

Problem A. Archery Tournament

Time limit: 3 seconds

You were invited to the annual archery tournament. You are going to compete against the best archers from all of the Northern Eurasia. This year, a new type of competition is introduced, where a shooting range is dynamic and new targets might appear at any second.

As the shooting range is far enough from you, it can be represented as a 2D plane, where $y = 0$ is the ground level. There are some targets in a shape of a circle, and all the targets are standing on the ground. That means, if a target's center is (x, y) ($y > 0$), then its radius is equal to y , so that it touches the line $y = 0$. No two targets simultaneously present at the range at any given time intersect (but they may touch).

Initially, the shooting range is empty. Your participation in this competition can be described as n events: either a new target appears at the range, or you shoot an arrow at some point at the range. To hit a target, you must shoot strictly inside the circle (hitting the border does not count). If you shoot and hit some target, then the target is removed from the range and you are awarded one point.

Input

The first line of the input contains integer n ($1 \leq n \leq 2 \cdot 10^5$). Next n lines describe the events happening at the tournament. The i -th line contains three integers t_i , x_i , and y_i ($t_i = 1, 2$; $-10^9 \leq x_i, y_i \leq 10^9$; $y_i > 0$).

- If $t_i = 1$, then a new target with center (x_i, y_i) and radius y_i appears at the range.
- If $t_i = 2$, then you perform a shot, which hits the range at (x_i, y_i) .

Output

For each of your shots, output a separate line with the single integer. If the shot did not hit any target, print “-1”. If the shot hit a target, print the number of event when that target was added to the range. Events are numbered starting from 1.

Examples

input	output	illustration
8	-1	
1 0 12	-1	
2 -11 22	3	
1 24 10	1	
1 12 3		
2 12 12		
2 16 14		
1 28 15		
2 3 6		

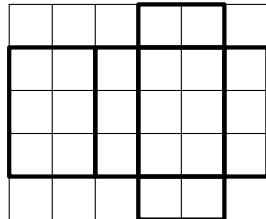
Note

Illustration shows the state of the range after first six events. The rightmost target was hit by the last shot and is going to be removed.

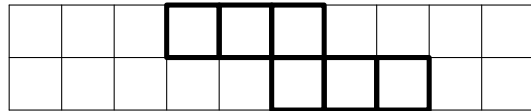
Problem B. Box

Time limit: 3 seconds

Bella is working in a factory that produces boxes. All boxes are in a shape of rectangular parallelepipeds. A *net* of the corresponding parallelepiped is cut out of a flat rectangular piece of cardboard of size $w \times h$. This net is a polygon with sides parallel to the sides of the rectangle of the cardboard. The net is bent along several lines and is connected along the edges of the resulting parallelepiped to form a box. The net is bent only along the edges of the resulting box.



The first example



The third example

Bella is a software developer and her task is to check whether it is possible to make a box of size $a \times b \times c$ out of a cardboard of size $w \times h$. Bella did write a program and boxes are being produced. Can you do the same?

Input

The first line contains three integers a , b , and c — the dimensions of the box.

The second line contains two integers w and h — the width and the height of the cardboard.

All integers are positive and do not exceed 10^8 .

Output

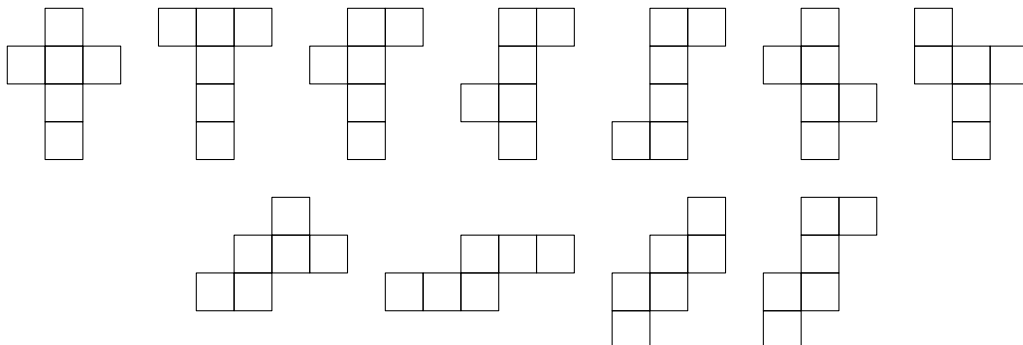
Print “Yes” if it is possible to cut a box $a \times b \times c$ out of a cardboard of size $w \times h$. Print “No” otherwise.

Examples

input	output
1 2 3 6 5	Yes
1 2 3 5 5	No
1 1 1 10 2	Yes

Note

There are 11 different nets of a cube, ignoring rotations and mirror images.



Problem C. Connections

Time limit: 3 seconds

Hard times are coming to Byteland. Quantum computing is becoming mainstream and Qubitland is going to occupy Byteland. The main problem is that Byteland does not have enough money for this war, so the King of Byteland Byteman 0x0B had decided to reform its road system to reduce expenses.

Byteland has n cities that are connected by m one-way roads and it is possible to get from any city to any other city using these roads. No two roads intersect outside of the cities and no other roads exist. By the way, roads are one-way because every road has a halfway barrier that may be passed in one direction only. These barriers are intended to force enemies to waste their time if they choose the wrong way.

The idea of the upcoming road reform is to abandon some roads so that exactly $2n$ roads remain. Advisers of the King think that it should be enough to keep the ability to get from any city to any other city. (Maybe even less is enough? They do not know for sure.) The problem is how to choose roads to abandon. Everyone in Byteland knows that you are the only one who can solve this problem.

Input

Input consists of several test cases. The first line of the input contains the number of tests cases.

The first line of each test case contains n and m — the number of cities and the number of roads correspondingly ($n \geq 4$, $m > 2n$). Each of the next m lines contains two numbers x_i and y_i denoting a road from city x_i to city y_i ($1 \leq x_i, y_i \leq n$, $x_i \neq y_i$). It is guaranteed that it is possible to get from any city to any other city using existing roads only. For each pair (x, y) of cities there is at most one road going from city x to city y and at most one road going from city y to city x . The solution is guaranteed to exist. The sum of m over all test cases in a single input does not exceed 100 000.

Output

For each test case output $m - 2n$ lines. Each line describes a road that should be abandoned. Print the road in the same format as in the input: the number of the source city and the number of the destination city. The order of roads in the output does not matter, but each road from the input may appear in the output at most once and each road in the output must have been in the input. It still must be possible to get from any city to any other city using the remaining roads.

Example

input	output	illustration
1 4 9 1 2 1 3 2 3 2 4 3 2 3 4 4 1 4 2 4 3	1 3	

Problem D. Designing the Toy

Time limit: 3 seconds

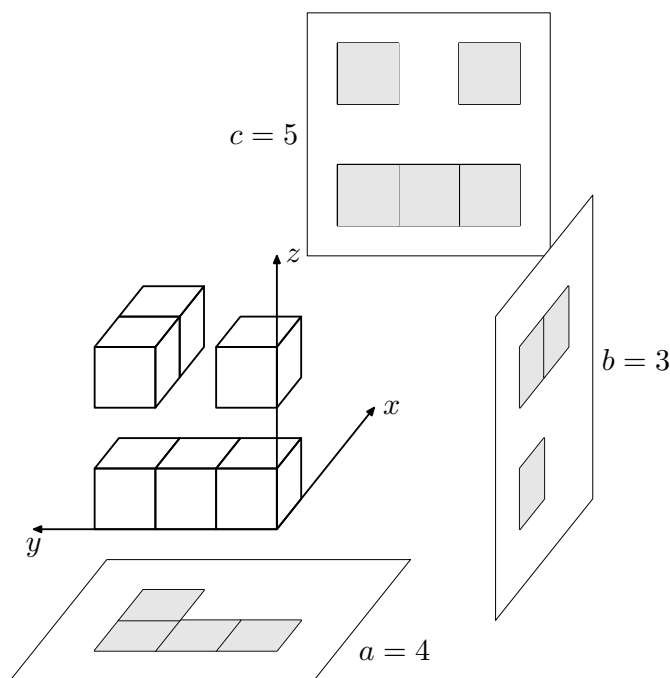
You are the main toy producer in the city. Recently you bought a 3D-printer that provides you with an unprecedented opportunity for designing new fancy toys for children.

In a package with the 3D-printer there is a booklet containing several examples of what can be created with it. One of the examples is a figure that looks like a triangle, like a circle, or like a square depending on which of its sides you are looking at.

Unfortunately, it turned out that the booklet describes the most recent version of your printer. Your printer is only able to create figures that consist of voxels (three-dimensional pixels), i.e. figures that look like a union of a unit-length cubes that are the cells of a three-dimensional grid. Thus, you are not able to print any “smooth” figures (like a sphere, for example) with it. On the other hand, important feature of your model is its ability to create figures whose parts are not even connected with each other by putting wires of a negligible thickness between them.

You like the idea of the figure in the booklet, so you decided to improve upon this idea. Instead of specifying shapes of figure projections from different perspectives, you would specify their areas.

In this problem, a voxel is defined by a triple of integers (x, y, z) , which corresponds to a unit cube $[x, x + 1] \times [y, y + 1] \times [z, z + 1]$.



You are given three positive integers a , b and c . Your task is to find a description of a figure F consisting of one or more voxels, for which the area of its orthogonal projection onto the plane Oxy is a , the area of its orthogonal projection onto the plane Oxz is b , and the area of its orthogonal projection onto the plane Oyz is c , or to determine that it is impossible.

Input

The only line of the input contains three integers a , b , and c ($1 \leq a, b, c \leq 100$) — the desired area of orthogonal projections onto the planes Oxy , Oxz , and Oyz correspondingly.

Output

If it is impossible to find the desired figure, print only the integer -1 .

Otherwise, in the first line print the integer n , defining the number of voxels in the figure. Then print n triples x, y, z ($-100 \leq x, y, z \leq 100$) defining the voxels of the figure. Voxels may be printed in any order, but no voxel may be repeated twice.

The number of voxels n should not exceed 10^6 .

Any figure with the requested projection areas is accepted.

Examples

input	output
4 3 5	6 0 0 0 0 1 0 0 2 0 0 2 2 1 2 2 0 0 2
100 1 1	-1

Problem E. Easy Quest

Time limit: 3 seconds

A young hero is starting his heroic life. The wise wizard suggested him an easy first quest. During this quest our young hero meets n magical creatures, in specific order. In order to help the young hero, the wizard gave him a clue — a list of n integers a_i .

- If a_i is positive, then the i -th magical creature is benevolent and gives to our hero one magical item of type a_i . The hero can keep several items of the same type.
- If a_i is negative, then the i -th magical creature is evil and in order to defeat it the young hero needs one magical item of type $-a_i$. All magical items are fragile and can be used only once.
- If a_i is zero, then the i -th creature is a unicorn. It gives the hero any magical item he asks for, but only one.

Your task is to help the young hero to finish the first quest, defeating all enemies on the way, or say that it is impossible.

Input

The first line of input contains one integer n ($1 \leq n \leq 1000$). The second line contains n integers a_i ($-1000 \leq a_i \leq 1000$).

Output

If it is impossible to defeat all enemies, then output one string “No”. If it is possible, then output string “Yes”, and in the next line output the types of items the hero should ask the unicorns for, in order they meet during the quest. Types must be integers in range from 1 to 1000 inclusive. If there are several solutions, output any of them.

Examples

input	output
10 1 0 -4 0 0 -1 -3 0 -1 -2	Yes 4 1 3 2
5 5 8 0 -6 -3	No
3 2 -2 -2	No

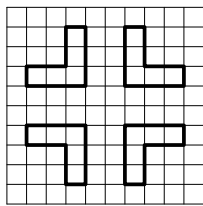
Problem F. The Final Level

Time limit: 3 seconds

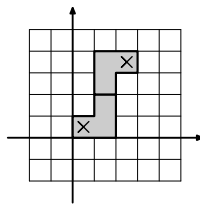
Fiora is a game designer. Now she is designing the final level for her new game.

A level for this game is a labyrinth on a rectangular grid with lots of enemies. Player starts her game at the square $(0,0)$ and her purpose is to get to the square (a,b) . Fiora has lots of ideas on how to put enemies, but she does not like designing labyrinths. She needs your help here.

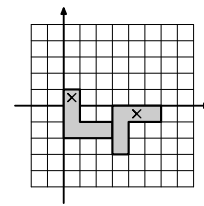
Fiora is drawing levels in a special level editor which supports one basic block to design a labyrinth. This block is an L-shaped corner, consisting of two perpendicular rectangles $1 \times n$ squares in size intersecting at 1×1 square. It is possible to rotate this block in four ways. Blocks cannot intersect, but they can touch each other. Player can move through all the squares lying in any block. She can move between two squares if they are sharing a side, even if they are in different blocks.



Blocks with $n = 3$



The first example



The second example

Fiora wants to design the final level with the minimal possible number of blocks. Of course, it should be possible to move from square $(0,0)$ to square (a,b) .

Input

The first line of the input consists of a single integer m ($1 \leq m \leq 100$) — the number of test cases. It is followed by m test cases. Each test case is on a separate line and consists of three integers a , b , and n ($-10^8 \leq a, b \leq 10^8$; $2 \leq n \leq 10^8$) — a is the coordinate of the final point along the horizontal axis, b is the coordinate of the final point along the vertical axis, and n is the size of the block. The final point is not same as the starting one (either $a \neq 0$ or $b \neq 0$).

Output

For each test case, in the first line print the minimal number k of blocks you need. In the following k lines print description of these blocks. Each L-shaped corner block is described by coordinates of two cells. Print coordinates of the end of its vertical rectangle, followed by coordinates of the end of its horizontal rectangle. Specify the coordinates of the ends that are opposite to the intersection of the rectangles. Note that the order of cells in the block description matters, since a change of the order results in a reflected block. Coordinates of each end should be printed with the coordinate along the horizontal axis first, followed by the coordinate along the vertical axis.

All coordinates in the output should not exceed 10^9 by absolute value.

It is guaranteed that the total number of blocks in the correct output does not exceed 10^5 for all test cases combined.

Examples

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

Problem G. The Great Wall

Time limit: 3 seconds

Recently you had become an emperor of a small country of Sinai. You had decided to build a big wall at the border to save your country from barbarian raids. You had contacted “W Corp”, the only company in the world that builds non-penetrable walls.

“W Corp” builds each wall using the same pattern. The length of the wall is n meters. Each one-meter piece of the wall is numbered by an integer from 1 to n along its length and may have a different height. The height pattern is based on three fixed arrays a , b , and c of n elements each, such that $a_i < b_i < c_i$ for all $1 \leq i \leq n$, and an integer r ($1 \leq r < n$). These arrays and r are the same for any wall that is built by “W Corp”.

The choice of the specific wall design is determined by two distinct integers x and y ($1 \leq x < y \leq n-r+1$) in the following way. Take two ranges of integers: $[x, x+r-1]$ and $[y, y+r-1]$ (these ranges are inclusive of their ends). Then the height of the wall at one meter piece i for all $1 \leq i \leq n$ is equal to:

- a_i if i does not belong to any of the chosen ranges;
- b_i if i belongs to exactly one chosen range;
- c_i if i belongs to both chosen ranges.

A *strength* of a wall is defined as the sum of all heights of its n one meter pieces.

The arrays a , b , c , and an integer r are the same for any wall built by “W Corp”, so the company provides a price list with all the possible wall designs, sorted in non-decreasing order of their strength. You choose the k -th wall design from the list. The task is to find the strength of the chosen wall.

Input

The first line of the input contains three integers n , r and k ($2 \leq n \leq 30\,000$, $1 \leq r < n$, $1 \leq k \leq \frac{(n-r)(n-r+1)}{2}$) — the length of the wall, the length of the segments to choose, and the position of the wall in the price list.

The second line of the input contains the elements of the array a ($1 \leq a_i \leq 10^6$).

The third line of the input contains the elements of the array b ($a_i < b_i \leq 10^6$).

The fourth line of the input contains the elements of the array c ($b_i < c_i \leq 10^6$).

Output

Print one integer — the strength of the k -th wall from “W Corp” price list.

Examples

input	output
4 2 1 1 2 3 4 3 3 5 5 7 7 7 7	16

Note

In the sample test we could build three different walls:

- The choice of $x = 1$ and $y = 2$ yields heights “3 7 5 4” with the strength of 19;
- The choice of $x = 1$ and $y = 3$ yields heights “3 3 5 5” with the strength of 16;
- The choice of $x = 2$ and $y = 3$ yields heights “1 3 7 5” with the strength of 16.

Problem H. Hack

Time limit: 10 seconds

Heidi is analyzing a peculiar device. This device takes an a as input and computes $a^d \pmod n$ using the following pseudocode and some integers d and n stored in this device:

```
1 modPow(a, d, n) {
2   r = 1;
3   for (i = 0; i < 60; ++i) {
4     if ((d & (1 << i)) != 0) {
5       r = r * a % n;
6     }
7     a = a * a % n;
8   }
9 }
```

Note that the pseudocode assumes arbitrary sized integers, \ll denotes bitwise shift left, $\&$ denotes bitwise and, and $\%$ denotes modulo.

The device does *not* tell Heidi the result of the computation. However, Heidi can measure how long does the computation take. She knows that only multiplication modulo n (lines 5 and 7 in the above pseudocode) takes any measurable amount of time, all other lines can be assumed to take 0 nanoseconds. Moreover, she knows that it takes $(\text{bits}(x) + 1) \cdot (\text{bits}(y) + 1)$ nanoseconds to multiply x by y modulo n , where $\text{bits}(x)$ is the number of bits in the binary representation of x without leading zeros, or more formally $\text{bits}(x) = \lceil \log_2(x + 1) \rceil$.

Heidi knows the integer n but does not know the integer d . She wants to find d by feeding the device different integers a as input and measuring the time the computation takes for each a .

She knows that n and d were chosen in the following way: first, two prime numbers p and q with 30 bits in binary representation (in other words, between 2^{29} and $2^{30} - 1$) were picked independently and uniformly at random. Then the number n was computed as $n = p \cdot q$. Then the number $m = \phi(n) = (p - 1) \cdot (q - 1)$ was computed. Then d was picked uniformly at random between 1 and $m - 1$ inclusive, such that it is coprime with m .

Interaction Protocol

First, the testing system writes the integer n — the modulo used by the device. Note that n and the hidden number d are guaranteed to have been generated according to the procedure described above.

Your solution shall print requests of two types:

- “? a ” tells to feed a as input to the device. a must be an integer between 0 and $n - 1$ inclusive. The testing system responds with the time it took the device to compute $\text{modPow}(a, d, n)$ in nanoseconds.
- “! d ” tells the value of d that your program has determined.

Don't forget to flush the output after each request!

Your solution must issue exactly one request of the second type, which must be the last request, and the solution must terminate gracefully after issuing it.

Your solution is allowed to issue at most 30 000 requests of the first type.

Your solution will be run on 30 testcases, working with one (n, d) pair per run. For each testcase the numbers n and d are fixed and were generated using the procedure described above. The example below was *not* generated in that manner and thus will *not* be used for testing your solution; it only serves to illustrate the input/output format and provide a sanity check for your calculation of the computation time.

Examples

input	output
15	? 3
980	? 8
293	! 5

Notes

In the first request in the example case, the following multiplications are done by the device when computing $\text{modPow}(3, 5, 15)$:

- $1 \cdot 3 \pmod{15} = 3$, taking 6 nanoseconds
- $3 \cdot 3 \pmod{15} = 9$, taking 9 nanoseconds
- $9 \cdot 9 \pmod{15} = 6$, taking 25 nanoseconds
- $3 \cdot 6 \pmod{15} = 3$, taking 12 nanoseconds
- $6 \cdot 6 \pmod{15} = 6$, taking 16 nanoseconds
- $6 \cdot 6 \pmod{15} = 6$, taking 16 nanoseconds
- $6 \cdot 6 \pmod{15} = 6$, taking 16 nanoseconds
- ... (55 more repetitions of the last multiplication)

The computation takes $6 + 9 + 25 + 12 + 58 \cdot 16 = 980$ nanoseconds.

A positive integer is *prime* if it has exactly two divisors: 1 and itself. Two positive integers are *coprime* if their greatest common divisor is 1.

Here are a few first values of the function $\text{bits}(x)$:

- $\text{bits}(0) = 0$
- $\text{bits}(1) = 1$
- $\text{bits}(2) = 2$
- $\text{bits}(3) = 2$
- $\text{bits}(4) = 3$
- ...

Problem I. Interactive Sort

Time limit: 10 seconds

Ivan wants to play a game with you. He took all integers from 1 to n inclusive, shuffled them and then put all even numbers into array e and all odd numbers into array o .

Your task is to find arrays e and o .

You can ask Ivan questions of certain kind. Each question consists of two integers i and j . For each question Ivan says whether $e[i] < o[j]$ or not.

You can ask at most 300 000 questions.

Interaction Protocol

First, the testing system writes the integer n ($1 \leq n \leq 10\,000$) — the number of integers Ivan used.

Your solution shall print requests of two types:

- “? i j ”. $1 \leq i \leq \lfloor \frac{n}{2} \rfloor, 1 \leq j \leq \lceil \frac{n}{2} \rceil$. The testing system responds with the symbol “<” if $e[i] < o[j]$ or with the symbol “>” otherwise.
- “! e_1 e_2 ... $e_{\lfloor \frac{n}{2} \rfloor}$ o_1 o_2 ... $o_{\lceil \frac{n}{2} \rceil}$ ” tells the values of e and o that your program has determined.

Don't forget to flush the output after each request!

Your solution must issue exactly one request of the second type, which must be the last request, and the solution must terminate gracefully after issuing it.

Your solution is allowed to issue at most 300 000 requests of the first type.

For each test case the number n is fixed and the numbers are shuffled using Java built-in shuffle function with fixed seed.

Examples

input	output
5	? 1 1
>	? 1 2
>	? 1 3
<	? 2 1
>	? 2 2
<	? 2 3
<	! 4 2 1 3 5

Problem J. Journey from Petersburg to Moscow

Time limit: 3 seconds

To conduct The World Programming Cup 2112 a network of wonderful toll roads was build in European part of Russia. This network consists of m bidirectional roads that connect n cities. Each road connects exactly two distinct cities, no two roads connect the same pair of cities, and it is possible to travel from any city to any other city using only this road network. Moreover, to ease the process of charging, no two roads intersect outside of the cities.

Each road is assigned some individual positive cost. Normally, if a driver makes a trip using some of these toll roads, at the end of his journey he would be charged the total cost equal to the sum of individual costs of all roads he has used. To increase the popularity of car travels between two capitals, the operator company Radishchev Inc introduced a special offer: make a journey from Saint Petersburg to Moscow and pay for only k most expensive roads along your path.

Formally, let some path consists of l roads. Denote as c_1 the cost of the most expensive road along this path, as c_2 the second most expensive and so on. Thus, we have a sequence $c_1 \geq c_2 \geq c_3 \geq \dots \geq c_l$ of individual costs of all roads along the chosen path. If $l \leq k$, then the path is too short and the driver pays the sum of all individual costs as usual, i.e. $\sum_{i=1}^l c_i$. If $l > k$, then the driver only pays for k most expensive roads, that is $\sum_{i=1}^k c_i$.

As the chief analyst of Radishchev Inc you were assigned a task to compute the cheapest possible journey from Saint Petersburg to Moscow.

Input

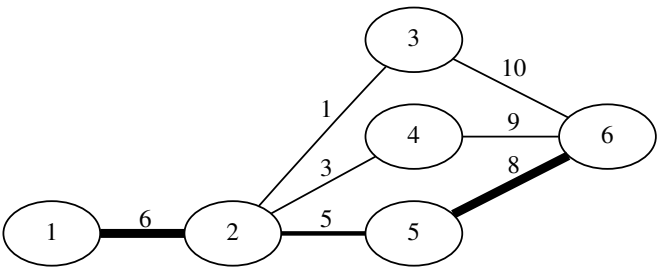
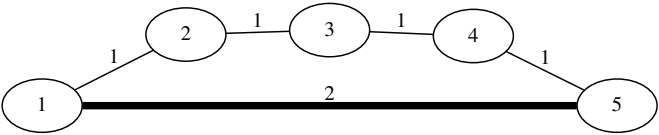
The first line of the input contains three integers n , m and k ($2 \leq n \leq 3000$, $1 \leq m \leq 3000$, $1 \leq k < n$) — the number of cities, the number of roads in the road network, and the maximum number of roads that one should pay for in a single journey.

Next m lines contain description of roads. Each road description contains three integers u_i , v_i , and w_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$, $1 \leq w_i \leq 10^9$) — the i -th bidirectional road that connects cities u_i and v_i with the cost of w_i for any direction. It is guaranteed that there is at most one road between each pair of cities and it is possible to get from any city to any other city using only these roads.

Output

Print one integer equal to the minimum possible cost of travel from the city number 1 (Saint Petersburg) to the city number n (Moscow).

Examples

input	output	illustration
<pre>6 7 2 1 2 6 2 3 1 2 4 3 2 5 5 3 6 10 4 6 9 5 6 8</pre>	14	
<pre>5 5 3 2 1 1 3 2 1 4 3 1 4 5 1 1 5 2</pre>	2	

Problem K. Knapsack Cryptosystem

Time limit: 3 seconds

The Merkle–Hellman Knapsack Cryptosystem was one of the earliest public key cryptosystems invented by Ralph Merkle and Martin Hellman in 1978. Here is its description.

Alice chooses n positive integers $\{a_1, \dots, a_n\}$ such that each $a_i > \sum_{j=1}^{i-1} a_j$, a positive integer q which is greater than the sum of all a_i , and a positive integer r which is coprime with q . These $n + 2$ integers are Alice's private key.

Then Alice calculates $b_i = (a_i \cdot r) \bmod q$. These n integers are Alice's public key.

Knowing her public key, Bob can transmit a message of n bits to Alice. To do that he calculates s , the sum of b_i with indices i such that his message has bit 1 in i -th position. This value s is the encrypted message.

Note that an eavesdropper Eve, who knows the encrypted message and the public key, has to solve a (presumably hard) instance of the knapsack problem to find the original message. Meanwhile, after receiving s , Alice can calculate the original message in linear time; we leave it to you as an exercise.

In this problem you deal with the implementation of the Merkle–Hellman Knapsack Cryptosystem in which Alice chose $q = 2^{64}$, for obvious performance reasons, and published this information. Since everyone knows her q , she asks Bob to send her the calculated value s taken modulo 2^{64} for simplicity of communication.

You are to break this implementation. Given the public key and an encrypted message, restore the original message.

Input

The first line contains one integer n ($1 \leq n \leq 64$).

Each of the next n lines contains one integer b_i ($1 \leq b_i < 2^{64}$).

The last line contains one integer $s \bmod q$ — the encrypted message s taken modulo q ($0 \leq s \bmod q < 2^{64}$).

The given sequence b_i is a valid public key in the described implementation, and the given value $s \bmod q$ is a valid encrypted message.

Output

Output exactly n bits (0 or 1 digits) — the original message.

Examples

input	output
5	01001
10	
20	
50	
140	
420	
440	

Problem L. Laminar Family

Time limit: 3 seconds

While studying combinatorial optimization, Lucas came across the notion of “laminar set family”. A subset family \mathcal{F} of some set Ω is called *laminar* if and only if it does not contain an empty set and for any two distinct sets $A, B \in \mathcal{F}$ it is correct that either $A \subset B$ or $B \subset A$ or $A \cap B = \emptyset$.

As an experienced problem setter Lucas always tries to apply each new piece of knowledge he gets as an idea for a programming competition problem. An area of his scientific interests covers *recognition problems* that usually sound like “Given some weird combinatorial property, check if the given structure satisfies it”.

Lucas believes that the perfect programming competition problem should contain ~~a cactus~~ a tree in it. Trying to put together laminar sets and trees into a recognition problem, he finally came up with the following problem: given an undirected tree on n vertices and a family $\mathcal{F} = \{F_1, \dots, F_k\}$ of sets, where F_i consists of all vertices belonging to the simple path between some two vertices a_i and b_i of the tree, check if the family \mathcal{F} is a laminar family. Note that in this case $\Omega = V$, and each $F_i \subseteq V$.

As you can see, Lucas had succeeded in suggesting this problem to the programming contest. Now it is up to you to solve it.

Input

The first line of the input contains two integers n and f ($1 \leq n, f \leq 100\,000$) — the number of vertices in the tree and the number of elements in a family \mathcal{F} .

Next $n - 1$ lines describe the tree structure. In the i -th line there are two integers u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) — the indices of the vertices that are connected by the i -th edge of the tree.

Next f lines describe the sets forming the family \mathcal{F} . In the i -th line there are two integers a_i and b_i ($1 \leq a_i, b_i \leq n$), such that F_i consists of all vertices belonging to the simple path between vertices a_i and b_i in the tree (including a_i and b_i).

Output

Output the only word “Yes” or “No” depending on whether or not the given set family is laminar.

Examples

input	output
4 2 1 2 2 3 2 4 1 2 4 2	No
6 5 1 2 2 3 3 4 5 6 5 2 2 1 6 6 1 4 3 4 4 1	Yes