A. Fibonotci

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fibonotci sequence is an integer recursive sequence defined by the recurrence relation

$$F_n = s_{n-1} \cdot F_{n-1} + s_{n-2} \cdot F_{n-2}$$

with

$$F_0 = 0, F_1 = 1$$

Sequence s is an infinite and almost cyclic sequence with a cycle of length N. A sequence s is called almost cyclic with a cycle of length N if $s_i = s_{i \bmod N}$, for $i \geq N$, except for a finite number of values s_i, for which $s_i \neq s_{i \bmod N}$ ($i \geq N$).

Following is an example of an almost cyclic sequence with a cycle of length 4:

s = (5,3,8,11,5,3,7,11,5,3,8,11,…)

Notice that the only value of s for which the equality $s_i = s_{i \bmod 4}$ does not hold is s_6 ($s_6 = 7$ and $s_2 = 8$). You are given $s_0, s_1, \ldots s_{N-1}$ and all the values of sequence s for which $s_i \neq s_{i \bmod N}$ ($i \geq N$).

Find $F_K \bmod P$.

Input

The first line contains two numbers K and P. The second line contains a single number N. The third line contains N numbers separated by spaces, that represent the first N numbers of the sequence s. The fourth line contains a single number M, the number of values of sequence s for which $s_i \neq s_{i \bmod N}$. Each of the following M lines contains two numbers j and v, indicating that $s_j \neq s_{j \bmod N}$ and $s_j = v$. All j-s are distinct.

- $1 \leq N, M \leq 50000$
- $0 \leq K \leq 10^{18}$
- $1 \leq P \leq 10^9$
- $1 \leq s_i \leq 10^9$, for all $i = 0, 1, \ldots N - 1$
- $N \leq j \leq 10^{18}$
- $1 \leq v \leq 10^9$
- All values are integers

Output

Output should contain a single integer equal to $F_K \bmod P$.

Examples

input

```
10 8
3
1 2 1
2
7 3
5 4
```

output

```
4
```

B. Bribes

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ruritania is a country with a very badly maintained road network, which is not exactly good news for lorry drivers that constantly have to do deliveries. In fact, when roads are maintained, they become **one-way**. It turns out that it is sometimes impossible to get from one town to another in a legal way – however, we know that all towns are **reachable**, though **illegally**!

Fortunately for us, the police tend to be very corrupt and they will allow a lorry driver to break the rules and drive in the wrong direction provided they receive 'a small gift'. There is one patrol car for every road and they will request 1000 Ruritanian dinars when a driver drives in the wrong direction. However, being greedy, every time a patrol car notices the same driver breaking the rule, they will charge **double** the amount of money they requested the previous time on that particular road.

Borna is a lorry driver that managed to figure out this bribing pattern. As part of his job, he has to make K stops in some towns all over Ruritania and he has to make these stops in a certain order. There are N towns (enumerated from 1 to N) in Ruritania and Borna's initial location is the capital city i.e. town 1. He happens to know which ones out of the $N - 1$ roads in Ruritania are currently unidirectional, but he is unable to compute the least amount of money he needs to prepare for bribing the police. Help Borna by providing him with an answer and you will be richly rewarded.

Input

The first line contains N, the number of towns in Ruritania. The following $N - 1$ lines contain information regarding individual roads between towns. A road is represented by a tuple of integers (a,b,x), which are separated with a single whitespace character. The numbers a and b represent the cities connected by this particular road, and x is either 0 or 1: 0 means that the road is bidirectional, 1 means that only the $a \to b$ direction is legal. The next line contains K, the number of stops Borna has to make. The final line of input contains K positive integers $s_1, \ldots, s_K$: the towns Borna has to visit.

- $1 \le N \le 10^5$
- $1 \le K \le 10^6$
- $1 \le a, b \le N$ for all roads
- $x \in 0, 1$ for all roads
- $1 \le s_i \le N$ for all $1 \le i \le K$

Output

The output should contain a single number: the least amount of thousands of Ruritanian dinars Borna should allocate for bribes, modulo $10^9 + 7$.

Examples

input

```
5
1 2 0
2 3 0
5 1 1
3 4 1
5
5 4 5 2 2
```

output

```
4
```

Note

Borna first takes the route $1 \to 5$ and has to pay 1000 dinars. After that, he takes the route $5 \to 1 \to 2 \to 3 \to 4$ and pays nothing this time. However, when he has to return via $4 \to 3 \to 2 \to 1 \to 5$, he needs to prepare 3000 (1000+2000) dinars. Afterwards, getting to 2 via $5 \to 1 \to 2$ will cost him nothing. Finally, he doesn't even have to leave town 2 to get to 2, so there is no need to prepare any additional bribe money. Hence he has to prepare 4000 dinars in total.

C. Party

time limit per test: 2 seconds
memory limit per test: 4 megabytes
input: standard input
output: standard output

Note the unusual memory limit for the problem.

People working in MDCS (Microsoft Development Center Serbia) like partying. They usually go to night clubs on Friday and Saturday.

There are N people working in MDCS and there are N clubs in the city. Unfortunately, if there is more than one Microsoft employee in night club, level of coolness goes infinitely high and party is over, so club owners will never let more than one Microsoft employee enter their club in the same week (just to be sure).

You are organizing night life for Microsoft employees and you have statistics about how much every employee likes Friday and Saturday parties for all clubs.

You need to match people with clubs maximizing overall sum of their happiness (they are happy as much as they like the club), while **half** of people should go clubbing on Friday and the other **half** on Saturday.

Input

The first line contains integer N — number of employees in MDCS.

Then an $N \times N$ matrix follows, where element in i-th row and j-th column is an integer number that represents how much i-th person likes j-th club's **Friday** party.

Then another $N \times N$ matrix follows, where element in i-th row and j-th column is an integer number that represents how much i-th person likes j-th club's **Saturday** party.

- $2 \leq N \leq 20$
- N is even
- $0 \leq$ level of likeness $\leq 10^6$
- All values are integers

Output

Output should contain a single integer — maximum sum of happiness possible.

Examples

input

```
4
1 2 3 4
2 3 4 1
3 4 1 2
4 1 2 3
5 8 7 1
6 9 81 3
55 78 1 6
1 1 1 1
```

output

```
167
```

Note

Here is how we matched people with clubs:

Friday: 1st person with 4th club (4 happiness) and 4th person with 1st club (4 happiness).

Saturday: 2nd person with 3rd club (81 happiness) and 3rd person with 2nd club (78 happiness).

4+4+81+78 = 167

D. Tablecity

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There was a big bank robbery in Tablecity. In order to catch the thief, the President called none other than Albert – Tablecity's Chief of Police. Albert does not know where the thief is located, but he does know how he moves.

Tablecity can be represented as 1000×2 grid, where every cell represents one district. Each district has its own unique name "(X, Y)", where X and Y are the coordinates of the district in the grid. The thief's movement is as

Every hour the thief will leave the district (X, Y) he is currently hiding in, and move to one of the districts: $(X - 1, Y)$, $(X + 1, Y)$, $(X - 1, Y - 1)$, $(X - 1, Y + 1)$, $(X + 1, Y - 1)$, $(X + 1, Y + 1)$ as long as it exists in Tablecity.

Below is an example of thief's possible movements if he is located in district (7,1):

Albert has enough people so that **every hour** he can pick any **two** districts in Tablecity and fully investigate them, making sure that if the thief is located in one of them, he will get caught. Albert promised the President that the thief will be caught in **no more** than 2015 hours and needs your help in order to achieve that.

Input

There is no input for this problem.

Output

The first line of output contains integer N – duration of police search in hours. Each of the following N lines contains exactly 4 integers X_{i1}, Y_{i1}, X_{i2}, Y_{i2} separated by spaces, that represent 2 districts (X_{i1}, Y_{i1}), (X_{i2}, Y_{i2}) which got investigated during i-th hour. Output is given in **chronological** order (i-th line contains districts investigated during i-th hour) and should **guarantee** that the thief is caught in no more than 2015 hours, **regardless of thief's initial position and movement**.

- $N \le 2015$
- $1 \le X \le 1000$
- $1 \le Y \le 2$

Examples

input

```
В этой задаче нет примеров ввода-вывода.
This problem doesn't have sample input and output.
```

output

```
Смотрите замечание ниже.
See the note below.
```

Note

Let's consider the following output:

2

5 1 50 2

8 1 80 2

This output is not guaranteed to catch the thief and is not correct. It is given to you only to show the expected output format. There exists a combination of an initial position and a movement strategy such that the police will not catch the thief.

Consider the following initial position and thief's movement:

In the first hour, the thief is located in district (1,1). Police officers will search districts (5,1) and (50,2) and will not find him.

At the start of the second hour, the thief moves to district (2,2). Police officers will search districts (8,1) and (80,2) and will not find him.

Since there is no further investigation by the police, the thief escaped!

E. Spectator Riots

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

It’s riot time on football stadium Ramacana! Raging fans have entered the field and the police find themselves in a difficult situation. The field can be represented as a square in the coordinate system defined by two diagonal vertices in (0,0) and $(10^5, 10^5)$. The sides of that square are also considered to be **inside** the field, everything else is **outside**.

In the beginning, there are N fans on the field. For each fan we are given his speed, an integer v_i as well as his integer coordinates (x_i, y_i). A fan with those coordinates might move and after one second he might be at any point $(x_i + p, y_i + q)$ where $0 \le |p| + |q| \le v_i$. p, q are both integers.

Points that go **outside** of the square that represents the field are excluded and all others have equal probability of being the location of that specific fan after one second.

Andrej, a young and promising police officer, has sent a flying drone to take a photo of the riot from above. The drone’s camera works like this:

1. It selects three points with **integer coordinates** such that there is a chance of a fan appearing there after one second. They must not be collinear or the camera won’t work. It is guaranteed that not all of the initial positions of fans will be on the same line.
2. Camera focuses those points and creates a circle that passes through those three points. A photo is taken after one second (one second after the initial state).
3. Everything that is on the circle or inside it at the moment of taking the photo (one second after focusing the points) will be on the photo.

Your goal is to select those three points so that the expected number of fans seen on the photo is maximized. If there are more such selections, select those three points that give the circle with **largest radius** among them. If there are still more suitable selections, **any one** of them will be accepted. If your answer follows conditions above and radius of circle you return is smaller then the optimal one by 0.01, your output will be considered as correct.

No test will have optimal radius bigger than 10^{10}.

Input

The first line contains the number of fans on the field, N. The next N lines contain three integers: x_i ,y_i, v_i. They are the x-coordinate, y-coordinate and speed of fan i at the beginning of the one second interval considered in the task.

- $3 \le N \le 10^5$
- $0 \le x_i, y_i \le 10^5$
- $0 \le v_i \le 1000$
- All numbers are integers

Output

You need to output the three points that camera needs to select. Print them in three lines, with every line containing the x-coordinate, then y-coordinate, separated by a single space. The order of points does not matter.

Examples

input

```
3
1 1 1
1 1 1
1 2 1
```

output

```
2 2
2 1
1 0
```

F. Bulbo

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bananistan is a beautiful banana republic. Beautiful women in beautiful dresses. Beautiful statues of beautiful warlords. Beautiful stars in beautiful nights.

In Bananistan people play this crazy game – Bulbo. There’s an array of bulbs and player at the position, which represents one of the bulbs. The distance between two neighboring bulbs is 1. Before each turn player can change his position with cost $|pos_{new} - pos_{old}|$. After that, a contiguous set of bulbs lights-up and player pays the cost that’s equal to the distance to the closest shining bulb. Then, all bulbs go dark again. The goal is to minimize your summed cost. I tell you, Bananistanians are spending their nights playing with bulbs.

Banana day is approaching, and you are hired to play the most beautiful Bulbo game ever. A huge array of bulbs is installed, and you know your initial position and all the light-ups in advance. You need to play the ideal game and impress Bananistanians, and their families.

Input

The first line contains number of turns n and initial position x. Next n lines contain two numbers l_{start} and l_{end}, which represent that all bulbs from interval $[l_{start}, l_{end}]$ are shining this turn.

- $1 \leq n \leq 5000$
- $1 \leq x \leq 10^9$
- $1 \leq l_{start} \leq l_{end} \leq 10^9$

Output

Output should contain a single number which represents the best result (minimum cost) that could be obtained by playing this Bulbo game.

Examples

input

```
5 4
2 7
9 16
8 10
9 17
1 6
```

output

```
8
```

Note

Before 1. turn move to position 5

Before 2. turn move to position 9

Before 5. turn move to position 8

G. Run for beer

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

People in BubbleLand like to drink beer. Little do you know, beer here is so good and strong that every time you drink it your speed goes 10 times slower than before you drank it.

Birko lives in city Beergrade, but wants to go to city Beerburg. You are given a road map of BubbleLand and you need to find the fastest way for him. When he starts his journey in Beergrade his speed is 1. When he comes to a new city he always tries a glass of local beer, which divides his speed by 10.

The question here is what the minimal time for him to reach Beerburg is. If there are several paths with the same minimal time, pick the one that has least roads on it. If there is still more than one path, pick any.

It is guaranteed that there will be at least one path from Beergrade to Beerburg.

Input

The first line of input contains integer N — the number of cities in Bubbleland and integer M — the number of roads in this country. Cities are enumerated from 0 to $N - 1$, with city 0 being Beergrade, and city $N - 1$ being Beerburg. Each of the following M lines contains three integers a, b ($a \neq b$) and len. These numbers indicate that there is a bidirectional road between cities a and b with length len.

- $2 \le N \le 10^5$
- $1 \le M \le 10^5$
- $0 \le len \le 9$
- There is at most one road between two cities

Output

The first line of output should contain minimal time needed to go from Beergrade to Beerburg.

The second line of the output should contain the number of cities on the path from Beergrade to Beerburg that takes minimal time.

The third line of output should contain the numbers of cities on this path in the order they are visited, separated by spaces.

Examples

input

```
8 10
0 1 1
1 2 5
2 7 6
0 3 2
3 7 3
0 4 0
4 5 0
5 7 2
0 6 0
6 7 7
```

output

```
32
3
0 3 7
```

H. Bots

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Sasha and Ira are two best friends. But they aren't just friends, they are software engineers and experts in artificial intelligence. They are developing an algorithm for two bots playing a two-player game. The game is cooperative and turn based. In each turn, one of the players makes a move (it doesn't matter which player, it's possible that players turns do not alternate).

Algorithm for bots that Sasha and Ira are developing works by keeping track of the state the game is in. Each time either bot makes a move, the state changes. And, since the game is very dynamic, it will never go back to the state it was already in at any point in the past.

Sasha and Ira are perfectionists and want their algorithm to have an optimal winning strategy. They have noticed that in the optimal winning strategy, both bots make exactly N moves each. But, in order to find the optimal strategy, their algorithm needs to analyze all possible states of the game (they haven't learned about alpha-beta pruning yet) and pick the best sequence of moves.

They are worried about the efficiency of their algorithm and are wondering what is the total number of states of the game that need to be analyzed?

Input

The first and only line contains integer N.

- $1 \le N \le 10^6$

Output

Output should contain a single integer – number of possible states modulo $10^9 + 7$.

Examples

input
2

output
19

Note

Start: Game is in state A.

- Turn 1: Either bot can make a move (first bot is red and second bot is blue), so there are two possible states after the first turn – B and C.
- Turn 2: In both states B and C, either bot can again make a turn, so the list of possible states is expanded to include D, E, F and G.
- Turn 3: Red bot already did N=2 moves when in state D, so it cannot make any more moves there. It can make moves when in state E, F and G, so states I, K and M are added to the list. Similarly, blue bot cannot make a move when in state G, but can when in D, E and F, so states H, J and L are added.
- Turn 4: Red bot already did N=2 moves when in states H, I and K, so it can only make moves when in J, L and M, so states P, R and S are added. Blue bot cannot make a move when in states J, L and M, but only when in H, I and K, so states N, O and Q are added.

Overall, there are 19 possible states of the game their algorithm needs to analyze.

I. Robots protection

time limit per test: 1.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Company "Robots industries" produces robots for territory protection. Robots protect triangle territories — right isosceles triangles with catheti parallel to North-South and East-West directions.

Owner of some land buys and sets robots on his territory to protect it. From time to time, businessmen want to build offices on that land and want to know how many robots will guard it. You are to handle these queries.

Input

The first line contains integer N — width and height of the land, and integer Q — number of queries to handle.

Next Q lines contain queries you need to process.

Two types of queries:

1. $1\ dir\ x\ y\ len$ — add a robot to protect a triangle. Depending on the value of dir, the values of x, y and len represent a different triangle:
   - $dir = 1$: Triangle is defined by the points (x, y), $(x + len, y)$, $(x, y + len)$
   - $dir = 2$: Triangle is defined by the points (x, y), $(x + len, y)$, $(x, y - len)$
   - $dir = 3$: Triangle is defined by the points (x, y), $(x - len, y)$, $(x, y + len)$
   - $dir = 4$: Triangle is defined by the points (x, y), $(x - len, y)$, $(x, y - len)$
2. $2\ x\ y$ — output how many robots guard this point (robot guards a point if the point is inside or on the border of its triangle)

- $1 \le N \le 5000$
- $1 \le Q \le 10^5$
- $1 \le dir \le 4$
- All points of triangles are within range $[1, N]$
- All numbers are positive integers

Output

For each second type query output how many robots guard this point. Each answer should be in a separate line.

Examples

input

```
17 10
1 1 3 2 4
1 3 10 3 7
1 2 6 8 2
1 3 9 4 2
2 4 4
1 4 15 10 6
2 7 7
2 9 4
2 12 2
2 13 8
```

output

```
2
2
2
0
1
```