

## Problem A. A day in the woods

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

People from the city really like to spend their weekends at the forester's range. A bus would take them to the foresters hut and theyd hike to forest clearings nearby to run barbecue parthes. But sometimes it happened so that they lost their way and roamed from one clearing to another, taking different trails.

Alex, the head forester, decided to help the silly citizens. He put numbers on all clearings and put arrows on trails so that a citizen walking from the forester's hut along the arrows, passing on to other clearings no more than once, would find himself on clearings with increasing numbers. If this citizen decided that he wants to return, hed have to go against the direction of the arrows. The decreasing numbers of the clearings would mean he is going in the right direction. After renumbering all the clearings should have different numbers from 1 to  $n$ .

Alex has a map of his forest where all the clearings and the trails linking the clearings are marked. Any two clearings can be linked with one or more trails. The forester put the arrows correctly, but now he is having trouble with the clearing numbers. Help him to forester put numbers on the clearings according to his plan.

### Input

The input contains several testcases.

The first line of each test set contains three integers  $n$ ,  $m$ , and  $s$ , — the number of clearings, the number of trails, and the number of the clearing where the forester's hut is located, respectively ( $1 \leq n \leq 1000$ ,  $0 \leq m \leq 1000$ ,  $1 \leq s \leq n$ ). All clearings are enumerated with integers from 1 to  $n$ .

The next  $m$  lines describe the trails — each of them contains two integers, numbers of the clearings a trail connects in the direction of the arrow put by the forester. Note that loop trails are possible, i.e. in some cases these numbers can be equal.

The sum of  $n$  in all tests is not greater than 1000. The sum of  $m$  in all tests is not greater than 1000. The input ends with the line with three zeroes, which shouldn't be processed.

### Output

For each test case print the numbering of the clearings in a separate line. If no acceptable numbering exists, the output must contain the phrase "No solution".

### Example

standard input	standard output
3 3 1	1 2 3
1 2	2 3 4 1
2 3	No solution
1 3	
4 4 4	
4 1	
1 2	
2 3	
3 1	
4 6 1	
1 2	
1 3	
1 4	
2 3	
3 4	
4 2	
0 0 0	

## Problem B. Bus Stops

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

A core part of the decision of where to live in a city like Bytesburg is the availability of transport links to interesting places. This is particularly interesting to Barbara, who enlivens her stressful life as an organiser of programming camps by making frequent sightseeing travels around town in a yellow-blue bus.

Barbara's idea of a good time is a visit to a spot that takes exactly one bus journey to get to. She is considering moving her house to be near one specific spot along her favourite bus route—how many other scenic spots can she reach from there (assuming that on a given trip she can choose a new bus route each time)?

### Input

- The first line of input contains three integers: Barbara's starting stop,  $m$  ( $1 \leq m \leq s$ ), the number of buses,  $b$  ( $1 \leq b \leq 50$ ), and the number of stops,  $s$  ( $1 \leq s \leq 50$ ).
- The next  $b$  lines contain the bus routes, each written as a string of  $s$  characters where having the  $i$ th character as '1' denotes that this bus route has a stop at  $i$ , and '0' denotes that it does not.

### Output

Output the maximum number of other stops Barbara can reach from the starting stop by taking exactly one bus.

### Example

standard input	standard output
1 3 5 01100 10011 10111	3
2 2 3 101 101	0

## Problem C. Campus for SIT

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Carol is the urbanist, and now she is working on the project of the new campus for Schlaffhausen Institute of Technology (SIT). Territory of the campus can be represented as  $n$  by  $m$  grid, with beautiful lakes in some cells and forest in another. Carol wants the campus buildings to fulfill the next conditions:

1. Buildings shall be put to forest cells only, no more than one building per forest cell.
2. To make campus compact enough, each forest cell must have a building directly on it, or adjacent to it.
3. For the environmental purposes no two buildings can be adjacent to each other.

Two cells are adjacent if they share a side.

Find any placement of buildings that satisfies these constraints.

### Input

The first line of the input consists of two integers  $n$  and  $m$  ( $1 \leq n, m \leq 100$ ). The following  $n$  lines each contain a string of length  $m$  consisting only of the characters 'F' (forest) and 'L' (lake). This is the map of the land for the campus. It is guaranteed that the map contains at least one forest cell.

### Output

Output a copy of the map, where some of the land cells have been replaced with the letter 'B', meaning that a building was placed on the corresponding land cell. This placement should satisfy the constraints above. If there are many solutions, any one will be accepted.

### Examples

standard input	standard output
5 6 FFFFLF FFFFLF LFFFFF FFFFFF LLFFFL	BFFBLF FFBFLB LFFBFF FBFFFB LLBFBL
10 16 LLLLLLLFLFFFLL LLLLLFFFFFLLLL LLLLLFFFFFLLLL LLLLLFFFFFLLLL LLLFFFFFLLLL LLFFFFFLLLL LLFFFFFLLLL LLFFFFFLLLL LLFFFFFLLLL LLFFFLLLL LLFFFLLLL LLFFFLLLL	LLLLLLLBLFBFBL LLLLLBFFFBFL LLLLLBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL LLBFFBFBFL

## Problem D. Data Consistency

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Diana is a system administrator in the datacenter of the Byteland Astronomical Institute. The datacenter is planning to buy the special workstation for loseless real-time data communication with the satellites. Each satellite sends block of  $D$  mebibytes of the data to the Earth per session. All this data must be kept.

Workstation supports the SSD modules of  $S$  mebibytes each. Modules can be installed in multiple layers, one module per layer. When module is full, it must be immediately replaced with the empty one, but replacement takes some time and cause data loss, so no SSD module can be replaced until transaction of current block is finished.

To prevent data loss, Diana use very recent solution from Acronis called Acronis Infinite Space. It works in the next way. Each new block of the data is written to SSD in the first layer. When the SSD runs out of the free space and current block is still not received, last part of the block is written on the SSD in the second layer; if this SSD runs out of the free space while receiving this part, remaining part goes to the SSD in third layer and so on. When the block is received, all full SSD's are moved for the further processing and replaced with empty ones.

To perform the initial setup of the Acronis Infinite Space Diana wants to know minimal number of layers  $L$  needed to prevent the data loss.

### Input

The input consists of a single line containing the two integers  $S$  and  $D$  ( $1 \leq D \leq S \leq 10^{10}$ ).

### Output

Output the smallest integer  $L$  such that  $L$  layers will be enough to prevent the data loss. If it is impossible to prevent the data loss with any number of layers, print  $-1$  instead.

### Example

standard input	standard output
31 6	4

### Note

In the sample, at each 6'th session the first layer SSD is full and  $6 \cdot 6 - 31 = 5$  mebibytes go to second layer SSD. At each  $6 \cdot 7$  session the second layer SSD is full, and  $7 \cdot 5 - 31 = 4$  mebibytes go to third layer SSD. At each  $6 \cdot 7 \cdot 8$ 'th session the third layer SSD is full, and  $8 \cdot 4 - 1 = 1$  mebibyte goes to fourth layer SSD. After each  $6 \cdot 7 \cdot 8 \cdot 31$  time the fourth SSD is full right at the end of transaction, and no fifth SSD is needed.

## Problem E. Evenly Divided

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 second  
Memory limit: 512 mebibytes

The Byteland ICPC Community oversaw a sharp resurgence in membership this year, and must now face the inevitable strains of growth: the group photo they usually take can no longer fit everyone in one long row.

Edward, the photographer, asks members to split into two groups: Tall and Short, so that the picture can be doubled up with taller people standing behind shorter people in two rows of  $\frac{n}{2}$  each.

The ICPC Community annual meetings is an opportunity for the members to meet people. Many new joiners were assigned a mentor from the members who had already signed up before they joined. And Edward wants to choose a way of arranging the rows such that nobody is standing directly in front of or behind their mentor, assuming they have one.

Find a way of arranging the two rows such that this is possible. The number of tall people is always the same as the number of short people.

### Input

The input consists of:

- a line consisting of the number of members in the mountaineering society, which is a positive even integer  $m$  ( $1 \leq m \leq 10^5$ ).
- $m$  further lines, with the  $i$ th line ( $1 \leq i \leq m$ ) consisting of an integer indicating whether the  $i$ th member is short (0) or tall (1), then the number of the  $i$ th member's mentor,  $t_i$  ( $0 \leq t \leq m$ ). When  $t_i = i$ , this indicates that the  $i$ th member did not have a mentor.

### Output

If an arrangement is possible, output 2 lines of  $\frac{n}{2}$  numbers each to show which member should stand where.

Every number of type 1 should occur somewhere on the first row, and every number of type 0 should occur somewhere on the second row. Nobody should share a column with their mentor.

Otherwise, output impossible.

### Examples

standard input	standard output
4 0 1 1 1 1 2 0 3	3 2 1 4
4 0 1 1 1 0 1 1 1	impossible
10 0 1 1 1 1 1 1 1 0 1 0 4 0 6 1 1 0 7 1 2	10 8 3 2 4 1 6 7 9 5

## Problem F. Formula-1

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Those of you, who watch Formula-1 races, know that for several seasons Acronis supports one of teams: they started with Toro Rosso and now they are IT partners of Williams team.

Traditionally, the information for the drivers is shown by teams using special boards with big letters on it. Usually its time gap to the next (or previous) driver. Sure, this information is sent to the onboard electronics by the radio channel... but sometimes radio does not work properly.

Farid, one of the engineers of Williams come to Acronis with brilliant idea: let in this cases the video recognition software does all the work and reads the data from the tables.

Important part of this module is the recognition of the single letters. Your task is to implement it.

### Input

First line of the input contains one integer  $n$  ( $1 \leq n \leq 100$ ) — number of the possible letters.

Then the  $n$  letter descriptions come. Description of  $i$ -th letter starts with the two integers  $h_i$  and  $w_i$  ( $1 \leq h_i, w_i \leq 200$ ) — numbers of rows and columns in the letter shape. Each of the next  $h_i$  lines contains exactly  $w_i$  characters. ‘.’ means that pixel is white, while ‘X’ means that pixel is black. It is guaranteed that each letter is 4-connected, i.e. between any two ‘X’ pixels exists a path consisting entirely from ‘X’ pixels where two consecutive pixels share a side, and that any two letters are pairwise distinct and cannot be transformed one to another by a rotation of multiple of 90 degrees.

Then come description of the picture of the board with message. The first line of the description contains two integers  $h$  and  $w$  — number of rows and columns of the picture. Each of the next  $h$  lines contains exactly  $w$  characters ‘.’ and ‘X’. You may assume that:

- Each ‘X’ pixel in the messages are parts of exactly one letter.
- Letters does not overlap neither touch by sides (but they may touch diagonally).
- All the letters in the message are the letters described in the previous section; they may be rotated by 0, 90, 180 or 270 degrees.

### Output

For each letter in order they are listed in the input data, print number of its occurrences in the message.

## Examples

standard input	standard output
<pre> 2 5 8 ....XXXX ...XX.XX ..XX..XX .XXXXXXX XX...XX 5 6 XXXXXX ..XX.. ..XX.. ..XX.. ..XX.. 11 13 .....X.... .....X.... .....XXXXX .....XXXXX .....X.... .....X.... ....XXXX.... ...XX.XX.... ..XX..XX.... .XXXXXX.... XX...XX.... </pre>	<pre> 1 1 </pre>
<pre> 2 3 1 X X X 5 7 .XXXXX. XX...XX X....X XX...XX .XXXXX. 5 7 .XXXXX. XX...XX X.XXX.X XX...XX .XXXXX. </pre>	<pre> 1 1 </pre>

## Problem G. Generator

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Gregor is an engineer. He now is working on the project of new generator for water power plants. One of important parts of generators is a water wheel.

A water wheel is a device for converting the potential energy of water into the kinetic energy of rotation. We have attached buckets along our otherwise perfectly-balanced wheel. Each bucket has a specific weight when empty, and a specific weight when full of water, both in kilograms.

Whenever a bucket reaches the top of the water wheel, it is filled to its maximum capacity. When the bucket reaches the bottom, diametrically opposite, it is emptied again.

The acceleration of the wheel depends on the horizontal distance between the centre of mass of the wheel and its axis of rotation. Centre of mass of the wheel at some time is defined as the sum of the co-ordinates of the buckets at that time multiplied by their weight.

Our water wheel has a radius of exactly 1 metre.

To calculate effectivity of his construction Gregor wants to know maximum positive  $x$ -component of centre of mass achieved by the wheel at any angle. Help him to find it.

### Input

- The first line of input contains the number of buckets on the wheel,  $n$  ( $2 \leq n \leq 10^5$ ).
- The remaining  $n$  lines of input each contain a description of a bucket, in angular order:
  - the decimal clockwise angle of the  $i$ th bucket in degrees,  $d_i$  ( $0.0 \leq d \leq 360.0$ ),
  - the decimal weights of this bucket when empty and full,  $e_i$  and  $f_i$  ( $0 \leq e_i \leq f_i \leq 10^4$ ).

### Output

Output the maximum horizontal component of the centre of mass of the wheel over all possible angles. Your answer should be within an absolute or relative error of at most  $10^{-6}$ .

### Examples

standard input	standard output
9 0.000 0.5 1.8 22.50 1.0 1.7 90.00 0.5 1.0 115.0 1.0 1.2 135.0 1.0 1.2 180.0 1.0 1.2 225.0 1.0 1.2 270.0 1.0 1.2 295.0 1.0 1.2	1.65664252966349700991

### Note



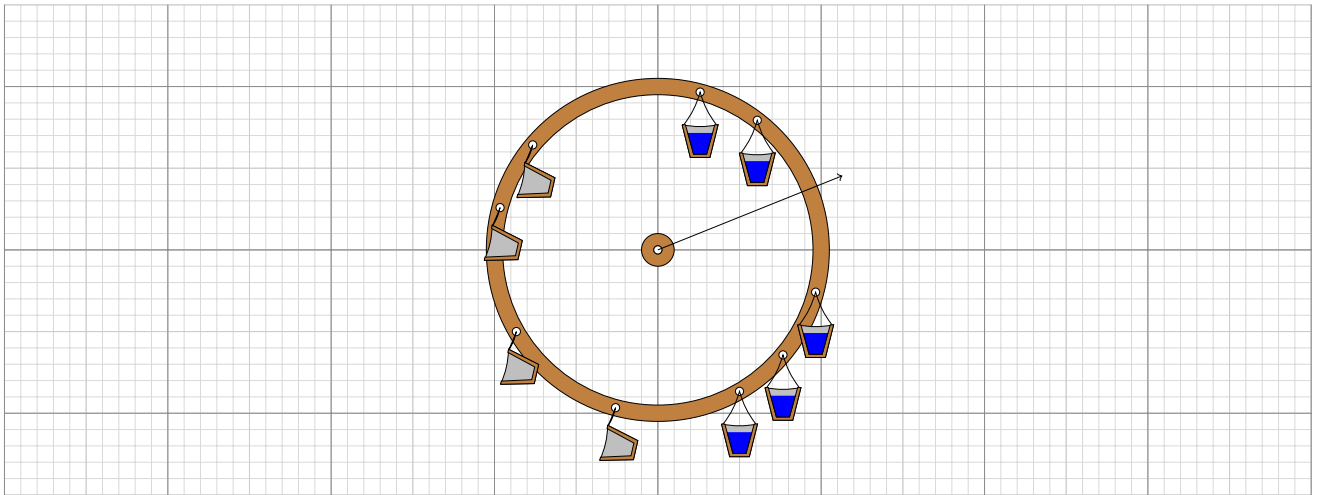


Illustration of Sample Input 1 at approximately 15.0 degrees clockwise from the start. The centre of mass is drawn as a vector from the origin.

## Problem H. Hacker's Heist

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Hatiko is a skilled hacker and works for the Byteland Intelligence Service.

She tricked her way into the server of Berland embassy through their local WiFi and managed to copy the files the informant told her about. Unluckily, the Berland authorities caught the informant, so they know what Hatiko did and sealed off the city. Hatiko needs to upload the data to the internet before they catch her.

Luckily the city is full of coffee shops with free wifi and Hatiko researched their details before she started her heist. The only downside is that they have strange opening hours and she may need to move between them. When travelling between coffee shops Hatiko can't upload any data at all.

Given the list of coffee shops, their opening hours and Wifi speed, what is the earliest time when all of the important data can be uploaded?

### Input

- The first line contains an integer  $d$  ( $1 \leq d \leq 10^9$ ), the size of the data in megabytes.
- The second line contains an integer  $n$  ( $1 \leq n \leq 100$ ), the number of cafes.
- The next  $n$  lines each contain:
  - The integer opening and closing times of the cafe in seconds,  $o$  and  $c$  ( $0 \leq o \leq c \leq 24 * 60 * 60$ ).
  - The wifi speed of the cafe  $w$  ( $1 \leq w \leq 1000$ ) in megabytes per second.
- $n$  lines, each with  $n$  integers, the time to travel to each cafe from the  $n^{th}$  cafe, in seconds ( $0 \leq d_i \leq 24*60*60$ ).

Hatiko may start in any single cafe when it opens and start using wifi the moment she arrives at a cafe.

### Output

One line containing an integer number of seconds, the smallest integer time by which all of the data can be uploaded.

### Examples

standard input	standard output
200 2 10 20 15 10 40 5 0 5 5 0	35

## Problem I. Interesting Game

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Igor and Ilona, both enjoying playing games very much, have discovered a new type of game: tree game. In this game, first player chooses a vertex of a tree. Then players (beginning with second player) alternately choose a neighbor of the last chosen vertex that was not chosen before, until a player can't make a move. This player is then declared loser and the other one is the winner. Igor moves first but unfortunately Ilona is a very experienced player and she will never make a mistake. Therefore Igor has asked you for help.

Our tree has  $N$  vertices conventionally numbered  $1 \dots N$  and exactly  $N - 1$  edges connecting them. Write a program, which determines all vertices in which Igor can start the game and win despite Ilona playing perfectly.

### Input

On the first line, there is a single number  $N$ , ( $1 \leq N \leq 2 \cdot 10^6$ ), which is equal to the number of nodes in the tree.  $N - 1$  lines follow, on the  $i$ -th line is a single integer  $a_i$  ( $1 \leq a_i \leq i$ ), which means there is edge in the tree connecting the vertex  $i + 1$  with the vertex  $a_i$ .

### Output

Output consists of several lines, on each one there is a single number of a node, where Igor can start the game and win, regardless of how Ilona would play. Numbers in the output should be sorted in ascending order.

### Examples

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

## Problem J. Jackpot

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

The thunderingly fun new game show of 'Jackpot' has bounded onto televisio screens up and down the land and is the next big thing. It runs along a simple premise: there are a number of doors, behind one of which is hidden a monetary prize and behind all of the others of which are hidden goats. Each contestant chooses how many doors will be opened before they pick a door to look at. The prize decreases as each door is opened. If they manage to open the door and find the money they get to keep it!

Not content with guessing, Jessica decided to try and work out the best number of doors to open to maximise her expected profit, where expected profit is the probability of winning multiplied by the remaining prize fund.

Jessica managed to find out that the remaining prize fund  $P_R$  is a function of the initial prize fund  $m$ , the number of doors opened  $d$  and a scaling factor  $f$ , ie.

$$P_R = m - (d \cdot f)^2$$

Given the number of doors, initial prize fund and the scaling factor can you work out how many doors should be opened before Jessica will try to pick a door to maximise her expected profit?

### Input

One line of input containing:

- One integer  $n$  ( $1 \leq n \leq 10^{18}$ ), the number of doors.
- One integer  $m$  ( $1 \leq m \leq 10^{18}$ ) the initial prize fund.
- One floating point  $f$  ( $0 < f \leq 1$ ), the scaling factor.

There will always only be one door with the best value.

### Output

One line containing one floating point number, the value of the expected payout. The output must be accurate to an absolute or relative error of at most  $10^{-6}$ .

### Example

standard input	standard output
4 100 1	91.0
20000 100500 0.8	5.0254931

## Problem K. Kings

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Kevin invented the new game, called “chess-tidying”. The game starts with a messy board containing several king pieces strewn across the cells. The player’s job is to put all of the pieces into a line along the primary diagonal of the board, which runs along black squares.

In each move, unlike in normal chess, one king may be moved one place either horizontally or vertically (but not both) to another unoccupied cell.

Kevin wants, given an instance of the game, to know how many moves you will need in order to finish it. Help him to do that

### Input

- One line with the number of rows and columns,  $n$  ( $1 \leq n \leq 500$ ).
- Each of the following  $n$  lines contains the two-dimensional integer coordinates  $c$  and  $r$  ( $1 \leq c, r \leq n$ ), the position of one of the kings.

Each of the kings starts at a unique position.

### Output

Output the minimum number of moves necessary to cover the main diagonal ( $r = c$ ) with kings.

### Examples

standard input	standard output
3 1 1 2 3 3 2	2
8 6 4 6 8 5 5 5 4 4 8 5 7 7 4 3 7	28

### Note

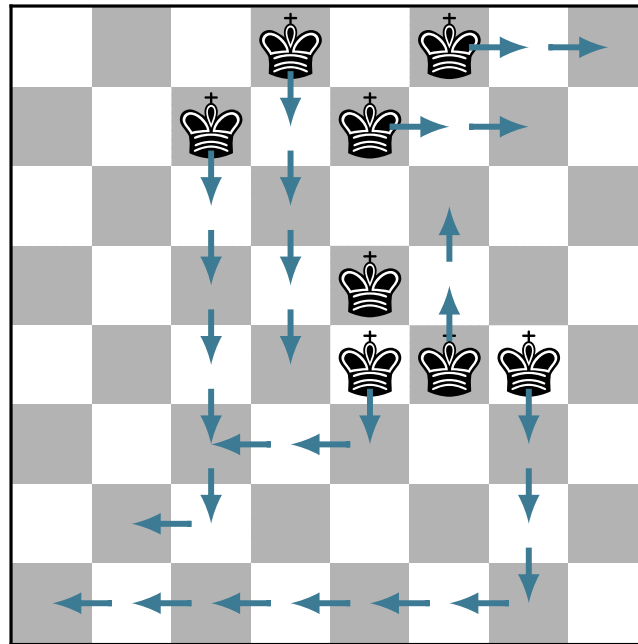


Illustration of Sample Input 2. In this case, the minimum number of moves necessary to put all of the kings along the black diagonal is 28 as pictured.

## Problem A. Polynomial in a Black Box

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

*This is an interactive problem.*

Alice has a black box which works with integers modulo  $m = 10^9 + 7$ . If a user types a number  $x$  on the keyboard of the box, the screen shows the number equal to the value of the polynomial  $p(x) = (a_d x^d + a_{d-1} x^{d-1} + \dots + a_1 x^1 + a_0) \bmod m$ . The degree  $d$  of the polynomial is unknown, as are its coefficients  $a_i$ . It is only known that  $0 \leq d \leq 10$  and  $a_d \neq 0$ .

Alice can type several numbers  $x$  and learn the values of the polynomial for these numbers. Help her find the degree  $d$  of the polynomial. She can input an  $x$  at most  $d + 3$  times.

### Interaction Protocol

To learn the value of the polynomial for number  $x$ , print a line of the form “ask  $x$ ” ( $0 \leq x < 10^9 + 7$ ). As a result, you will get a line with the value  $p(x)$ , or, if you asked more than  $d + 3$  such questions, you will get the number  $-1$  instead of the value, and evaluation of your solution will terminate.

To give the answer, print a line of the form “degree  $d$ ”. After that, terminate your solution gracefully.

After printing each line, flush the output buffer, or you will get the outcome **Idleness Limit Exceeded**: this can be done by calling, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python.

### Example

standard input	standard output
1000000006	ask 1
7	ask 3
34	ask 6
98	ask 10
	degree 2

### Note

In each test, the degree and the coefficients of the polynomial  $p(x)$  are chosen and fixed in advance.

In the example, which is also the first test in the testing system,  $p(x) = x^2 + 1\,000\,000\,005$ . All other tests were created as follows: first, the degree  $d$  was chosen ( $0 \leq d \leq 10$ ), and after that, one of the polynomials of such degree was chosen as  $p(x)$  uniformly at random.

## Problem B. Board Trick

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Danila the Magician and Mia the Assistant are showing a trick with a board.

First, Danila secludes himself, while Mia asks the audience to fill the  $8 \times 8$  square table drawn on the board: each cell can contain either zero or one.

Next, another member of the audience chooses the secret number: an integer from 1 to 64. The number is announced to Mia but not to Danila.

After that, Mia comes to the board and changes exactly one number on it: either a zero into one, or a one into zero.

Finally, Danila returns, looks at the board for a while, and happily announces the secret number he did not know. How do they perform the trick? Help Danila and Mia to prepare and then show it.

### Interaction Protocol

In this problem, your solution will be run twice on each test. Each test can contain several test cases.

During the first run, the solution acts as Mia the Assistant: given a board and a secret number, it has to change exactly one number on the board. The first line contains the word “Mia”. The second line contains an integer  $t$ : the number of test cases ( $1 \leq t \leq 1000$ ). The test cases follow. Each consists of nine lines. The first eight lines contain eight space-separated integers each: the numbers on the board (each equal to either 0 or 1). The ninth line contains the secret number (an integer from 1 to 64).

To answer a test case, the solution must print nine lines. On each of the first eight lines, print eight space-separated numbers: the numbers on the board (each equal to either 0 or 1). When comparing to the input, exactly one of them must be different: either a zero must turn into one or a one must turn into zero. On the ninth line, print three “minus” signs.

During the second run, the solution acts as Danila the Magician: given the final state of the board, it has to guess the secret number. The first line contains the word “Danila”. The second line contains an integer  $t$ : the number of test cases, the same as during the first run ( $1 \leq t \leq 1000$ ). The test cases follow. Each consists of nine lines. The first eight lines contain eight space-separated integers each: the numbers on the board, exactly the same as were printed (each equal to either 0 or 1). The ninth line contains three “minus” signs.

To answer a test case, the solution must print one line containing the secret number (an integer from 1 to 64).

During each run, each line of input including the last one is terminated by a newline.

### Examples

On each test, the input during the second run depends on the solution’s output during the first run. In the example, we will consider a solution where Danila simply says the sum of all numbers on the board as a guess. And Mia tries to make this sum equal to the secret number. Surely, this solution does not always work.

The two runs of this solution on the first test are shown below.



standard input	standard outptut
Mia 2 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 31	1 0 --- 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 ---
Danila 2 1 0 --- 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 ---	1 31

## Problem C. Connectivity

Input file: *standard input*  
Output file: *standard output*  
Time limit: 3 seconds  
Memory limit: 512 mebibytes

*This is an interactive problem.*

The jury made a random undirected graph of  $n$  vertices and  $m$  edges with no loops or parallel edges. In each test, the graph is randomly uniformly sampled from all possible graphs with fixed  $n$  and  $m$  before the testing of your program starts.

Your program has to determine whether the graph is connected, while knowing only  $n$  and  $m$  at first. The program can perform a “?  $i$ ” query ( $1 \leq i \leq n$ ) at most  $2 \cdot n$  times. In response for such query, the testing system will give either:

- Positive integer  $j$  ( $1 \leq j \leq n$ ) meaning that there is an edge between vertices  $i$  and  $j$  and that the edge was not previously communicated to the program (including any responses to “?  $j$ ” queries).
- Number  $-1$  meaning that all edges adjacent to the vertex  $i$  were already communicated to the program.

### Interaction Protocol

The first line of the input contains an integer  $T$ : the number of independent test cases for your program to handle. Your program will then have  $T$  interactions with the testing system.

An interaction starts with integers  $n$  and  $m$ , communicated to your program on a separate line: the number of vertices and edges in the secret graph ( $2 \leq n \leq 10^4$ ,  $1 \leq m \leq \min(\frac{n(n-1)}{2}, 10^4)$ ).

Afterwards, you can repeatedly print a request on a separate line in the following format: question mark without quotes (“?”, ASCII code 63), followed by a space, followed by an integer  $i$  ( $1 \leq i \leq n$ ): the number of the vertex for which you want to know the next adjacent edge.

In response for each such query, the testing system will print either a positive number of another vertex adjacent to  $i$ , or  $-1$ .

To give the answer, print a line with either “+” (plus sign, ASCII code 43) if the graph is connected, or “-” (minus sign, ASCII code 45) if the graph is not connected.

After printing the answer for the last  $T$ -th test case, terminate your solution gracefully.

The sum of  $n$  for all test cases in a run does not exceed  $10^5$ .

After printing each line, flush the output buffer, or you will get the outcome **Idleness Limit Exceeded**: this can be done by calling, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python.

## Example

standard input	standard output
2	
5 4	
	? 1
2	
	? 4
1	
	? 1
5	
	? 1
-1	
	? 4
3	
	? 4
-1	
	+
5 4	
	? 5
-1	
	-

## Note

Empty lines are added for clarity, they are absent during interaction.

In the example test, the two following graphs are used. The testing system responds to “?  $i$ ” with the first edge in the list which is adjacent to  $i$  and was not yet communicated to the program.

1.  $n = 5$ ,  $m = 4$ , edges are ordered as follows: 1–2, 1–4, 3–4, 1–5.
2.  $n = 5$ ,  $m = 4$ , edges are ordered as follows: 1–4, 1–3, 1–2, 3–4.

## Problem D. Absolute Pairwise Distance

Input file: *standard input*  
Output file: *standard output*  
Time limit: 3 seconds  
Memory limit: 512 mebibytes

John Doe invented a nice way to measure distance between two arrays of different length. Let  $a_1, \dots, a_{l_1}$  be the first array and  $b_1, \dots, b_{l_2}$  be the second one. Then  $d(a, b) = \sum_{i=1}^{l_1} \sum_{j=1}^{l_2} |a_i - b_j|$ . Unfortunately, this distance function does not satisfy the triangle inequality, but John decided to conduct a few experiments anyway.

John has a large array  $a_1, \dots, a_n$ . For  $q$  instances of values  $(l_1, r_1, l_2, r_2)$ , he would like to know the values  $d((a_{l_1}, a_{l_1+1}, \dots, a_{r_1}), (a_{l_2}, a_{l_2+1}, \dots, a_{r_2}))$ . Help him find these values.

### Input

The first line contains two integers  $n$  and  $q$ : the number of elements in the array and the number of queries ( $1 \leq n, q \leq 10^5$ ). The second line contains  $n$  integers  $a_1, \dots, a_n$ : the elements of John's large array ( $0 \leq a_i \leq 10^8$ ). The next  $q$  lines contain four integers each:  $l_1, r_1, l_2, r_2$ , which are the parameters of the respective query ( $1 \leq l_1 \leq r_1 \leq n, 1 \leq l_2 \leq r_2 \leq n$ ).

### Output

For each query, print the value of  $d((a_{l_1}, a_{l_1+1}, \dots, a_{r_1}), (a_{l_2}, a_{l_2+1}, \dots, a_{r_2}))$  on a separate line.

### Example

standard input	standard output
5 5	1
1 2 3 4 5	3
1 1 2 2	6
1 1 2 3	4
1 1 2 4	40
1 2 2 3	
1 5 1 5	

## Problem E. Multiplication and Division by 2

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Consider the number  $x$  stored in `uint32` data type. We can multiply or divide it by 2 any number of times in any order. Can we obtain the number  $y$  after some sequence of operations?

When  $a$  is stored in an `uint32`, and we multiply it by 2, it transforms into  $(a \cdot 2) \bmod 2^{32}$ . For example,  $(3 \cdot 2) \bmod 2^{32} = 6$ , and  $(2\,147\,483\,649 \cdot 2) \bmod 2^{32} = 2$ .

When  $a$  is stored in an `uint32`, and we divide it by 2, it transforms into  $\lfloor \frac{a}{2} \rfloor$ . For example,  $\lfloor \frac{6}{2} \rfloor = 3$ , and  $\lfloor \frac{3}{2} \rfloor = 1$ .

### Input

The first line contains an integer  $t$ , the number of test cases ( $1 \leq t \leq 1000$ ). The next  $t$  lines describe test cases, one per line. Each test case is given by two integers  $x$  and  $y$  ( $0 \leq x, y < 2^{32}$ ).

### Output

For each test case, print a single word on a separate line: “Yes” if we can turn  $x$  into  $y$  using the allowed operations, or “No” otherwise.

### Example

standard input	standard output
2	Yes
2147483649 1	No
9 13	

### Example explanation

In the first test case, we can multiply  $x$  by 2, and then divide the result by 2 to get  $y$ .

In the second test case, there is no way to turn  $x$  into  $y$ .

## Problem F. Festive Baobab

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Do you know where is the baobab in the Math&Mech facility? Well, if no, ask someone after the contest to show you...

Everyone knows that baobab is a tree. So, as any tree, it has trunk and branches. Branches grow from the trunk or from other branches.

For September 1, this baobab is usually decorated to make new students feel comfortable. For this purpose, there is a large chest with small colorful decorations deep in the building dungeons. Every decoration weighs exactly one gram.

Unfortunately, the baobab is very old, therefore it will be broken if overloaded with decorations. Specifically, for every branch, you know the maximum weight of decorations it can carry. If a branch, together with all branches growing from it (directly or via other branches), carries decorations such that their total weight exceeds this limit, this branch will break. This is unacceptable. But you can put several decorations on a single branch unless it causes some branch to break.

Some branches are near the entry and well shown, others are hidden in the depths of the baobab. Therefore, every decoration can bring more or less joy, depending on its position. The total joy of the baobab is the sum of joys of the branches where the decorations are located. If there are several decorations on a single branch, its joy is multiplied by their quantity.

What is the maximum possible total joy the baobab can have?

### Input

The first line contains two integers  $n$  and  $t$ : the number of branches on the baobab and the number of decorations ( $1 \leq n \leq 100\,000$ ,  $1 \leq t \leq 10^9$ ). Each of the next  $n$  lines contains three integers,  $d_i$ ,  $p_i$ , and  $w_i$ : the joy delivered by a single decoration on  $i$ -th branch, the branch that  $i$ -th branch grows from (or 0 if this branch grows directly from the trunk), and the maximum weight of decorations  $i$ -th branch can carry ( $1 \leq d_i, w_i \leq 10^9$ ,  $0 \leq p_i \leq n$ ).

It is guaranteed that every branch grows from the trunk, directly or via other branches. Also it is guaranteed that it is possible to use all decorations.

### Output

Output one integer: the maximum possible total joy delivered by the decorated baobab.

### Example

standard input	standard output
9 6 30 0 4 40 9 2 80 8 3 20 9 2 10 4 3 70 5 8 90 2 4 50 0 6 60 1 3	490

## Problem G. Flatland Elections

Input file: *standard input*  
Output file: *standard output*  
Time limit: 3 seconds  
Memory limit: 512 mebibytes

The elections to the parliament of Flatland are taking place soon!

There are  $N$  voters in Flatland. For each voter, we know their political preferences. Namely, we know their score on the scale Economic-left–Economic-right and on the scale Libertarian–Authoritarian, and each score is an integer. In this way, each voter is represented by a point on the plane with these two axes. Your party needs to do good in the elections, and for that it needs to get at least  $K$  votes.

Before the elections each party needs to present its program. The program is defined by the general line of the party, which is a line on the same plane going through the origin.

Of course, the voters prefer to vote for the party which is closer to their political preferences. The closeness is measured as the distance from the point corresponding to the voter to the line corresponding to the party. You can persuade any voter to vote for your party, but your expenses will be equal to the closeness of this voter to your party.

Your party has emerged very recently, and it does not have a clear political program yet. You are in charge of planning the party's budget. The budget must be sufficiently large to get at least  $K$  votes and to cover all expenses, no matter which general line your party will choose. Of course, you want to minimize the budget.

Can you determine the worst possible political program of your party and choose  $K$  voters to minimize the total expenses of this program?

### Input

The first line of input contains two integers  $N$  and  $K$ : the number of voters and the number of votes required for your party ( $1 \leq K \leq N \leq 2000$ ).

The following  $N$  lines contain coordinates of the voters on the political preference plane, two numbers on each line. The coordinates are integer and do not exceed  $10^6$  by the absolute value.

### Output

On the first line, output the minimum budget. On the second line, describe the worst political program of your party. Namely, output the coordinates of a point lying on the corresponding line which is not equal to the origin. The coordinates must not exceed  $2 \cdot 10^9$  by the absolute value. The coordinates and the total expenses may be non-integer.

On the third line, output the space-separated list of  $K$  indices of the voters which vote for your party. The voters are numbered in the order of appearance in the input, the order in which you output the indices does not matter.

The output is considered correct if the absolute or relative error does not exceed  $10^{-6}$  by the absolute value, both for the total expenses in the output compared to the correct answer, and for the total expenses in the output compared to the actual expenses for the political program and the list of voters in the output.

### Example

standard input	standard output
4 3	12.369316876852980869
3 3	-3.0 12.0
5 4	1 3 4
5 -5	
4 5	

## Problem H. Eggs

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Gosha has an egg tray for 16 eggs on the refrigerator door: two rows of sockets, eight sockets in each row, with an immovable separator in the middle. Initially, there are no eggs in the tray.

When there is at least one egg in the tray, Gosha can take one of them and eat it. When there are at most five eggs in the tray, Gosha can go to a grocery store, buy ten eggs there and place them in the tray. Please note that, after he buys new eggs and placing them in the tray, there is always at least one free socket left.

The eggs will spoil if they are not eaten for too long. So, every time Gosha takes an egg to eat, he should pick one of the oldest eggs in the tray: the ones which were bought earliest. He wants to place the newly bought eggs in such a way that, when he sees the positions of all eggs in the tray, he immediately knows which of them are the oldest. Help Gosha devise a strategy: where to place the eggs he buys and which egg to pick for eating so that the picked egg is one of the oldest.

### Input

There are two tests in this problem. In the first test, the first line contains the word “**sample**”, and in the second test the word “**test**”. The answer for the first test is checked for validity only, and on the second test, the strategy is also checked.

### Output

Print your strategy as a list of instructions.

Each instruction consists of an initial state (to the left), an action (at the center) and a final state (to the right). Please follow the format shown in the example! The formal output rules are given below the example.

The “**buy**” action must add exactly ten eggs, and the “**eat**” action must remove exactly one egg. A single initial state can occur in the instructions at most twice: at most once with the “**buy**” action and at most once with the “**eat**” action.

In this problem, formatting and the above rules are checked on both tests. However, the strategy checks described below are performed only on the second test.

Gosha uses the strategy as follows. First, he chooses the next possible action: he can eat an egg if there is at least one in the refrigerator, and he can place ten eggs he bought if there are at most five eggs in the refrigerator. After that, Gosha searches for an instruction with the action he chose where the initial state is the actual state of eggs in his refrigerator. Finally, Gosha changes the state to the final state of this instruction.

If there exists a sequence of possible actions such that, eventually, Gosha can not find the instruction he needs, or he picks an egg to eat which is not one of the oldest, the strategy is considered incorrect. If, for every sequence of possible actions, Gosha can find the next instruction, and the egg he eats is always one of the oldest, the strategy is considered correct. Please note that the strategy should account for every sequence of possible actions: when Gosha has two options (eat or buy), both have to be covered.

You are allowed to print instructions with initial states that can not be reached by any sequence of possible actions.

### Example

standard input	standard output
sample	<pre> **** **** eat **** **** ...* ****    ... **** --- .... .... buy ...* **** .... ....    ...* **** ---</pre>

### Note

Here are the rules for printing instructions. Each instruction takes three lines: the first two describe the states and the action, and the third one consists of three “-” characters (minus, ASCII code 45). In each state, empty sockets



are denoted by “.” (dot, ASCII code 46), eggs by “\*” (asterisk, ASCII code 42), and separators by “|” (vertical line, ASCII code 124). The actions are denoted by words “**buy**” (add ten new eggs) and “**eat**” (remove one egg). On the first line, the action is separated from the states by single spaces, and on the second line, the states are separated by five spaces.

## Problem I. Removing Pairs

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

The *pair removal* operation is removing two adjacent characters from a string. For example, given the string “abcd”, we can obtain “ab”, “ad”, and “cd” by pair removal.

You are given a string  $t$ . You can apply pair removal any number of times. Is it possible to obtain the string  $s$ ?

### Input

The first line contains the string  $s$ , and the second line contains the string  $t$  ( $1 \leq |s| \leq |t| \leq 10^5$ ). Both strings are composed of lowercase letters of English alphabet.

### Output

Print “YES” if it is possible to obtain  $s$  from  $t$  using pair removal, or “NO” otherwise.

Examples

standard input	standard output
axa abcbcxdda	YES
rrr rdfgdfgrdr	NO
w uwwu	NO

### Note

In the first test, one of the possible pair removal sequences is the following:

- ab**cb**cxdda
- abcx**dd**a
- ab**cx**a
- axa

## Problem J. Boxes on a Shelf

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Varya has a barn where she stores  $n$  boxes with important stuff. She recently nailed another shelf to the barn wall. Now she wants to put as many boxes as possible on the new shelf.

We shall look at the wall with the shelf as a plane. The shelf is then a horizontal segment of length  $L$ . There is empty space to the left and to the right of the shelf, which we can consider to be infinite. Box number  $i$  is a rectangle with sides  $a_i$  and  $b_i$ .

A box can stand on the shelf and not fall when one of its sides, or a part of it, lies on the shelf, and the center of the box is above an interior point of the shelf. Boxes can be placed only on the shelf, close to the wall: they can not be placed on top of other boxes or in several layers.

Find is the maximum number of boxes Varya can put on the shelf.

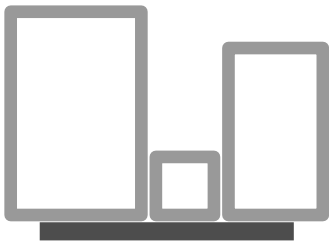

### Input

The first line contains an integer  $n$ : the number of boxes ( $1 \leq n \leq 100\,000$ ). Each of the following  $n$  lines contains two integers  $a_i$  and  $b_i$ : the sides of the respective box ( $1 \leq a_i, b_i \leq 100\,000$ ). The last line contains an integer  $L$ : the length of the shelf ( $1 \leq L \leq 10^9$ ).

### Output

Print one integer: the maximum number of boxes Varya can put on the shelf so that they don't fall off.

### Examples

standard input	standard output	Notes
<pre>4 3 5 6 4 10 20 2 2 7</pre>	3	
<pre>3 1 1 1 1 1 1 2</pre>	2	

### Explanations

In the first example, Varya can place three boxes on the shelf: for example, the second box lying on side of length 4 on the left, the fourth box lying on side of length 2 at the center, and the first box lying on side 3 on the right.

In the second example, if all three boxes are placed on the shelf symmetrically, they will fall off, as the centers of left and right boxes will be located exactly above the ends of the shelf.

## Problem K. Two Slicers

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Petya and his friends want to celebrate his birthday party! Petya has a round cake for the occasion. The only thing left is to divide it, and they will have tea.

In order to divide the cake, Petya is going to use slicers. Each slicer has a few blades: a  $k$ -slicer is a device which makes  $k$  straight cuts from the center of the cake to its boundary, and thus divides the round cake into  $k$  identical sectors.

Petya has a red  $a$ -slicer and a blue  $b$ -slicer. Fortunately, there are exactly  $a + b$  friends at the party including Petya. So he decided to use each slicer once, so that the cake will be divided into exactly  $a + b$  sectors.

After using the two slicers, the resulting  $a + b$  sectors may have different sizes. Nevertheless, the cake should be divided as fairly as possible: the difference between the largest sector and the smallest sector should be the minimum possible.

Find out what is the minimum possible difference that Petya can achieve. Find the difference between the areas of the largest and the smallest sectors after the optimal division. Regard the area of the whole cake as 1. Print the resulting area difference as an irreducible fraction.

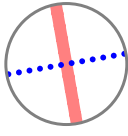
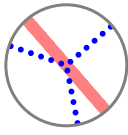
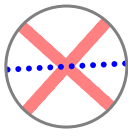
### Input

The first line of input contains two space-separated integers  $a$  and  $b$ : the parameters of red and blue slicers ( $2 \leq a, b \leq 100$ ).

### Output

Print an irreducible fraction in the form  $a / b$ : the difference between the areas of the largest and the smallest of the  $a + b$  resulting sectors after the slicers are applied optimally.

### Examples

standard input	standard output	Notes
2 2	0 / 1	
2 3	1 / 4	
4 2	1 / 8	

### Explanations

The result of optimal use of slicers is shown to the right of the examples. The cuts made by the red  $a$ -slicer are shown as pale red thick lines. The cuts made by the blue  $b$ -slicer are shown as blue thin dotted lines.

## Problem A. Array

Input file: *standard input*  
Output file: *standard output*  
Time limit: 3 seconds  
Memory limit: 512 mebibytes

bobo has an array  $a[1], a[2], \dots, a[n]$ .

He subsequently presents  $q$  questions of the following 4 types:

- 1  $l_i r_i c_i$  - Update  $a[k]$  with  $a[k] + c_i$  for all  $l_i \leq k \leq r_i$ ;
- 2  $l_i r_i c_i$  - Update  $a[k]$  with  $\min\{a[k], c_i\}$  for all  $l_i \leq k \leq r_i$ ;
- 3  $l_i r_i c_i$  - Update  $a[k]$  with  $\max\{a[k], c_i\}$  for all  $l_i \leq k \leq r_i$ ;
- 4  $l_i r_i$  - Ask for  $\min\{a[k] : l_i \leq k \leq r_i\}$  and  $\max\{a[k] : l_i \leq k \leq r_i\}$ .

### Input

The first line contains 2 integers  $n, q$  ( $1 \leq n, q \leq 200000$ ).

The second line contains  $n$  integers  $a_1, a_2, \dots, a_n$  which denotes the initial values of the array ( $|a_i| \leq 10^9$ ).

Each of the following  $q$  lines contains an integer  $t_i$  which denotes the type of  $i$ -th question. If  $t_i = 1, 2, 3$ , 3 integers  $l_i, r_i, c_i$  follows. If  $t_i = 4$ , 2 integers  $l_i, r_i$  follows. ( $1 \leq t_i \leq 4, 1 \leq l_i \leq r_i \leq n$ )

- If  $t_i = 1$ ,  $|c_i| \leq 2000$ ;
- If  $t_i = 2, 3$ ,  $|c_i| \leq 10^9$ .

### Output

For each question of type 4, two integers denote the minimum and the maximum.

### Examples

standard input	standard output
3 4	1 3
1 2 3	2 2
4 1 3	
1 1 2 1	
2 1 3 2	
4 1 3	

## Problem B. Chromatic Number

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 mebibytes

bobo has a **connected** graph  $G$ , and he wants to color each vertices with one of the  $c$  colors so that no two adjacent vertices share the same color.

Find the number of ways to color modulo  $(10^9 + 7)$ .

### Input

The first line contains 3 integers  $n, m, c$ , which denote the number of vertices, edges, and colors, respectively ( $1 \leq n \leq 10^5, n - 1 \leq m \leq n + 8, 1 \leq c \leq 10^9$ ).

The vertices are conveniently numbered by  $1, 2, \dots, n$ .

Each of the following  $m$  lines contains 2 integers  $a_i, b_i$ , which denotes an edge between vertices  $a_i$  and  $b_i$  ( $1 \leq a_i, b_i \leq n, a_i \neq b_i$ ).

### Output

A single integer denotes the number of ways.

### Examples

standard input	standard output
3 3 3 1 2 2 3 3 1	6
4 3 1000000000 1 2 2 3 3 4	3584

## Problem C. Nearest friend

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 mebibytes

In the country there are  $n$  houses connected by  $m$  bidirected roads. Distance between two houses is the length of shortest path between them.

There are  $k$  bobo living in the houses. For each bobo, find another bobo living nearest to him.

### Input

The first line contains 3 integers  $n, m, k$  ( $2 \leq n \leq 200000, n-1 \leq m \leq 200000, 2 \leq k \leq n$ ).

The houses are conveniently labeled by  $1, 2, \dots, n$ .

Each of the following  $m$  lines contains 3 integers  $a_i, b_i, c_i$ , which denotes a road between houses  $a_i$  and  $b_i$  with length  $c_i$  ( $1 \leq a_i, b_i \leq n, 1 \leq c_i \leq 10000$ ).

The last line contains  $k$  integers  $v_1, v_2, \dots, v_k$ , where  $v_i$  denotes the house the  $i$ -th bobo lives in ( $1 \leq v_i \leq n$ ).

It is guaranteed that every two houses can reach each other, and no two bobo live in the same house.

### Output

For each bobo, a single integer denotes the house where the nearest bobo lives. If there are multiple such bobo, find the house with the smallest label.

### Examples

standard input	standard output
4 3 3 1 2 1 2 3 1 3 4 1 2 3 4	3 2 3
3 3 3 1 2 1 2 3 1 3 1 1 3 2 1	1 1 2

## Problem D. Sequence Sorting

Input file: *standard input*  
Output file: *standard output*  
Time limit: 3 seconds  
Memory limit: 512 mebibytes

Given  $n$  sequences  $s_1, s_2, \dots, s_n$ , sort them in lexicographic order.

### Input

The first line contains an integer  $n$  ( $1 \leq n \leq 1000000$ ).

Each of the following  $n$  lines contains an integer  $k_i$  which denotes the length of  $s_i$ , followed by  $k_i$  integers  $s_{i,1}, s_{i,2}, \dots, s_{i,k_i}$  which denotes the sequence  $s_i$  ( $1 \leq s_{i,1} \leq s_{i,2} \leq \dots \leq s_{i,k_i} \leq 5000000$ ).

( $k_i \geq 1, k_1 + k_2 + \dots + k_n \leq 5000000$ )

Since the input is extremely large, it is recommended to use the following code snippet to read a 32-bit integer.

```
#include <cctype>
```

```
int get_int() {
    char ch = getchar();
    while (!isdigit(ch)) ch = getchar();
    int ret = 0;
    while (isdigit(ch)) {
        ret = ret * 10 + ch - '0';
        ch = getchar();
    }
    return ret;
}
```

### Output

$n$  integers  $p_1, p_2, \dots, p_n$ , where  $p_i$  denotes the index of the  $i$ -th smallest sequence.

If two or more sequences are the same, sort them according to their indices.

### Examples

standard input	standard output
4	1
2 1 3	4
1 12	3
3 1 3 4	2
2 1 3	



## Problem E. New Point

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 mebibytes

There are  $n$  distinct lattice points (whose coordinates are both integers) in a two-dimension plane, forming geometric patternings.

bobo is going to add a new lattice point so that the total number of right triangles whose legs are parallel to the coordinate axis is maximized.

Note that the new point should be chosen carefully to avoid coincidence.

### Input

The first line contains an integer  $n$  ( $1 \leq n \leq 200000$ ).

Each of the following  $n$  lines contains 2 integers  $x_i, y_i$  which denotes the point  $(x_i, y_i)$  ( $|x_i|, |y_i| \leq 10^9$ ).

### Output

A single integer denotes the maximum number of triangles after adding the point.

### Examples

standard input	standard output
4 0 1 1 0 0 -1 -1 0	4
5 0 0 0 1 1 0 0 -1 -1 0	9

## Problem F. Planar Graph Connectivity

Input file: *standard input*  
Output file: *standard output*  
Time limit: 4 seconds  
Memory limit: 512 mebibytes

bobo has a connected planar graph with  $n$  vertices.

He subsequently presents  $q$  questions of the following 2 types:

- -  $a_i b_i$  - Remove the edge between vertices  $a_i$  and  $b_i$ , and ask for the number of connected components.
- ?  $a_i b_i$  - Ask if vertices  $a_i$  and  $b_i$  are connected.

Answer his questions.

### Input

The first line contains 2 integers  $n, q$  ( $1 \leq n \leq 100000, 1 \leq q \leq 200000$ ).

Vertices are numbered by  $1, 2, \dots, n$  for convenience.

Each of the following  $n$  lines starts with an integer  $k_i$  which denotes the number of neighbors of vertex  $i$ , followed by  $k_i$  integers  $v_{i,1}, v_{i,2}, \dots, v_{i,k_i}$  which denote the neighbors, ordered in clockwise direction ( $0 \leq k_i \leq n-1, 1 \leq v_{i,j} \leq n$ ).

The following  $q$  lines denote the questions.

Note that the numbers (in the questions) are encoded. If the answer of the last question is last, then number  $x$  appears as  $x \oplus \text{last}$ . (Assume last = 0 at the beginning. " $\oplus$ " denotes bitwise exclusive-or.)

### Output

For the first type of questions, a single integer denotes the number of components.

For the second type of questions, "1" for connected and "0" for disconnected.

### Examples

standard input	standard output
4 3	1
3 2 3 4	2
2 1 4	0
1 1	
2 1 2	
- 1 2	
- 0 2	
? 3 1	

## Problem G. Popo Sort

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 megabytes

Popo, one of bobo's silly friends, is learning bogo sort. He immediately invents his own version, as shown in the following code.

```
function sort(s)
  success := False
  for i := 1, 2, ..., m do
    if s[a[i]] > s[b[i]] then
      swap s[a[i]] and s[b[i]]
      success := True
    end
  end
  if success then
    return sort(s)
  else
    return s
  end
end
```

Determine whether Popo's version is correct (can sort all possible sequences of length  $n$  into non-descending order).

### Input

The first line contains 2 integers  $n, m$  ( $1 \leq n \leq 1000000, 0 \leq m \leq 1000000$ ).

Each of the following  $m$  lines contains 2 integers  $a_i, b_i$  ( $1 \leq a_i, b_i \leq n$ ).

### Output

Print "Yes" if the version is correct, or "No" otherwise.

### Examples

standard input	standard output
3 2 1 2 2 3	Yes
2 2 1 2 2 1	No

## Problem H. Power of Three

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 megabytes

bobo has a binary matrix of size  $n \times m$ . Today bobo is going to remove some (maybe none) of rows to maximize the *boboness* of the matrix.

The *boboness* is defined as follows. If there are odd number of ones in the  $i$ -th column,  $a_i \cdot 3^{b_i}$  points are added up to the *boboness*, initially 0.

Find the maximum of *boboness*.

### Input

The first line contains 2 integers  $n, m$  ( $1 \leq n \leq 200000, 1 \leq m \leq 70$ ).

Each of the following  $n$  lines contains  $m$  integers which denotes the matrix.

Each of the last  $m$  lines contains 2 integers  $a_i, b_i$  ( $a_i \in \{-1, 1\}, 1 \leq b_i \leq 35$ ).

It is guaranteed that for all  $i \neq j$ , either  $a_i \neq a_j$  or  $b_i \neq b_j$ .

### Output

A single integer denotes the maximum of *boboness*.

### Examples

standard input	standard output
2 4 1101 0010 -1 1 -1 2 1 1 1 2	3
1 1 1 -1 1	0

## Problem I. Remove Obstacles

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 mebibytes

bobo is in a maze with 2 rows and  $n$  columns. The rows are numbered from top to bottom, while the columns are numbered from left to right. Some cells may contain obstacles.

bobo starts at cell  $(1, 1)$  and is going to cell  $(2, n)$ . He can step upward, downward or rightward. However, he cannot step into cells with obstacles or go out of the maze. Meanwhile, he won't visit a cell more than once.

As a magician, bobo can remove at most  $k$  obstacles. He wonder the maximum number of cells he can visit.

### Input

The first line contains 2 integers  $n, k$  ( $1 \leq n \leq 1000000, 0 \leq k \leq 2000000$ ).

The second and third line each contains  $n$  characters which denotes the maze, where “#” denotes obstacle cell and “.” denotes empty cell.

It is guaranteed that bobo can go from  $(1, 1)$  to  $(2, n)$  without removing any obstacles.

### Output

A single integer denotes the maximum number of cells.

### Examples

standard input	standard output
2 1 .# ..	3
3 1 .#. ...	6

## Problem J. Salesmen

Input file: *standard input*  
Output file: *standard output*  
Time limit: 1 second  
Memory limit: 512 mebibytes

bobo lives in a country where personal rockets are big fashion. The country consists of  $n$  cities which are conveniently numbered by  $1, 2, \dots, n$ .

Cities are connected by bidirectional roads, and there is exactly one path between any two cities.

There are  $m$  salesmen in bobo's country. The  $i$ -th salesman travels along the roads between cities  $a_i$  and  $b_i$  and sells  $c_i$  rockets.

Since the rockets are not very high-quality, people in the  $i$ -th city will buy at most  $w_i$  rockets.

Now bobo wants to know how many rockets can be sold in salesmen's best effort (i.e. the maximum number).

### Input

The first line contains 2 integers  $n, m$  ( $1 \leq n, m \leq 10000$ ).

The second line contains  $n$  integers  $w_1, w_2, \dots, w_n$  ( $0 \leq w_i \leq 100000$ ).

Each of the following  $(n - 1)$  lines contains 2 integers  $u_i, v_i$  which denotes a road between cities  $u_i$  and  $v_i$  ( $1 \leq u_i, v_i \leq n$ ).

Each of the last  $m$  lines contains 3 integers  $a_i, b_i, c_i$  ( $1 \leq a_i, b_i \leq n, 0 \leq c_i \leq 100000$ ).

### Output

A single integer denotes the maximum number of rockets can be sold.

### Examples

standard input	standard output
4 2 0 1 2 2 1 4 2 4 3 4 1 2 2 1 3 3	5

## Problem A. Just a flow

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

You are given a system of nodes and pipes. Water may flow throw pipes. For each pipe you know maximal speed of the water flow. Each moment of time for each node except source and sink amount of water flowing in the node is equal to amount of water flowing out the node

You have to find maximal amount of water per unit time which can flow between source and sink. Also you have to find speed of the flow along each pipe, which should be to acquire maximal amount.

All pipes can be used in both directions. Between pair of nodes can be more than one pipe.

### Input

The first line contain integer  $N$  — number of nodes ( $2 \leq N \leq 100$ ). All nodes are numbered from 1 to  $N$ . The source has number one, sink has number  $N$ . The first line contain integer  $M$  — number of pipes ( $1 \leq M \leq 5\,000$ ). Next  $M$  lines describe pipes. Each pipe is described by three integers  $A_i, B_i, C_i$ , where  $A_i, B_i$  — indices of nodes, which the pipe connects, and  $C_i$  ( $0 \leq C_i \leq 10^4$ ) — maximal speed of the water flow throw the pipe.

### Output

In the first line output maximal amount of water per unit of time.  $i$ -th of the next  $M$  lines should contain speed throw the pipe number  $i$ . If direction of the flow throw the pipe differs from order of nodes in input, output the speed with sign minus; Output numbers with accuracy at least 3 digits after decimal point.

### Example

standard input	standard output
2	4.0000000
2	1.0000000
1 2 1	-3.0000000
2 1 3	

## Problem B. Cut

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

You are given undirected graph. Find minimal cut between vertices 1 and  $n$ .

### Input

The first line contain integers  $n$  ( $1 \leq n \leq 100$ ) — number of vertices in the graph, and  $m$  ( $0 \leq m \leq 400$ ) — number of edges in the graph. Next  $m$  lines describe the edges. Each edge is described by numbers of its ends and its capacity (positive integer no more than 10 000 000). There are no multiedges and loops.

### Output

On the first line output number of edges going throw minimal cut and its summary capacity. Next line should contain increasing sequence of indecies of edges (edges are numbered by integers from 1 to  $m$  in order they are given).

### Example

standard input	standard output
3 3	2 8
1 2 3	1 2
1 3 5	
3 2 7	



## Problem C. Decomposition of flow

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

You are given directed graph. Each edge has integer capacity. You have to find max flow from vertex number 1 to vertex number  $n$  and decompose it as union of paths.

### Input

The first line contains numbers  $n$  ( $2 \leq n \leq 500$ ) and  $m$  ( $1 \leq m \leq 10\,000$ ) — number of vertices and edges in the graph.  $i$ -th of next  $m$  lines contains three integers  $a_i, b_i, c_i$  means edge from  $a_i$  to  $b_i$  with capacity  $c_i$  ( $1 \leq c_i \leq 10^9$ ).

### Output

Output number of paths (equals to the value of max flow). Each of the next lines should contain description of pathes. Each path is described as *value*  $k\ e_1, e_2, \dots, e_k$  (value of path, number of edges in the path, indices of the edges of the path). Edges are numbered from 1 to  $m$  in order they are given.

### Examples

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

## Problem D. Snails

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

Two snails Masha and Petya are on the lawn with apricots and want to go back to their home. Lawns are numbered by integers from 1 to  $n$  and are connected by roads. Two lawns may be connected by many roads. It may happen, a road connects a lawn with itself. Due to reasons of hygiene, if one snail already went along the road, another one can not use this road. Help to Petya and Masha to get back home.

### Input

The first line contain four integer numbers —  $n, m, s$  и  $t$  (number of lawns, number of roads, index of lawn with apricots, index of lawn with the home). Next  $m$  lines describe roads. Each line contain pair of integers  $(x, y)$ , means there is road from lawn number  $x$  to lawn number  $y$ . (due to the nature of snails and the space all roads are one-way).

Limitations:  $2 \leq n \leq 10^5, 0 \leq m \leq 10^5, s \neq t$ .

### Output

If there is a solution output **YES** and two lines with sequences of indices of lawns in the paths. At first output the path for Masha, then output the path for Petya. If there is no solution output **NO**. If there are several solutions output any of them.

### Example

standard input	standard output
3 3 1 3	YES
1 2	1 3
1 3	1 2 3
2 3	

### Note

You are given oriented graph, find two disjoint by edges paths from  $s$  to  $t$ . Output vertices of desired paths.

## Problem E. Matan

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

In university of city M. experiment takes place. Tutors decide what to tell during the course of lections.

One tutor specialized on mathematical analysis (Matan) valued all known to him themes in the course. As result each theme has a number (possible negative) — usefulness of the topic. Tutor wants to maximize summary usefulness of the themes, but it's not so easy. To make students understand theme, you need to tell them some other themes, beacause of some proofs are based on facts from other themes. But if there is cycle of dependencies, you may read all the themes of the cycle and finally all stdudents will understand all the themes from the cycle.

You have to find set of themes. This set should be clear for students and usefulness of the set should be maximized.

### Input

The first line contains integer  $N$  ( $1 \leq N \leq 200$ ) — number of themes. The second line contains  $N$  integers, not exceeding 1 000 by absolute value — usefulness of themes. Next  $N$  lines describe dependencies between themes.  $i$ -th line start with  $k_i$  — number of themes students should already know to understand  $i$ -th theme. Then  $k_i$  diffetent integers from 1 to  $N$  — indecies of the themes;

Summary number of dependencies do not exceed 1 800.

### Output

Output the only number — maximal summary usefulness of choosen themes;

### Examples

standard input	standard output
4 -1 1 -2 2 0 1 1 2 4 2 1 1	2
3 2 -1 -2 2 2 3 0 0	0

## Problem F. Perspective

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

Breaking news! A Russian billionaire has bought a yet undisclosed NBA team. He's planning to invest huge effort and money into making that team the best. And in fact he's been very specific about the expected result: the first place.

Being his advisor, you need to determine whether it's possible for your team to finish first in its division or not.

More formally, the NBA regular season is organized as follows: all teams play some games, in each game one team wins and one team loses. Teams are grouped into divisions, some games are between the teams in the same division, and some are between the teams in different divisions.

Given the current score and the total number of remaining games for each team of your division, and the number of remaining games between each pair of teams in your division, determine if it's possible for your team to score at least as much wins as any other team in your division.

### Input

The first line of the input file contains  $N$  ( $2 \leq N \leq 20$ ) — the number of teams in your division. They are numbered from 1 to  $N$ , your team has number 1.

The second line of the input file contains  $N$  integers  $w_1, w_2, \dots, w_N$ , where  $w_i$  is the total number of games that  $i^{th}$  team has won to the moment.

The third line of the input file contains  $N$  integers  $r_1, r_2, \dots, r_N$ , where  $r_i$  is the total number of remaining games for the  $i^{th}$  team (including the games inside the division).

The next  $N$  lines contain  $N$  integers each. The  $j^{th}$  integer in the  $i^{th}$  line of those contains  $a_{ij}$  — the number of games remaining between teams  $i$  and  $j$ . It is always true that  $a_{ij} = a_{ji}$  and  $a_{ii} = 0$ , for all  $i$   $\sum_j a_{ij} \leq r_i$ .

All the numbers in the input file are non-negative and don't exceed 10 000.

### Output

On the only line of output, print "YES" (without quotes) if it's possible for the team 1 to score at least as much wins as any other team of its division, and "NO" (without quotes) otherwise.

### Example

standard input	standard output
3 1 2 2 1 1 1 0 0 0 0 0 0 0 0 0	YES
3 1 2 2 1 1 1 0 0 0 0 0 1 0 1 0	NO

## Problem G. Full orientation

Input file: *standard input*  
Output file: *standard output*  
Time limit: 8 seconds  
Memory limit: 256 mebibytes

You are given undirected graph without loops and multiedges. You have to make this graph directed in such a way, that maximal outgoing degree is minimal possible.

### Input

The first line contains numbers  $n$  and  $m$  — number of vertices and edges in the graph ( $1 \leq n, m \leq 25\,000$ ). Next  $m$  lines contain pairs of integers from 1 to  $n$  — edges of the graph.

### Output

Output  $m$  integers from 0 to 1. If  $i$ -th edge is given as  $a_i, b_i$  then zero means it should go from  $a$  to  $b$ , and one means edge from  $b$  to  $a$ .

### Examples

standard input	standard output
4 4 1 2 1 3 4 2 4 3	1 0 1 1 0
5 5 1 2 2 3 3 1 1 4 1 5	1 0 0 0 1 1

## Problem H. Looking for Brides

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

Once upon a time the king of Flatland decided to send his  $k$  sons in a journey, so each of them could find a bride. Everyone knows that Flatland consists of  $n$  cities, and some of them are connected by roads. The king lives in the capital, the city number 1, and the city number  $n$  is famous for its brides.

Long story short, king's will was that each his son must travel from city 1 to city  $n$  by roads. Many brides live in city  $n$ , though much less are beautiful, so young princes beware each other. If any son passes through some road then others avoid taking it even afterwards, because there may be traps on it.

The king loves his children and respects their time. He wants to select their paths to minimize the average travel time.

### Input

The first line of the input contains three integers  $n, m, k$  — the number of cities and roads in Flatland and the number of king's sons ( $2 \leq n \leq 200$ ,  $1 \leq m \leq 2000$ ,  $1 \leq k \leq 100$ ). Next  $m$  lines describe roads. Each of them contains three integers — the numbers of cities it connect and its length (not exceeding  $10^6$ ). Each road is bidirectional. There may be several roads between any pair of cities.

### Output

If it is impossible to send all sons using non-intersecting paths, print -1.

Otherwise print the minimum possible average travel time with absolute or relative precision  $10^{-5}$ . In the next  $k$  lines print sons' paths. Each path must follow the format  $c_i e_1 e_2 \dots$ , where  $c_i$  is the number of roads in the path and  $e_j$  are the ids of the roads. Roads are numbered from 1 in the order of input.

### Example

standard input	standard output
5 8 2	3.00000
1 2 1	2 3 8
1 3 1	2 2 6
1 4 3	
2 5 5	
2 3 1	
3 5 1	
3 4 1	
5 4 1	