Problem 1. Polesov and work

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 megabytes

Viktor Mikhailovich Polesov, being a true representative of the intelligentsia, is intelligent and fast-thinking. He has recently learned that work of a force is the dot product of the force vector by displacement vector. Naturally, he wants to maximize the work, so that nothing goes to waste. All he has to do is choose where to move.

It is possible to move from the point $(0, 0)$ to any integer point in the circle $x^2+y^2 \le R^2$. The force vector is known — it is the same everywhere and has the coordinates (a, b). Find the maximum work that Polesov is so curious about.

Input

The first line of the input file contains a single integer T — the number of tests ($1 \le T \le 1\,000$). It is followed by T lines, each containing three integers a, b — force vector coordinates, and R — the radius of the circle ($-10^9 \le a, b \le 10^9$, $1 \le R \le 10^9$).

Output

For each test, print a single integer — the maximum work.

Example

input.txt	output.txt
3 10 -10 2 2 3 3 5 1 3	20 10 15

Example explanation

In the first test, the circle with the center at the origin of coordinates and a radius of 2 contains 13 integer points. For the points $(2, 0)$, $(1, -1)$, $(0, -2)$, the dot product by the force vector $(10, -10)$ is maximal. For instance, $10 \times 1 + (-10) \times (-1) = 20$.

Problem 2. Orienteering

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 megabytes

Khodislav has taken up orienteering and is participating in a contest. The field is an endless plane without obstacles, and he can move at the same speed in all directions. There are N checkpoints in the field, which every participant must visit in the ascending order of their numbers. The check-in system is contactless — each checkpoint has a base station that automatically checks in any participant in a range less than or equal to R. It is guaranteed that checkpoint coverage areas do not overlap, but they can touch each other.

A participant starts at any point of the first checkpoint coverage and finishes at the moment of checking in at the last checkpoint. Participants are allowed to enter other checkpoint coverage areas on their way to the necessary checkpoint, but in this case, they are not checked in there.

Khodislav is feeling lucky and believes he will be able to cover the distance optimally. Help him calculate the distance he will have to cover.

Input

The first line of the input file contains two integers: N — the number of checkpoints ($2 \le N \le 100$) and R — the check-in radius ($1 \le R \le 10^9$).

Next come N lines describing the checkpoints in the required check-in order. Each line is a pair of integers x_i, y_i with the coordinates ($-10^9 \le x_i, y_i \le 10^9$).

Output

Print one real number — total distance passed if the route is optimal.

The relative or absolute error must not exceed **0.01**. This means that if the optimal answer equals X, your answer must differ from X by no more than $\frac{1}{100}\max(X, 1)$.

Examples

input.txt	output.txt
3 3 0 -3 0 100 0 50	141.0

Example explanation

Khodislav starts at $(0, 0)$, checks in at the second checkpoint at $(0, 97)$, turns and checks in at the third checkpoint at $(0, 53)$. The total distance covered is $97 + 44 = 141$.

Problem 3. Dice

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

Khodislav is playing a tabletop role-playing game. He has finally chosen his weapons to deal with a monster and casts the crushing strike. To do this, he rolls dice, calculates the sum of numbers on their faces, and says it aloud to the game master.

Rolling a group of identical dice is characterized by three numbers n, f, and m, where n is the number of the dice, f is the number of faces on each die, and m is the modifier. The faces carry all numbers from 1 through f, and each and any face can be rolled; all rolls are independent. For instance, if $n = 3$, $f = 8$, $m = 5$, to define the sum, the player must roll three eight-faced dice, sum up the results, and add five: this is usually written as $3d8 + 5$.

The game master wants to check if Khodislav could get the sum he has reported after rolling the dice.

Input

The first line of the input file contains a single integer B — the number of strikes ($1 \le B \le 10^5$) cast by Khodislav. The following lines describe the strikes, one per line. First comes an integer S — the sum reported by Khodislav. It is followed by three integers: n, f and m describing the group of dice ($1 \le S \le 300$, $1 \le n \le 10$, $2 \le f \le 20$, $0 \le m \le 10$).

Output

For each strike in a separate line, in the same order as in the input file, print `YES`, if the sum was achievable, and `NO` otherwise.

Examples

input.txt	output.txt
5 3 1 6 0 1 1 8 1 16 1 12 3 1 2 4 0 42 3 20 1	YES NO NO NO YES

Problem 4. Numeral systems

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 megabytes

Khodislav has learned that apart from the decimal numeral system, there exist other systems, such as the hexadecimal system. He is curious about the connection of the values of the same notation in these two systems. For instance, can a sequence of K digits have its hexadecimal value divisible by its decimal value? Will anything change if we subtract a specified number D from the decimal value of this notation?

A notation is a sequence of digits. Digits from 0 through 9 can be used in a notation: they belong both to the decimal and to the hexadecimal systems. A notation of a number cannot begin with the digit 0. The decimal value of a notation is the number resulting from interpreting the notation as a decimal number. The hexadecimal value of a notation is the number resulting from interpreting the notation as a hexadecimal number.

Input

The first line contains an integer T — the number of tests in the file ($1 \le T \le 100$). It is followed by T tests, one per line.

For each test, two integers are provided: K — the number of digits in the notation and D — the number to be subtracted from the decimal value ($2 \le K \le 15$, $0 \le D \le 10^6$).

Output

For each test, find all required notations. Notations are sequences of K digits without leading zeroes, with the value of X in the decimal system and the value of Y in the hexadecimal system, for which $X > D$ and Y is divisible by $(X - D)$ without remainder.

The answers to each test are printed in a single line. First, print the number of the found notations, followed by all found notations in the ascending order, separated by spaces.

Example

input.txt	output.txt
4 5 6 2 0 2 5 2 21	3 12510 24990 37950 0 2 16 22 4 22 24 34 96

Example explanation

In the first test, the notation `37950` means $X = 37950$ in the decimal system and $Y = 227664$ in the hexadecimal system. We can see that $Y = 227664$ is divisible by $X - D = 37944$ without remainder. Identically, for the notation `24990`, $Y = 149904$ is divisible by $X - D = 24984$, and for the notation `12510`, $Y = 75024$ is divisible by $X - D = 12504$.

Problem 5. Hockey

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

Rules described in this problem differ from the conventional hockey rules.

A hockey match lasts 60 minutes, with two teams trying to score as many goals as possible. A hockey team consists of five field players and a goalkeeper.

Penalties are an important part of hockey. A field player can be given a penalty: in this case, the offending player leaves the ice for a period of time which depends on the violation. As the result, the number of players on the ice temporarily decreases for the team the offending player belongs to. There are two types of penalties in hockey: *major* and *minor*. A major penalty means the player leaves the ice for five minutes; with minor penalty, it is two minutes. When penalty time runs out, the player returns to the ice.

A minor penalty can be ended prematurely. A team is said to be playing short-handed when it has less players on the ice than the other team. If a team is playing short-handed and opponent scores a goal, then one of its players with minor penalty returns to the ice with his penalty expired ahead of time. If the team has several players with minor penalty, only the player who got the penalty first returns to the ice. If there are no players with minor penalties in the team, no one returns ahead of time.

Penalties during the game mean that the teams can play in various formats regarding the number of field players on the ice. We will denote the game format by `AxB`, meaning the first team currently has `A` field players on the ice, and the second team has `B`. For instance, in the beginning of the game each team has five players on the ice, and this format is denoted as `5x5`. If the first team currently has two players with penalty, and the second team has one, the format is denoted as `3x4`.

You are given a game protocol, registering the time of all penalties and goals. Calculate which formats happened during the game and for how long each format was played.

Input

The first line of the input file contains an integer N — the number of events in the match ($0 \le N \le 1\,000$).

The following N lines describe the events of the match, one per line. Events are described in the following format:

```
mm:ss.d team type
```

Where `mm:ss.d` — time of event with the precision of tenths of a second ($0 \le$ `mm` ≤ 59, $0 \le$ `ss` ≤ 59, $0 \le$ `d` ≤ 9), `team` — team number (either 1 or 2), `type` — event type:

- `goal` — team scores a goal;
- `minor` — team player receives minor penalty;
- `major` — team player receives major penalty.

It is guaranteed that events of the type `goal` have non-zero decimal of a second, i.e. `d` $\neq 0$, and events of the type `minor` and `major` always have zero decimals of a second, i.e. `d` $= 0$.

Events are listed chronologically, i.e. they are arranged in the order of non-reduction of event times. It is guaranteed that at any moment of time each team has no more than 5 players.

Output

For each format of the game in which the teams have played non-zero time, print the format denotation and the time spent by the teams in this format in a separate line, separated by a space character. The format of time must be exactly the same as the format used in the input data. Lines can be printed in arbitrary order.

Example

input.txt	output.txt
10 06:41.0 1 minor 07:20.4 2 goal 22:22.0 2 minor 22:32.0 1 minor 23:00.1 1 goal 23:12.0 2 minor 23:59.9 1 goal 41:02.0 1 major 41:04.5 2 goal 59:00.0 1 minor	4x3 00:47.9 4x4 01:12.1 4x5 06:39.4 5x4 00:50.0 5x5 50:30.6

Example explanation

The game from the example had the following intervals:

- `[00:00.0; 06:41.0)` — until the first penalty, the game went in the initial format `5x5`;
- `[06:41.0; 07:20.4)` — after a penalty, the teams were playing in the format `4x5` until the first team lost a goal while playing short-handed, and the player removed for a minor penalty returned to the ice;
- `[07:20.4; 22:22.0)` — the teams were playing in full numbers `5x5` until a penalty;
- `[22:22.0; 22:32.0)` — until the next penalty, the teams were playing in the format `5x4`;
- `[22:32.0; 23:00.1)` — until a goal was scored, the teams were playing in the format `4x4`, however, no players returned to the ice after the goal, because it was scored with equally-sized teams;
- `[23:00.1; 23:12.0)` — the teams continued playing in the format `4x4` until another penalty;
- `[23:12.0; 23:59.9)` — after that, the teams were playing in the format `4x3` until a goal was scored, and the second team player who had been penalized at `22:22.0` returned to the ice;
- `[23:59.9; 24:32.0)` — the teams were playing `4x4` until the first team player's penalty ran out;
- `[24:32.0; 25:12.0)` — the teams were playing in the `5x4` format until the second team player's penalty ran out;
- `[25:12.0; 41:02.0)` — the teams were playing in the full format `5x5` until a major penalty;
- `[41:02.0; 41:04.5)` — before the goal, the teams played in the format `4x5`, but because a player of the team that lost a goal had a major penalty, that player does **not** leave the penalty box;

- `[41:04.5; 46:02.0)` — the teams continued playing as `4x5`, until the first player's penalty ran out;
- `[46:02.0; 59:00.0)` — before the penalty, the teams were playing with all players `5x5`;
- `[59:00.0; 60:00.0)` — teams ending the game in the format `4x5`.

Problem 6. Days of the week

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

The Architect has created a multiverse with M universes. In each universe, the day of the week has been set individually. The Architect also has N buttons. Each button is connected to a certain set of universes. Pressing a button shifts all of its connected universes one day forward.

The Architect is curious: is there such a configuration of weekdays in the universes which is unachievable by any sequence of pressing buttons? Help the Architect solve this problem.

All universes use the Earth week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. When shifting the day of the week forward: Monday changes to Tuesday, Tuesday to Wednesday, etc, and Sunday changes to Monday.

The Architect can press buttons any number of times and in any order. When a button is pressed, all its connected universes change instantly and simultaneously.

Input

The first line contains the integer T — the number of tests in the input file ($1 \le T \le 500$). It is followed by the tests.

Each test begins with a line containting two integers: N and M — the number of buttons and the number of universes, respectively ($1 \le N, M \le 500$).

The second line of the test contains the initial configuration: which day of the week has been initially set in each universe. The universes are numbered from 1 through M: days of the week in the universes in the configuration are written in the same order.

This is followed by descriptions of buttons. For each button, a separate line defines which universes it is connected to, in the following format: the first number K defines how many universes are connected, and the following K numbers denote the numbers of these universes ($0 \le K \le M$). It is guaranteed that the specified universe numbers are different for each button.

It is guaranteed that the total number of buttons over all tests does not exceed 500, and the total number of universes over all tests also does not exceed 500.

Output

For each test, print an answer in a separate line.

If all 7^M possible configurations of days of the week are obtainable, print the word `NO` as the answer. Otherwise print `YES`, followed by any unachievable configuration, separated by a space symbol.

Examples

input.txt	output.txt
3 3 3 Monday Saturday Thursday 1 1 1 2 1 3 3 3 Friday Thursday Thursday 1 3 1 3 1 3 4 3 Sunday Sunday Monday 2 1 3 3 1 2 3 0 1 3	NO YES Saturday Thursday Thursday NO

Problem 7. Balls

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

Alice and Bob are playing a game. They've got B blue and R red balls laid out in front of them. Alice has the first move, and after that, players alternate moves. Alice picks a single random ball and removes it. Bob removes a single red ball.

Alice chooses her balls randomly with equal probability regardless of their color. It does not matter which red ball Bob removes.

The game ends when one of the two outcomes occurs:

- there are no more blue balls — Alice wins;
- there are strictly more blue balls than there are red balls — Bob wins.

Alice and Bob would like a balance of outcomes, and are curious as for what number of blue balls is necessary for a game of $C = B + R$ balls for the probability of Alice winning h to be as close to 50% as possible. In other words, they want to minimize the value $|h - 0.5|$.

Input

The first line of the input file contains a single integer G — the number of games Alice and Bob are going to play ($1 \le G \le 10^5$).

The following lines define the number of balls C in each game ($2 \le C \le 2 \cdot 10^5$), one line per game.

Output

For each game in a separate line, in the same order as in the input file, print the number of blue balls necessary for the chance of Alice's victory to be as close as possible to 50%.

Examples

input.txt	output.txt
5 2 3 6 7 8	1 1 2 1 2

Problem 8. Romualdych and remainders

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

Old man Romualdych learned about division with remainders and it just threw him off the hinges. The thing got him all mixed up and sweating like a pig. What a sad sight he was, hoping to find some number x in the interval $[a, b]$, that would produce the specified remainder r when divided by some number y. Let's face it — Romualdych is not the sharpest tool in the shed, and even if he sinks his few remaining teeth into the task, he is not likely to cope without your help.

Input

The first line contains a single integer T — the number of tests in the file ($1 \le n \le 200\,000$).

Each of the following T lines contains three integers: a, b — interval bounds, and r — required remainder ($0 \le a \le b \le 10^{18}$, $0 \le r \le 10^{18}$).

Output

Print T answers in the same order as the tests in the input file are given, one answer per line.

Each answer consists of two integers x and y, such that $a \le x \le b$, $1 \le y \le 2 \cdot 10^{18}$, and the remainder from the division of x by y equals r. If there are several possible answers that fit all the requirements, choose any answer with the minimal x. If there are no possible answers, print two integers: $x = -1$ and $y = -1$.

Example

input.txt	output.txt
2 6 8 0 3 5 10	6 3 -1 -1

Example explanation

In the first test, 6 is divided by 3, and the remainder is indeed 0. Since 6 is the smallest number in the interval $[6, 8]$, this is the correct answer. Instead, the following answer can be printed too: $x = 6$ and $y = 2$ (minimizing y is not required), while the answer $x = 8$ and $y = 4$ cannot be printed, because its x is not minimal.

In the second test, there are no answers, since it is impossible to get a remainder of 10 for x in the interval [3,5] regardless of the y it is divided by.

Problem 9. Polynomials

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

On the left side of the board, there are N polynomials, and on the right side, there are M polynomials. Your task is to *construct* each of the M polynomials from the right side of the board with the minimum number of actions.

To construct a polynomial from the right side, first choose any of the N polynomials from the left side. The following transformations can be applied to the chosen polynomial:

- Differentiation: a polynomial is replaced by its derivative.
- Integration: a polynomial is replaced by its antiderivative with an arbitrary integration constant.

Transformations can be applied in arbitrary order any number of times, however, one application counts as one action. You can use no transformations at all if a polynomial from the right side matches some polynomial from the left side.

Input

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

Each of the following $N + M$ lines describes the polynomials, one polynomial per line. The first N polynomials are polynomials from the left side of the board, the rest are from the right side of the board.

The description of the polynomial $a_0 + a_1 x + \cdots + a_K x^K$ begins with a nonnegative integer K — its degree. Next come integers $a_0, a_1, \ldots, a_K$, which are the coefficients of the polynomial ($-10^9 \le a_i \le 10^9$). Herewith, $a_K \ne 0$.

It is guaranteed that the sum of degrees of all polynomials on the left side of the board is not greater than 10^5. It is the same for the polynomials on the right side of the board.

Output

For each polynomial from the right side, print the minimum number of actions necessary for its construction. Print your answers one per line in the same order as the order in which the polynomials are listed in the input file.

Examples

input.txt	output.txt
2 1 2 1 1 1 2 7 6 2 2 7 6 1	4
2 1 2 1 1 1 0 1 1 1 1	1

Example explanation

In the second example there are two polynomials on the left side of the board: $p_1(x) = 1 + x + x^2$ and $p_2(x) = 7 + 6x + 2x^2$. We need to obtain the polynomial $q(x) = 7 + 6x + x^2$. In order to do that we apply differentiation to p_1 twice obtaining $p_1'(x) = 1 + 2x$ first, and $p_1''(x) = 2$ next. Now, let's integrate $p_1''(x)$ with 6 as the integration constant producing $6 + 2x$. We integrate the result once again with 7 as the integration constant to get $7 + 6x + x^2$. We used 4 actions in total.

Problem 10. Find the vault

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	6 seconds
Memory limit:	512 megabytes

Khodislav is playing his favorite roguelike game. Every level of the game is a rectangular grid, and each cell is either empty or containing a wall. Until the player gets near the cell, they have no way of knowing what is in it.

Vault rooms, which are especially valued, can be hidden in such a manner that the player cannot reach them by usual means. To get into such a room, the player must cast a spell that uncovers a portion of cells on that level, as well as the teleportation spell. Luckily, the wiki page of the game contains a layout of the vault, which is rectangular. The vault can be located anywhere in the game level, but its orientation must exactly match the layout on the wiki.

Khodislav has already unlocked a part of the level and is curious as for where the vault can be. Find all possible positions of the top left corner of the vault room that do not contradict what is already known about the level. Note that the vault must fully fit into the level.

Input

The first line contains four integers: R, C — the number of rows and columns in the game level, respectively, and A, B — the number of rows and columns in the vault layout, respectively ($1 \le R, C \le 2\,000$, $1 \le A \le R$, $1 \le B \le C$).

It is followed by the map of the level: R lines each containing C characters. For each cell, one of the three characters is provided:

- '`#`' (ASCII 35) — the cell is wall,
- '`.`' (ASCII 46) — the cell is empty,
- '`_`' (ASCII 95) — the contents of the cell are unknown.

The remaining A lines describe the vault layout, B characters per line. For each cell, one of the three characters is provided:

- '`#`' (ASCII 35) — the cell must be wall,
- '`.`' (ASCII 46) — the cell must be empty,
- '`_`' (ASCII 95) — the cell can be anything.

Output

In the first line, print an integer K — the number of possible positions of the vault ($0 \le K \le (R - A + 1) \cdot (C - B + 1)$). In the remaining K lines, print these positions, one per line. Each position is defined by two space-separated integers u and v — the indices of the row and column in the level where the top left cell of the vault layout is located ($1 \le u \le R - A + 1$, $1 \le v \le C - B + 1$).

The positions must be arranged in the ascending order of u, and for equal u — in the ascending order of v.

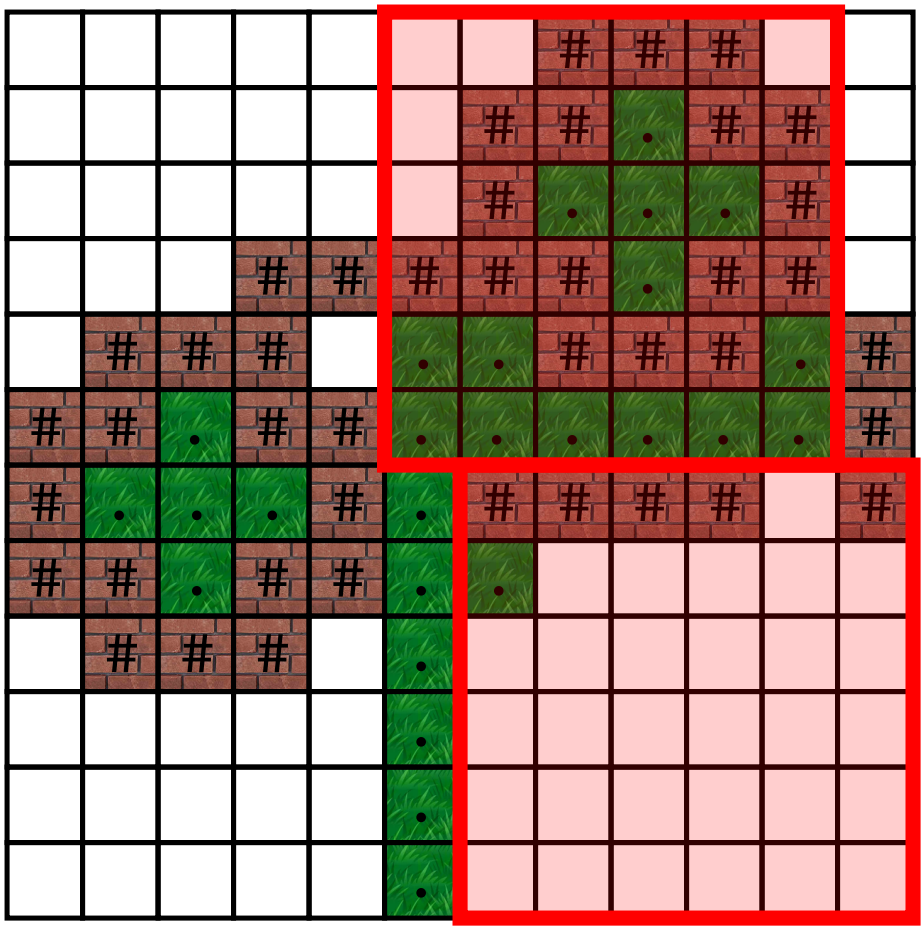
Examples

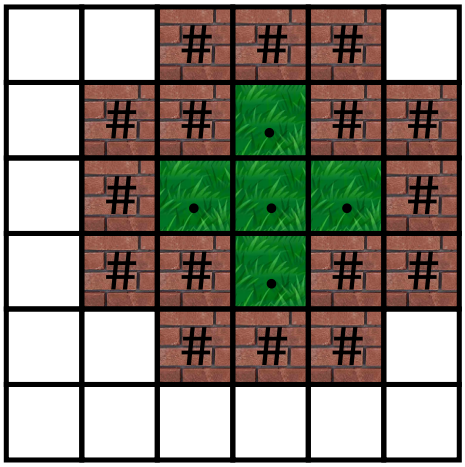
input.txt	output.txt
12 12 6 6 _____###__ _____##.##_ _____#...#_ ___#####.##_ _###_..###.# ##.##.....# #...#.####_# ##.##.._____ _###_._____ _____·_____ _____·_____ _____·_____ __###_ _##.## _#...# _##.## __###_ _____	2 1 6 7 7
4 5 2 3 _____ _._## _·_·_ __#__ _## ##.	0

Illustration

The illustration to the first sample is given on the next page.

Two possible positions of the vault are shown with bold red frames. In the bottom position, contents of the most of the cells are unknown: only two cells are precisely defined both on the level map and on the vault layout. A structure closely resembling a vault can be seen on the left. However, it does not fully fit the level map without an additional column on the left.





Problem 11. Captivating process

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
Memory limit:	512 megabytes

Yulia wrote the number x on the blackboard, and Zakhar wrote y. The kids are bored and have come up with an extremely captivating activity. Once every minute, they erase their numbers simultaneously and write new numbers instead. Yulia writes new numbers according to the following rule: if her number equaled i, it is substituted by f_i. Zakhar does the same, but the rule is different: if his number equaled i, it is substituted by g_i.

They will stop when their numbers match. This could happen right away (if $x = y$), or later, or maybe never. Your task is to determine for different values of x and y, if the kids will ever end writing out numbers.

Input

The first line contains two integers: N and Q ($1 \le N, Q \le 10^5$).

The second line contains N numbers separated by spaces: $f_1, \ldots, f_N$.

The third line contains numbers in the same format: $g_1, \ldots, g_N$.

In the jth of the following Q lines there are the initial numbers x_j and y_j.

It is guaranteed that the numbers f_i, g_i, x_j, y_j are all integers and fall within the range of 1 through N.

Output

Print Q lines: in the jth line, print `YES`, if the process that started from the numbers x_j and y_j ends, and `NO` otherwise.

Examples

input.txt	output.txt
3 2 2 3 1 2 3 1 1 2 1 1	NO YES
4 2 2 3 4 2 2 4 4 1 1 2 1 4	NO YES