Problem A. Stickers for BSUIR Open

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Someone suggested the idea of handing out a sticker with the BSUIR Open logo to each participant. To do this, a rectangular sheet of self-adhesive paper measuring $n \times m$ was found at the university. In total, k participants will come to the competition, so you need to cut out exactly k logos in the form of a square of the same size. Now it remains to determine the maximum size of each logo sticker.

Unfortunately, the organizers have no free hands for this task, so you are asked to help determine the maximum size of the logo (the length of the side of the square).

Apparently stickers will be next year...

Input

The first line contains three integers n, m and k — the width, the height of the sheet of paper and the number of participants respectively.

$$1 \le n, m \le 10^3$$

$$1 \le k \le 10^9$$

Output

In a single line print a single real number — the maximum logo size.

Your answer will be counted if the relative or absolute error does not exceed 10^{-6}. Formally, if a — is your answer, and b — is the jury's answer, then it will be counted if $\frac{|a-b|}{max(b,1)} \le 10^{-6}$.

Example

standard input	standard output
2 2 3	1.0

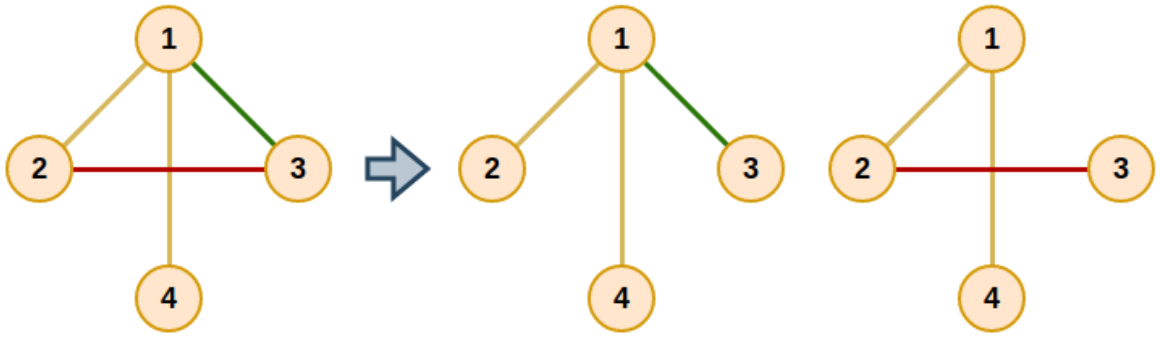
Problem B. Two trees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

Maxim Vitalievich decided to understand the graphs theory. He studied some theory regarding connected undirected graphs, but here Maksim Vitalievich had difficulties.

Among all types of connected undirected graphs, Maxim Vitalievich understood in detail only trees. Recall that a tree – is a connected graph that does not contain cycles and consists of n vertices and $n-1$ edges. Other kinds of graphs present some difficulties, as they may have cycles or they may not be connected.

To deal with more complex types of graphs, Maxim Vitalyevich decided to represent them as a union of a set of trees. For simplicity, he decided to limit himself to only two trees. In order to represent a graph of n vertices as a union of two trees, Maxim Vitalievich colored each edge in one of three colors. Next, he built the first n vertex tree using the 1 and 3 edges, and the second n vertex tree – using the 2 and 3 edges. Thus, if we combine two trees, we get the original graph, since each edge belongs to at least one tree.



The difficulties didn't end there. Maxim Vitalievich noticed that it is quite difficult to color some graphs so that the result is two trees with an initial set of vertices. Therefore, Maxim Vitalievich turned to you for help.

Help Maksim Vitalievich color m edges of a graph of n vertices with three colors so that it can be represented as two trees, each of which consists of n vertices.

Input

The first line contains two integers n and m — the number of vertices and edges in the original graph.

The next m lines contain two integers x_i and y_i — the numbers of the vertices connected by the i-th edge.

It is guaranteed that the original graph is connected and does not contain multiple edges and loops.

$$2 \le n \le 100$$

$$n - 1 \le m \le 200$$

$$1 \le x_i, y_i \le n$$

Output

On the first line print "`No`" (without quotes) if there is no answer.

Otherwise, on the first line print "`Yes`" (without quotes).

In the second line print m space-separated integers c_1, c_2, ... c_m, where c_i ($1 \le c_i \le 3$) — color of the i-th ribs.

If there are multiple answers, print any of them.

Examples

standard input	standard output
4 4 1 2 1 3 1 4 2 3	Yes 3 1 3 2
5 10 1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 4 5	No

Problem C. Sum of fractions

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Once Maxim Vitalievich decided to deal with ordinary fractions. After quite a long study of the theory, he came up with an entertaining problem. Given the following equation:

$$\frac{1}{a} + \frac{1}{b} = \frac{c}{k}$$

It is necessary to determine for a given integer k the number of triplets of positive integers a, b and c that are the solution of this equation such that $a \leq b$.

Maksim Vitalievich solved this problem for a given number k and now he is wondering at what k ($1 \leq k \leq r$) the number of solutions will be maximum.

Input

The single line contains a single integer r — the right bound for the number k.

$$1 \leq r \leq 10^9$$

Output

In a single line print two space-separated integers k and s ($1 \leq k \leq r$) — the value at which the largest number of solutions and the number of solutions for the number k are achieved.

If there are several solutions, then print the solution where k is the smallest.

Examples

standard input	standard output
3	3 7
5	4 10

Note

In the first example, for $k = 3$ there are the following solutions:

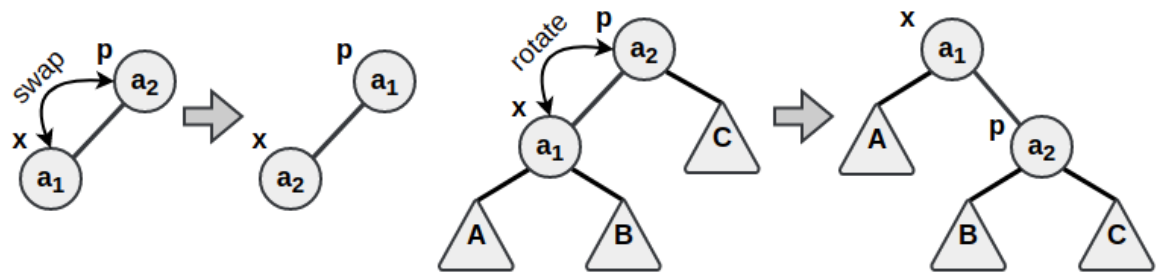
$(1, 1, 6)$, $(2, 2, 3)$, $(3, 3, 2)$, $(6, 6, 1)$, $(1, 3, 4)$, $(2, 6, 2)$, $(4, 12, 1)$.

Problem D. Binary tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

Maxim Vitalievich has a binary tree consisting of n vertices. All vertices are numbered from 1 to n. Each vertex of the tree contains some integer, and all numbers are different. He decided to turn it into a binary search tree using two kinds of operations:

- `swap` x — swap values at vertex x and parent vertex p.
- `rotate` x — rotate (either left or right) vertex x and parent vertex p.



The figure above shows an example of the "`swap`" operation and the "`rotate`" operation. Note that operations cannot be applied to the root of the tree because it has no parent vertex.

Help Maksim Vitalyevich turn the original binary tree into a **binary search tree in the minimum number of** "`swap`" operations. Recall that a binary tree is a binary search tree if and only if all values in the left subtree are strictly less than the value at the vertex, and all values in the right subtree are strictly greater than.

Input

The first line contains an integer n — the number of vertices in the binary tree.

The next n lines contain three integers a_i, x_i, y_i ($0 \le x_i, y_i \le n$) — the value at the i-th vertex, the numbers of the left and right vertices, respectively. If $x_i = 0$, then the i-th vertex has no left child vertex. If $y_i = 0$, then the i-th vertex has no right child vertex.

It is guaranteed that the description of the vertices in the input corresponds to the correct description of the binary tree and that all values of a_i are unique.

$$1 \le n \le 5\,000$$

$$1 \le a_i \le 10^9$$

Output

In the first line print the number m — the total number of operations.

In the next m lines print a description of the operations. The description of the i-th operation must be output in one of the following formats:

- `swap` x_i, where x_i — vertex number.
- `rotate` x_i, where x_i — vertex number.

If there are multiple answers, print any one. The total number of transactions must not exceed 300 000.

Examples

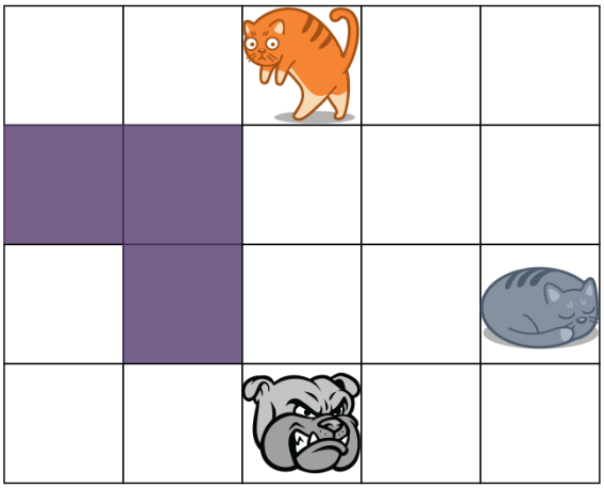
standard input	standard output
2 1 2 0 2 0 0	1 swap 2
3 1 2 3 3 0 0 2 0 0	3 swap 2 rotate 3 swap 1

Problem E. Kitten rescue

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Maxim Vitalyevich loves to play games, more precisely, one particular game. Maxim Vitalievich spent so much time in this game that he got tired of it lately, and he decided to take a break. To be more precise, Maxim Vitalievich decided to find another interesting game.

After a diligent search on the Internet, Maxim Vitalievich came across an exciting puzzle game. This game has several levels. The picture below shows an example level.



Each level is represented by a checkered field of size $n \times m$, where n is the number of rows and m is the number of columns. Each cell of this field is either free or occupied by an obstacle, a cat, a kitten or a dog. There is always exactly one cat and one kitten on the field, as well as no more than one dog. The player controls the movements of the cat. To pass the level, the player needs to save the kitten: in a certain number of moves, move the cat to the cell with the kitten. In one move, the player can move the cat to a horizontally or vertically adjacent cell, while he cannot move the cat to a cell with an obstacle. If after the move the cat ended up in a cell with a kitten, then the level is considered passed. If after the move the cat ended up in a cell with a dog, then the level is considered not passed (and it will have to be passed again).

After each player's move, if there is a dog on the field, it makes two moves. In one move, the dog approaches the cat by one square, either vertically or horizontally, while it cannot move into a cell with an obstacle. The Manhattan distance between the cat and the dog is used as the distance. If the dog can approach both horizontally and vertically, then it will always approach horizontally (moves either to the left or to the right). If the dog cannot get close, then it skips the move. Please note that the dog cannot move away from the cat. If the dog ends up in a cell with a cat after the move, the level is considered failed. The dog can move to the cell with the kitten, and if the player moves to the cell with the kitten and the dog, the level is also considered to be failed.

Maxim Vitalievich was able to pass several levels, but he got stuck at the next level. Help Maxim Vitalievich pass the level or tell him that this level cannot be passed.

Input

The first line contains two integers n and m — the number of rows and columns for the field, respectively.

The next n lines contain m characters each — description of the playing field from top to bottom, from left to right. Each character means:

- character "`.`" — free cell;
- character "`#`" — obstacle;
- character "`C`" — cat;
- character "`K`" — kitten;
- character "`@`" — dog.

$$1 \le n, m \le 60$$

Output

If the level cannot be completed, then print "`No`" in a single line (without the quotes).

Otherwise, on the first line print "`Yes`" (without quotes). In the second line print the moves for the cat. Each move is represented by one symbol:

- character "`L`" — move one cell to the left;
- character "`R`" — move one cell to the right;
- character "`U`" — move one cell up;
- character "`D`" — move one cell down.

The number of moves must not exceed 100 000. If there are several solutions, then print any one, not necessarily the shortest one.

Examples

standard input	standard output
4 6 ..C... ##.... .#...K ..@...	Yes LLRRRRRDD
1 6 K.@..C	No

Problem F. 4 positions

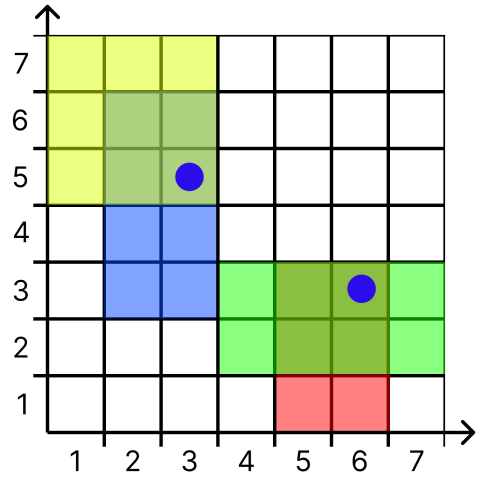
Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Martian colony is going through hard times. After a long sandstorm, all the elevators-entrances to underground shelters were covered with Martian sand. The colony is having difficulty supplying food to the shelters, so rescue modules have been dispatched to the colony.

The number of shelters is equal to k. Each i-th shelter can be represented on a flat map $10^9 \times 10^9$. Shelters can be located one under the other at different levels, respectively, on a flat map they will intersect. The locations of shelters on the plane are given by two coordinates, where the lower left vertex has the coordinates (l_i, d_i), and the upper right vertex (r_i, u_i). In the middle of each cell of the flat map is an elevator entrance that connects all the shelters below it, which can be cleared by the rescue module team.

There was very little time left, so the rescue module teams decided to clear only those entrance elevators that were necessary to quickly provide food to all Martian shelters.

You are the only one who did not skip work with contour maps at school, which means you can quickly inform the teams **the minimum number of those entrance elevators and their cell coordinates, by clearing which we will gain access to all shelters on Mars.** Teams have limited time and resources , so they can't clear all the entrance elevators. **It is also known that no more than m entrance elevators are needed to access all vaults.**



Picture for example 1

Input

The first line contains two integers k and m, where k — the number of Martian shelters, and m — the number of entrance elevators that rescue teams cannot clear more than.

The next k lines contain four space-separated integers l_i, d_i, r_i, u_i — coordinates for each hideout.

$$1 \le k \le 200\,000$$

$$1 \le m \le 4$$

$$1 \le l_i \le r_i \le 10^9$$

$$1 \le d_i \le u_i \le 10^9$$

Output

In the first line print a single integer n ($1 \le n \le m$), the minimum number of elevator exits needed to clear.

In the next n lines print two integers x_i and y_i — the coordinates of the cells (x_j, y_j), where the entrance elevators are located.

It is guaranteed that the answer always exists. If there are multiple answers, print any one.

Example

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

Problem G. Borrow checker

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Maxim Vitalievich decided to learn a new programming language. One of his acquaintances recommended learning the Rust language, as it is the most promising. Yes, the language is really inspiring, but it forces you to face the harsh reality — the great and mighty borrow checker.

To say that Maxim Vitalyevich was somewhat at a loss would be an understatement. To get out of this situation and deal with the borrow checker, Maxim Vitalievich decided to simplify the language.

The new language only supports a predefined set of functions. In total, n functions are available in the language, the description of each has the following syntax:

```
<char> := "a" | "b" | ... | "z"
<name> := <char> | <char> <name>
<nameOrRefName> := <name> | "&" <name>
<argument> := <nameOrRefName>
<arguments> := <argument> | <argument> ", " <arguments>
<functionName> := <nameOrRefName>
<functionArguments> := "" | <arguments>
<function> := "fn " <functionName> "(" <functionArguments> ")"
```

So <function> is a function description. You may notice that the function name can be either"name" or "&name". In the first case, the function returns a value; in the second case, a reference. Examples of correct functions:

```
fn input()          // The input function returns a value and does not contain any arguments.
fn sum(a, b)        // The sum function returns a value and takes two arguments by value.
fn add(&a, &b)      // The add function returns a value and takes arguments by reference.
fn &ref(a)          // The ref function returns a reference and takes an argument by value.
fn assign(a, &b)    // The assign function returns a value and takes the first argument
                    // by value, and the second by reference.
```

We also give incorrect examples (such examples cannot occur in the input data):

```
&fn funcd()       // Wrong character order.
fn funcb(a, b&)   // Wrong character order.
fn funcc(&&a)     // Cannot get reference to reference.
fn &&funca()      // Can't return a reference to a reference.
fn fn()           // Uses fn as the name.
fn funce(fn)      // Uses fn as the name.
fn funcf(a, a)    // Same argument names.
fn funcg(a, &a)   // Same argument names.
```

To implement some program, the language has the "<declare>" — construct to declare a new variable (or reference). The syntax can be described as follows:

```
<callArguments> := <expression> | <expression> ", " <callArguments>
<functionCallArguments> := "" | <callArguments>
<functionCall> := <name> "(" <functionCallArguments> ")"
<expression> := <nameOrRefName> | <functionCall>
<declare> := <name> " := " <expression>
```

Here are correct examples of using the construction:

```
a := input()       // Define a variable with the result value of input().
b := input()
c := a             // Assign the old value to the new variable.
d := add(&b, &c)   // Call the function, passing arguments by reference.
e := add(ref(b), ref(c))
f := &e
```

Here are incorrect examples of using the construction (such examples cannot be found in the input data):

```
a := &input()  // Can't get a reference to the result of a function.
fn := a        // Cannot use fn as a name.
&b := c        // Cannot declare a reference name.
```

Maxim Vitalievich believes that such a syntax is quite simple, so **he cannot make syntax errors when using the language**.

However, Maxim Vitalievich can make the following mistakes:

- Use a variable not previously declared with "`:=`".
- Use a non-existent function.
- Call a function with the wrong number of arguments (the number of arguments must match the definition).
- Pass a reference to a function argument that takes a value (or vice versa).
- **Use the previously moved value of a variable (or a reference to such a variable) in the expression**.

Let's take a closer look at the last point. Maksim Vitalyevich's language has its own borrow checker mechanism. The bottom line is that all variables that **are used by value are moved** and are no longer available. Consider the following example:

```
a := input()    // a is a value.
b := sum(a, a)  // Error!
```

In this case, the variable "`a`" is passed by value 2 times, but after the first use it is no longer available, thus an error occurs. In the same time:

```
a := input()       // a is a value.
b := sum(&a, &a)   // Correct if sum accepts references. Variable a
                   // remains available because it was not used by value.
```

After declaring a variable with the same name (`:=`), the old variable (if it is a value) **is considered to be moved** and is no longer available **from the next line**.

The following rules apply for links:

- Cannot get a reference for a moved variable.
- Cannot get a reference for a reference variable.
- You cannot use a reference to a moved variable.
- **You cannot pass a value variable and a reference to that variable at the same time on the same line.**

Consider the following examples:

```
a := input()
b := a
c := &a  // Error, variable a has been moved.

a := input()
b := &a
c := move(a)  // Same as c := a. In either case, the variable a is moved.
d := b        // Error, variable b is a reference to the value of a that has been moved.

a := input()
```

```
b := &a
c := call(a, b)  // Error. The variable a has been moved, so the reference cannot be used.

a := input()
b := &a
a := a           // The old value of variable a has been moved.
c := clone(b)    // Error, variable b refers to the old variable a, which
                 // has been moved to the previous line.
```

Since Maxim Vitalievich invented the language and wants to use it, he asks you to implement a program that will check Maxim Vitalievich's code for errors.

Input

The first line contains the number n — the number of functions.

The following n lines contain descriptions of functions according to the described syntax. It is guaranteed that all function names are unique, and all argument names for each function are also unique.

The next line contains the number m — the number of lines in Maxim Vitalyevich's code.

The following m lines contain constructs for declaring a new variable in accordance with the description of the syntax.

It is guaranteed that the total length of strings does not exceed 100 000. The input data does not contain extra spaces, and there cannot be spaces less than what is described in the syntax of the language.

$$1 \le n \le 20$$

$$1 \le m \le 100$$

Output

In a single line print "`Valid`" if the program is correct.

Otherwise, print "`Error in line N`" (without quotes), where `N` — is the number of the first line with an error in Maxim Vitalyevich's code.

Line numbering starts at 1.

Examples

standard input	standard output
3 fn input() fn clone(&a) fn sum(a, b) 5 a := input() b := input() br := &b s := sum(a, b) bc := clone(br)	Error in line 5
2 fn input() fn clone(&a) 3 a := input() a := &a b := clone(a)	Error in line 3
2 fn &input() fn sum(&a, &b) 3 a := input() b := sum(a, a) c := sum(a, &b)	Valid

Note

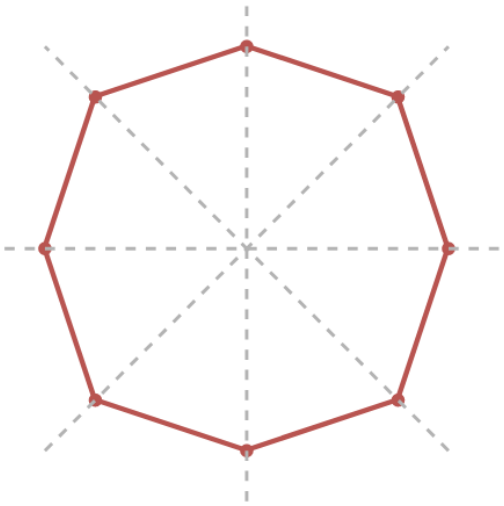
In the first test case, the fifth line uses the "`br`" reference to the value moved in the fourth line.

In the second test case, the value of the variable "`a`" is used on the third line, but it has already been moved on the previous line.

Problem H. Birthday

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Maxim Vitalievich was invited to a birthday party. He chose, in his opinion, an excellent gift — a cake in the form of a regular convex n-gon.



The holiday is in full swing, Maxim Vitalievich has already congratulated the birthday man and received a responsible task from him — it is necessary to cut the cake into pieces, not necessarily of equal size. The main condition is that the pieces are triangles.

Maxim Vitalyevich asked himself the question, what is the minimum number of cuts needed to divide the cake into triangles. It is allowed to cut the cake only along diagonals (straight lines passing through a pair of n-gon vertices).

Input

The single line contains an integer n — the number of vertices in the polygon.

$$4 \le n \le 100$$

Output

Print a single number — the minimum number of cuts required to divide the cake into triangular pieces.

Examples

standard input	standard output
5	2
8	4

Problem I. Palindrome tree

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

You are given a tree consisting of n vertices. Each vertex contains some symbol c_i. You need to remove a certain number of vertices so that the remaining set of vertices is connected and it is impossible to find a palindromic path of length greater than k. A simple path is called palindromic if writing out the characters written at the vertices in forward and reverse order results in the same string.

Input

The first line contains two integers n and k — the number of vertices in the tree and the maximum length of a palindromic path.

The second line contains a string s consisting only of lowercase English letters of length n, where the i-th character c_i corresponds to the character written at the i-th vertex.

The next $(n-1)$ lines contain two integers x_i and y_i ($x_i \neq y_i$, $1 \leq x_i, y_i \leq n$) — numbers of vertices connected by an edge.

$$1 \leq k \leq n \leq 2\,000$$

Output

In the first line print the number m — the number of vertices remaining after the removal.

In the second line print m integers b_i — the numbers of the remaining vertices.

If there is more than one answer, then print the minimum answer that is not a prefix of another answer.

The answer a is less than b if and only if there exists p such that $a_p < b_p$ and $a_j = b_j$ for $1 \leq j < p$.

The answer a is a prefix of b if and only if $|a| < |b|$ and $a_j = b_j$ for $1 \leq j \leq |a|$, where $|a|$ is the length of the array a.

Examples

standard input	standard output
3 2 aba 1 2 1 3	3 1 2 3
3 2 aba 1 2 2 3	2 1 2

Note

In the second example, several answers are suitable: $(1, 2)$, $(2, 1)$, $(2, 3)$, $(3, 2)$, (1), (2) , (3). The answers (1), (2), (3) are the prefixes of the answers $(1, 2)$, $(2, 1)$ and $(3, 2)$ respectively. Among the remaining suitable answers, the minimum is $(1, 2)$.

Problem J. Interdimensional Traveler

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

Maxim Vitalievich wanted to go on a trip. He must have gotten bored. In any case, for his new journey, Maxim Vitalyevich decided to use a ship for interdimensional travel, with which you can travel to n-dimensional space.

After some time of his journey, Maxim Vitalievich finds himself in a rather precarious position. In the literal sense, his ship begins to wobble violently from side to side. Maxim Vitalievich immediately realized that he had fallen into quantum turbulence.

Quantum turbulence can be described as follows: turbulence in the i-th dimension acts for the coordinates $x_i \geq 1$. If the ship is in quantum turbulence for the i-th dimension, then it becomes uncontrollable in this dimension. Instead, the ship moves from coordinate x_i to coordinate $x_i + 1$ with probability p and to coordinate $x_i - 1$ with probability $q = 1 - p$. If the ship enters the coordinate $x_i < 1$, it becomes controllable again and stops randomly moving in the i-th dimension.

You need to determine the probability ans with which Maxim Vitalievich will be able to completely restore control over all n dimensions, if initially he is in the coordinate $(a_1, a_2, ..., a_n)$.

Input

The first line contains the number n — the number of measurements.

The second line contains n integers a_1, a_2, ..., a_n — the initial coordinates of Maxim Vitalyevich's ship.

The next n lines contain two integers s_i, t_i — the value of the probability p_i, represented as a fraction $\frac{s_i}{t_i}$.

$$1 \leq n \leq 1\,000$$

$$1 \leq a_i \leq 1\,000$$

$$0 \leq s_i \leq 10^9$$

$$1 \leq t_i \leq 10^9$$

Output

It is guaranteed that the answer ans can be represented as an irreducible fraction $\frac{s}{t}$.

In a single line print the probability that Maxim Vitalievich will be able to restore the control represented as $s \cdot t^{-1} \mod 10^9 + 7$.

Example

standard input	standard output
3 1 2 3 4 6 8 10 5 6	714250005