are given three integers s_i, p_i and σ_i per line — specifications of ith roll of wire, where s_i — cross-sectional area of the wire in this roll, p_i — the material density of ith roll, i.e. the weight of a unit of volume, σ_i — the material strength limit of the wire in this roll, i.e. the maximum allowed weight per unit of cross section area that the material can handle; if it's exceeded the wire breaks ($1 \le s_i \le 10^3$, $1 \le p_i \le 20$, $10 \le \sigma_i \le 10^3$).

Output

The output file must contain a single integer — the maximum possible weight of the astrolabe, such that it can be lowered to the depth d without breaking the wire. If the wire cannot reach the bottom, print number 0. The absolute error of your answer must not exceed 10^{-5}.

Examples

input.txt	output.txt
10 1 2 8 200	240.00
30 1 2 8 200	0.0

Problem E. Navigation

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB

There are clearings in the forests of Akademgorodok which are linked by straight trails. You can ride a bike along these trails in any direction. All clearings are numbered from 1 to N. You're only allowed to switch trails on clearings. Any trail is a line segment linking two different clearings, and it never passes through a third clearing.

Every year navigation competitions are held in the forest. Each participant must start from the clearing number 1 and finish at the clearing number N. He has to check in at K clearings with the numbers $a_1, a_2, \ldots, a_K$ in the defined order.

During the competition the participant can run straight through the forest or ride along trails on a bicycle. You can borrow or return a bike on every clearing in rent-a-bike points. Every rental point always has available bikes. You can return your bike in any of those rental points, and not only in the one where you got it. It's strictly prohibited to leave your bike on the trail, and you can't run in the forest with a bike. You **are** allowed to cross trails while running.

You can ride along any trail any number of times, and visit any clearing any number of times as well. You don't necessarily have to check in on a clearing the first time you visit it.

Define the sequence of numbers of clearings that you have to ride or run through to be able to check in in the required order. Form your route in such a way that completing it takes the least possible time.

Input

The first line of the input file contains five integers N, M, K, V_r, V_f, where N — the number of clearings in the forest, M — the number of trails, K — the number of clearings where you must check in, V_r — the velocity of your bike on a trail, V_f — the velocity of running through the forest in meters per second ($3 \le N \le 1\,600$, $0 \le M \le 100\,000$, $1 \le K \le 20$, $1 \le V_f \le V_r \le 1\,000$).

The following N lines describe clearings in the order from 1 to N. Each clearing is defined with its own x and y coordinates ($|x|, |y| \le 10\,000$). All coordinates are integers and are defined in meters. There are no two clearings with the same coordinates.

The following M lines describe the trails. Each trail is defined with two different integers — the numbers of the clearings it connects. It is guaranteed that each pair of clearings is connected by no more than one trail.

The last line contains K space-separated integers — the numbers of clearings where you must check in in the required order. All these numbers are different, and are in the range from 2 to $N - 1$.

Output

The first line of the output file must contain a single real number — the minimal time in seconds required to complete the required route.

The next line must contain a sequence of numbers of clearings that a participant rides or runs through. If there is no trail between any two adjacent clearings in the sequence, it is assumed that the participant runs between them directly through the forest, and if such trail exists, the participants rides along this trail.

The absolute or relative error of your answer must be equal or less than 10^{-7}.

Example

input.txt	output.txt

4 3 2 10 4
0 0
100 0
0 400
100 400
1 3
3 4
2 4
2 3
	85.00
	1 2 4 3 4

Problem F. Bets

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 MiB

The most popular type of bets in bookmaker's offices are bets on game results. For instance, in football, a sport where two teams play a game, you can place bets on all three possible outcomes of the match: «first team wins», «draw» and «second team wins». The bookmaker sets a specific coefficient k for every event: if the match ends in a way that has a bet placed on it, the office pays out a sum k times larger than the initial bet, otherwise it pays out nothing.

«Live-bets» are also popular — those are bets placed during the game. As the game progresses, depending on the current score and the remaining time, the coefficients change. It is possible to place a bet on several different outcomes, including all possible outcomes. One of the ways to profit from bets are the so called «forks» — when bets are simultaneously placed on all three events with such sums that regardless of the outcome the gain covers the sum of all three bets.

Naturally, it's not in bookmakers' interests to allow such situations, and a fork bet is not that easy to place: when setting the coefficients bookmakers take into account not only their own preferences, but are forced to consider other bookmakers' policies as well. However, several different methods of placing a «fork» on «live-bets» exist. For example, when different bookmakers' coefficients change after the goal is scored, the short time gap when one office has already changed coefficients and another hasn't can be used to try to make a fork. There are even riskier options which don't guarantee good results. For example, placing a bet on the weaker team in a game of two apparently inequal teams: in case that team scores or holds a draw for long enough, the coefficients for the two remaining outcomes may raise to such values that a fork becomes possible.

In the given task the best coefficients from all bookmakers' offices have already been chosen for the bettor for each outcome. Define whether the current situation is a «fork»: whether or not it is possible to place such bets for each of the three outcomes as to get profit regardless of the match outcome.

Input

The first line of the input file contains an integer Q — the number of queries ($1 \le Q \le 10^4$).

The following Q lines contain queries descriptions. Each query consists of three real numbers w, d, l, defined with no more than 5 digits after the decimal point — the coefficients for the case the first team wins, for a draw, and for the case the second team wins, respectively ($1 \le w, d, l \le 10$).

Output

The output file must contain a single line per each request: if a fork is possible for the data in the request, print out the word `YES`, otherwise print `NO`.

Example

input.txt	output.txt

3
2.5 3.5 4.5
3.0 3.0 3.0
2.0 3.0 4.0
	YES
	NO
	NO

Note

The first query allows bets of \$250, \$200 and \$150: in case the bet wins, the winner gets \$625, \$700 and \$675, respectively, which is more than the sum of the three bets, \$600, in any case.

With the second query, if we bet the same sum on all outcomes, it is guaranteed that we lose nothing, but we won't win anything, either. With different bets there is a chance of loosing.

With the third query, losing is possible with any bets.

Problem G. Ant on the road

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 MiB

An ant has crawled up to a roadside at night. The road is an endless strip limited by the straight lines $y = 0$ and $y = 1$. There are two endless horizontal rows of streetlights: northern and southern streetlights. Each streetlight illuminates a parabolic-shaped patch of the road. Streetlights in both rows are evenly distributed with the same spacing of d.

k-th southern streetlight (k is an integer) illuminates all points with the coordinates (x, y), satisfying the inequality:

$$0 \le y \le a(x - b_k)^2 + c, \text{ where } b_k = b + kd.$$

Analogously, the spotlight for the k-th northern streetlight is described by the inequality:

$$A(x - B_k)^2 + C \le y \le 1, \text{ where } B_k = B + kd.$$

The ant is situated to the south of the road (in the area where $y < 0$) and intends to cross the road (to get to the $y > 1$ area) following the shortest route without entering the spotlights. It can begin and end its way at any point on the corresponding side of the road.

You must determine the length of his route.

Input

The first line of the input file contains a real number d — the distance between streetlights in a row ($0.01 \le d \le 1$).

The second line contains three real numbers a, b, c, where a is the quadratic coefficient of the southern streetlights parabolas ($a < 0$), b is the x-coordinate of one of the southern streetlights, c is the y-coordinate of all southern streetlights. The third and the last line contain three real numbers A, B, C — the analogous parameters of the northern streetlights ($A > 0$).

The streetlight parameters satisfy the restrictions: $0.001 \le -a, A \le 10, 0 \le b, B \le 1, 0.001 \le c, C \le 0.999$.

It is guaranteed that the illuminated areas of any two streetlights do not touch within the error bound of 10^{-6} (but they can intersect each other).

Output

Print a real number to the output file — the minimal distance the ant must cover to cross the road without entering the spotlights. If it is impossible, print -1.

The absolute or relative error of your answer must not exceed 10^{-6}.

Examples

input.txt	output.txt

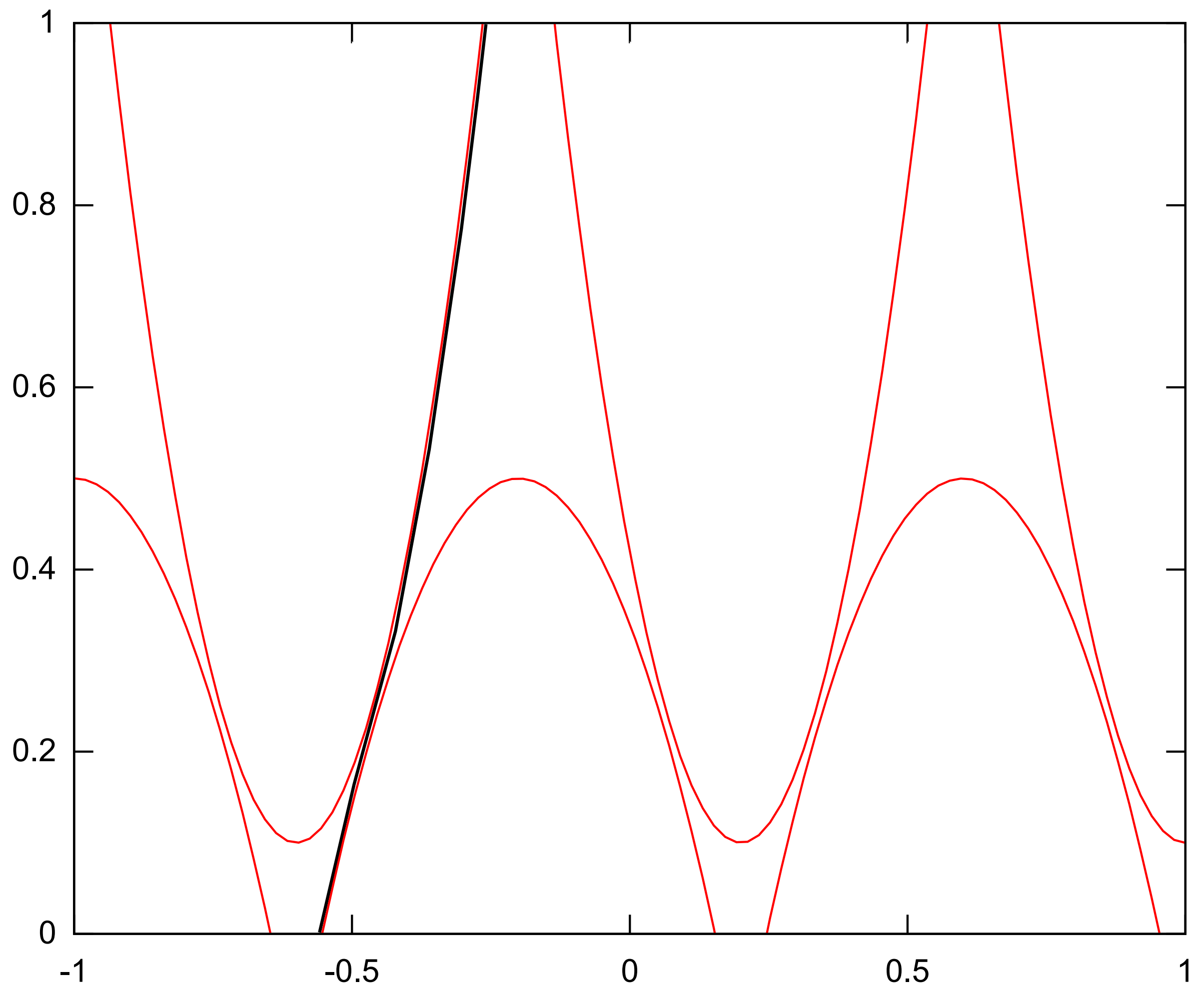
`1.0`
`-1.0 0.0 0.1`
`1.0 0.0 0.9`
	`1`

`1.0`
`-1.0 0.0 0.3`
`1.0 0.0 0.7`
	`-1`

`0.8`
`-4.0 0.6 0.5`
`8.0 0.2 0.1`
	`1.043157963`

Note

Spotlight boundaries in the third sample are as shown:



Problem H. Bouquet

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
	4 seconds (for Java)
Memory limit:	256 MiB

The commander's tender soul desires to make a sacrifice to the beloved woman. With little money left after the collapse of the «Horns and Hooves» enterprise, he's going to spend the rest of it on a gorgeous bouquet. The gorgeousness, in Ostap's eyes, is the number of species of flowers making up the bouquet. However, Mikhail, the old penny pincher, insists that it's not necessary to spend all money, since you can whip up a perfectly good bouquet for less, but no — it's either all or nothing! He MUST spend all his money.

Find the maximum possible number of flower species making up a bouquet of the given cost.

Input

The first line of the input file contains two integers N and S, where N — the number of flowers available for sale, S — the available sum of money ($1 \le N \le 10^3$, $1 \le S \le 5 \cdot 10^4$).

Each of the next N lines describes a single flower, containing two integers a and b — the flower's species index and the price of the flower, respectively ($1 \le a \le 10^3$, $1 \le b \le S$). Every flower on the market is unique, but parameters (the species index and the price) of different flowers may be equal.

Output

The output file must contain a single integer — the maximum possible number of flower species constituting a bouquet with the cost S. If it's impossible to create a bouquet with the given price, print out the word `Impossible`.

Examples

input.txt	output.txt

6 20
2 5
1 5
2 6
3 3
4 4
1 5
	3

1 10 1 8	Impossible

Note

In the first test, you must not buy flower of the 3rd species, because it is not possible to spend all the rest of the money on other flowers. However, you can buy flowers of the remaining three species. To do this, buy flowers cost of 4 and 6 and any two of the three flowers cost of 5.

Problem I. Hash function

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	256 MiB

The following hash function is given:

```
uint32 super_hash_func(uint32 value) {
    uint32 hash = value;
    hash = hash  +  (hash << 10);
    hash = hash xor (hash >> 6);
    hash = hash  +  (hash << 3);
    hash = hash xor (hash >> 11);
    hash = hash  +  (hash << 16);
    return hash;
}
```

Where:

- `uint32` — a 32-bit unsigned integer data type.
- `a + b` — the sum of the integers a and b. Since overflow is possible, the sum is calculated by modulo 2^{32}.
- `a xor b` — a bit-by-bit "XOR" of the numbers a and b. The k-th bit of the result equals 1 if and only if the k-th bits of the numbers a and b differ.
- `a << b` — the operation shifting bits of the number a to the left by b digits. This fills the b lower-order digits of the number with zeroes, and the b higher-order bits are discarded.
- `a >> b` — the operation shifting bits of the number a to the right by b digits. This fills the b higher-order digits of the number with zeroes, and the b lower-order bits are discarded.

You are to reverse the given hash function.

Input

The first line of the input file contains the integer Q ($1 \le Q \le 100$), — the number of queries.

The following Q lines each contain a single unsigned 32-bit integer.

Output

The output file must contain a 32-bit number for each query, such that the value of its hash function equals the number given in the query. It is guaranteed that such number exists for each of the queries.

Example

input.txt	output.txt
4 614278301 1228622139 1841720774 2457244278	1 2 3 4

Problem J. Civilization

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2.5 seconds
Memory limit:	256 MiB

Action in Civilization V, a turn-based strategy, takes place in a hexagonal field — a 2-dimensional grid made up of identical hexagons. Any two tiles sharing a common side are considered adjacent. The tiles can be green plains suitable for building cities and for moving units, or unpassable mountains.

Players take turns. During his or her turn, a player can move units according to the rules described below.

- Each unit at the beginning of a turn has several movement points (MP). Each moves takes away one MP. A unit can move as long as its MP is positive.
- A unit can move to an adjacent tile and such movement spends one MP.
- During a turn several units can be in a single tile simultaneously, but once the turn is over there should be no more than one unit in a single tile.
- Units can't move in the mountains.
- Units can't enter (or move through) tiles occupied by the enemy cities.

Each unit occupies precisely one tile of the field. It is guaranteed that the initial position of units agrees with the rules described above: no more than one unit per tile, and no units in the mountains and cities.

Each city has several hit points (HP). Different cities can have different HP. There are units called catapults. Catapults can perform ranged attacks to cities according to the following rules:

- Only a catapult with a positive MP can attack a city.
- After the attack the MP of the catapult is set to zero.
- You can only attack a city located within two tiles from the catapult position. The attack zone covers the following tiles:
  1. tiles adjacent to the catapult tile;
  2. tiles adjacent to tiles from the first case.

  A catapult can attack a city even if there is a mountain, another city or another catapult between it and the city.
- One attack of each catapult takes away one HP from a city.
- The HP of a city can't be lower than zero. Meanwhile even if a city has a zero HP count, you still can attack it, but with no effect.

The current problem presents the participant with a field with enemy cities and friendly catapults. It's your turn now. During this turn you must eliminate defense from as many cities as possible. To eliminate defense from a city, you must bring down its HP to zero.

Input

The first line of the input file contains two integers W and H — the width and the height of the field ($1 \le W, H \le 10^9$). All military actions are taken within the game field (i.e. in the tiles with coordinates (x, y) within $0 \le x < W$ and $0 \le y < H$), catapults cannot leave the field.

The field consists of hexagonal tiles which form H rows. Each row consists of W tiles, with all tiles oriented in such manner that below and above there are hexagon nodes, and to the left and right there are edges which the tile shares with its adjacent tiles in the row. Every consequent row is shifted relative to the previous row by half a tile. The Ox axis goes from left to right along the lower row of tiles. The Oy axis is inclined 60 degrees relative to the Ox axis. The legend is supplemented with an illustration showing the tiles numeration.

The second line contains an integer N — the number of cities ($1 \le N \le 4$).

The following N lines each contain three numbers x, y and p which describe, respectively, the coordinates of a city and its hit points ($1 \le p \le 100$).

The following line contains an integer C — the number of catapults ($1 \le C \le 100$).

The following C lines describe the catapults: each line contains three numbers x, y and v — the catapult coordinates and its movement points, respectively ($1 \le v \le 5$).

The following line contains an integer M — the number of tiles which are mountains ($0 \le M \le 1\,000$). The following M lines each contain two numbers x and y and describe the coordinates of the mountains. All tiles that are not mountains are plains.

It is guaranteed that all coordinates are pairwise different. For all coordinates x and y the limitations $0 \le x < W$ and $0 \le y < H$ hold true.

Output

The first line of the output file must contains a single integer — the maximum number of cities whose defense can be brought down to zero.

The following C lines must contain three space-separated integers for each catapult: the coordinates of the tile to which the catapult must be relocated and the index of the city the catapult will attack. Cities are numbered starting from 1 in the same order as described in the input file. If a catapult is not going to attack a city print 0 (zero) instead of the city number.

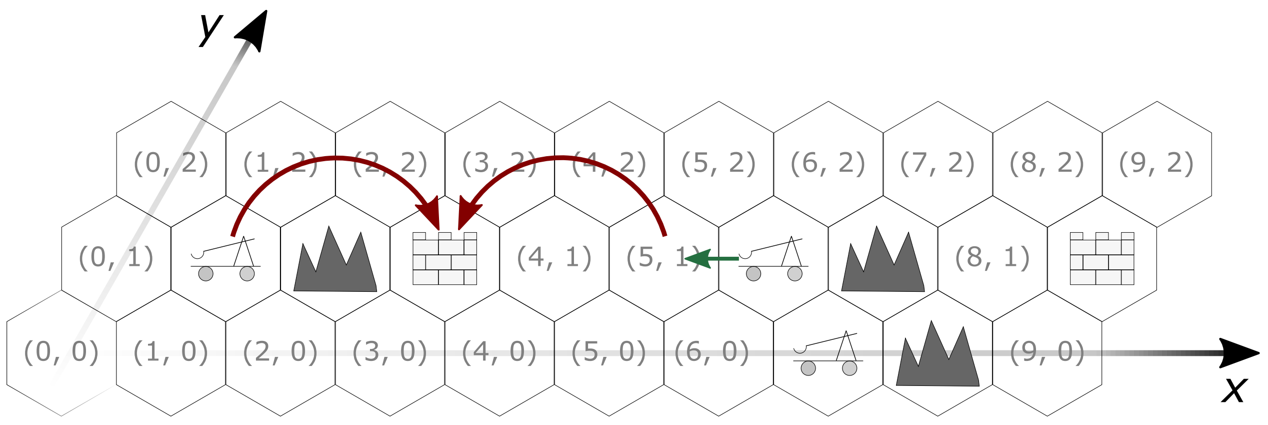
Example

input.txt	output.txt

10 3
2
3 1 2
9 1 1
3
1 1 1
6 1 2
7 0 3
3
2 1
7 1
8 0
	1
	1 1 1
	5 1 1
	7 0 0

Note

Military actions in the example look like this:


y
x
(0, 2)
(1, 2)
(2, 2)
(3, 2)
(4, 2)
(5, 2)
(6, 2)
(7, 2)
(8, 2)
(9, 2)
(0, 1)
(4, 1)
(5, 1)
(8, 1)
(0, 0)
(1, 0)
(2, 0)
(3, 0)
(4, 0)
(5, 0)
(6, 0)
(9, 0)

Problem K. Master Gambs chairs

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 MiB

Father Theodor once visited a furniture show, where they sell furniture of all sorts, made by master Gambs. Needless to say, he is very interested in a particular set of chairs from the house of Vorobyaninov, the head of the nobles. On the one hand, he knows from the head that all furniture at his home was marked, — «in case of all them guests», — on the other hand, he's got no idea what the particular set looks like. Here's a plan he's cooked up: he'll buy chairs from those sets (at least one chair from each set), something he's got the money to buy, and check them. If he finds a mark, then he can buy the remaining chairs from the desired set later on the new money he would get from his «mother».

Naturally, he'd like to check as many different sets as possible, but he's short on money, and he's slightly out of his mind, having spent so much time lurking in the mountains. The task can prove too difficult for him. Help him out! Determine how many sets can he check, so that he bought at least one chair from each set, and he had enough money on all the purchased chairs.

Input

The first line of the input file contains two integers N and S, where N — the number of chairs on sale, S — the available sum of money ($1 \le N \le 10^5$, $1 \le S \le 10^6$).

The following N lines each contain two integers a and b — the index of the set the chair belongs to, and its price, respectively ($1 \le a \le N$, $1 \le b \le 10^5$).

Output

The output file must contain a single integer — the maximum possible number of furniture sets from which it's possible to buy pieces with the total cost not greater than S.

Example

input.txt	output.txt

6 17
2 5
1 5
2 6
2 3
4 400
5 230
	2

Note

Father Theodor can buy one piece from the first set, one or two pieces from the second set. He has not enough money to buy anything from the other sets.

Problem L. Scrabble

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 MiB

Read the rules described in the problem attentively! They may be different from the rules of the original board game!

In «Scrabble», a board game, players score points by building words from available tiles, where each tile has a letter on it, on a 15×15 field — a grid of squares.

At the beginning of the game each player gets 7 tiles with letters from the tile «bank». Players take turns. When his or her turn comes up, a player builds several words (or just one), one by one. When building a word, the player places a non-zero number of his tiles on the field and marks some of the tiles that are already on the field. The number of tiles the player marks when building a word cannot be zero, if there are tiles already present on the field, i.e. if it's not the first word built in the course of the game. Let's call all tiles that the player has placed and marked on the field during building a word word-making tiles. Word-making tiles must be arranged either in a vertical or in a horizontal row. If these tiles are positioned horizontally, the word reads from left to right, and vertically — from top to bottom. The very first word built on the field must have one of its tiles positioned in the central cell of the field. After a player has taken his or her turn, he takes as many tiles as he needs to have a set of seven tiles. If there aren't enough tiles in the bank to make it seven, the player takes all remaining tiles from the bank.

The number of points a player scores during his or her turn is calculated as the sum of points for all words built during this turn. The number of points for a word is calculated based on the values of letters and colors of grid cells on which the word-making tiles are placed (the coloring of the field is described in the table 1). Each letter has a base value in points (base values are listed in the table 3). The color of a grid cell defines the coefficient of letter and the coefficient of word (specific coefficients for colors are provided in the table 2). The number of points for a letter on a tile placed on a grid cell equals the base value of the letter multiplied by the coefficient of the letter corresponding to the cell's color. The number of points for a word equals the sum of points for all the word-making tiles (of that word) multiplied by the product of all the coefficients of the word in all cells occupied by word-making tiles. If a player manages to put all 7 tiles on the field in one turn, he or she gets an additional bonus of 15 points. If a player has less than 7 tiles, he or she cannot get a bonus.

The number of points a player scores at the end of the game equals the sum of all points for all his turns. The sequence of players' turns is given. Calculate the total points scored.

All tabular data provided in the problem are available for downloading in text format (in archive with PDF statements from Yandex.Contest interface).

Input

The first line of the input file contains two integers N and M — the number of players and the number of turns ($2 \le N \le 4$, $1 \le M \le 130$).

M turn descriptions follow. Each description begins with the number W of words built within this turn ($1 \le W \le 7$).

The following W lines describe the words built by the players. Each description begins with the length of the word L, followed by a space-separated symbol showing the word direction: (`'h'` — for horizontal words and `'v'` — for vertical words), followed by the coordinates of the first letter of the word X, Y ($2 \le L \le 15$, $1 \le X, Y \le 15$). X is the column number, and Y is the row number. Columns are numbered from left to right, and rows — from top to bottom. Next follow L numbers encoding the letters of the built word.

Output

The output file must contain N lines with one number per line — points scored by the players based on M turns taken.

Example

input.txt	output.txt
2 6 2 3 h 7 8 13 6 24 5 v 9 6 17 28 24 1 4 1 5 h 5 9 3 6 18 14 1 2 4 h 6 10 11 17 20 4 4 h 7 6 11 15 17 5 3 5 v 6 8 18 6 11 1 24 4 v 7 3 32 26 9 11 3 h 8 9 14 1 10 3 5 h 4 12 16 15 24 11 1 4 v 8 10 20 12 1 14 2 h 7 4 26 9 4 3 h 7 4 26 9 19 4 v 7 9 18 17 15 11 4 h 2 12 5 6 16 15 4 h 7 12 11 1 16 1	121 102

Note

The players took 6 turns in the example. First, the first player built the words «МЕЧ» and «РЫЧАГ». Next, the second player built the word «ВЕСНА». On the third turn, the first player built the words «КРУГ» and «КОРД». On the fourth turn the second player built the words «СЕКАЧ», «ЯЩИК» и «НАЙ». Next was the first player's turn, and he built the words «ПОЧКА», «УЛАН», «ЩИ». The last turn was the second player's, and he built the words «ЩИТ», «СРОК», «ДЕПО» и «КАПА». The table 4 shows the state of the game field in the end.

Таблица 1: Game field

🟥	⬜	⬜	🟩	⬜	⬜	⬜	🟥	⬜	⬜	⬜	🟩	⬜	⬜	🟥
⬜	🟦	⬜	⬜	⬜	🟨	⬜	⬜	⬜	🟨	⬜	⬜	⬜	🟦	⬜
⬜	⬜	🟦	⬜	⬜	⬜	🟩	⬜	🟩	⬜	⬜	⬜	🟦	⬜	⬜
🟩	⬜	⬜	🟦	⬜	⬜	⬜	🟩	⬜	⬜	⬜	🟦	⬜	⬜	🟩
⬜	⬜	⬜	⬜	🟦	⬜	⬜	⬜	⬜	⬜	🟦	⬜	⬜	⬜	⬜
⬜	🟨	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟨	⬜
⬜	⬜	🟩	⬜	⬜	⬜	🟩	⬜	🟩	⬜	⬜	⬜	🟩	⬜	⬜
🟥	⬜	⬜	🟩	⬜	⬜	⬜	⬛	⬜	⬜	⬜	🟩	⬜	⬜	🟥
⬜	⬜	🟩	⬜	⬜	⬜	🟩	⬜	🟩	⬜	⬜	⬜	🟩	⬜	⬜
⬜	🟨	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟨	⬜
⬜	⬜	⬜	⬜	🟦	⬜	⬜	⬜	⬜	⬜	🟦	⬜	⬜	⬜	⬜
🟩	⬜	⬜	🟦	⬜	⬜	⬜	🟩	⬜	⬜	⬜	🟦	⬜	⬜	🟩
⬜	⬜	🟦	⬜	⬜	⬜	🟩	⬜	🟩	⬜	⬜	⬜	🟦	⬜	⬜
⬜	🟦	⬜	⬜	⬜	🟨	⬜	⬜	⬜	🟨	⬜	⬜	⬜	🟦	⬜
🟥	⬜	⬜	🟩	⬜	⬜	⬜	🟥	⬜	⬜	⬜	🟩	⬜	⬜	🟥

Таблица 2: Cell color meaning

Color	Code	Letter coefficient	Word coefficient
White	w	1	1
Black (denotes starting cell)	s	1	1
Green	g	2	1
Yellow	y	3	1
Blue	b	1	2
Red	r	1	3

Таблица 3: Letters' base value

The number of letter used	Letter	Base value
1	А	1
2	Б	3
3	В	2
4	Г	3
5	Д	2
6	Е	1
7	Ж	5
8	З	5
9	И	1
10	Й	2
11	К	2
12	Л	2
13	М	2
14	Н	1
15	О	1
16	П	2
17	Р	2
18	С	2
19	Т	2
20	У	3
21	Ф	10
22	Х	5
23	Ц	10
24	Ч	5
25	Ш	10
26	Щ	10
27	Ъ	10
28	Ы	5
29	Ь	5
30	Э	10
31	Ю	10
32	Я	3

Таблица 4: Game field after turns were taken

🟥			🟩				🟥				🟩			🟥
	🟦				🟨				🟨				🟦	
		🟦				🟩 Я		🟩				🟦		
🟩			🟦			Щ	🟩 И	Т			🟦			🟩
				🟦		И				🟦				
	🟨					К	О	Р	Д				🟨	
		🟩				🟩		🟩 Ы				🟩		
🟥			🟩		С	М	▨ Е	Ч			🟩			🟥
		🟩		В	Е	🟩 С	Н	🟩 А	Й			🟩		
	🟨				К	Р	У	Г					🟨	
				🟦	А	О	Л			🟦				
🟩	Д	Е	🟦 П	О	Ч	К	🟩 А	П	А		🟦			🟩
		🟦				🟩	Н	🟩				🟦		
	🟦				🟨				🟨				🟦	
🟥			🟩				🟥				🟩			🟥