Problem A. Chip Play

Input file:	`chip.in`
Output file:	`chip.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Consider the following game. We have a rectangular field $n \times m$ in size. Some squares of the field contain chips. Each chip has an arrow painted on it. Thus, each chip on the field points in one of the following directions: up, down, left or right. The player should choose a chip and make a move with it.

The move is the following sequence of actions. The chosen chip is marked as the current one. After that the player checks whether there are more chips in the same row (or in the same column) with the current one that are pointed by the arrow on the current chip. If there is at least one chip then the closest of them is marked as the new current chip and the former current chip is removed from the field. After that the process is repeated. This process can be repeated several times. If a new chip is not found, then the current chip is removed from the field and the player's move ends. By the end of a move the player receives several points equal to the number of the deleted chips.

Given initial chip arrangement, determine the maximum number of points that a player can receive during one move. Also determine the number of such moves.

Input

The first line contains two integers n and m ($n, m \geq 1, n \times m \leq 5000$). Then follow n lines containing m characters each, which forms the game field description. "`.`" means that this square is empty. "`L`", "`R`", "`U`", "`D`" mean that this square contains a chip and an arrow on it says left, right, up or down correspondingly. It is guaranteed that a field has at least one chip.

Output

Print two numbers — the maximum number of points a player can get after a move and the number of different moves that yield this maximum number of points.

Examples

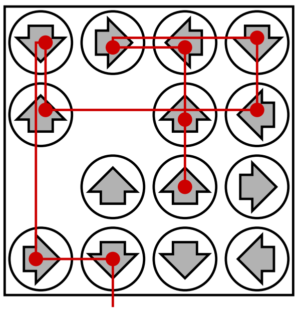
chip.in	chip.out

`4 4`
`DRLD`
`U.UL`
`.UUR`
`RDDL`
	`10 1`

`3 5`
`.D...`
`RRRLL`
`.U...`
	`6 2`

Note

In the first sample the maximum number of points is earned by the chip in the position $(3, 3)$. You can see its progress at the picture on the right. All other chips earn fewer points.



Problem B. Two Sets

Input file:	`twosets.in`
Output file:	`twosets.out`
Time limit:	1 second
Memory limit:	256 megabytes

Little X has n distinct integers: $p_1, p_2, \ldots, p_n$ and two more integers a and b. He wants to divide all of them into two sets A and B. The following two conditions must be satisfied:

- If the number x belongs to set A, then the number $a - x$ must also belong to set A.
- If the number x belongs to set B, then the number $b - x$ must also belong to set B.

Help Little X divide the numbers into two sets or determine that it is impossible.

Input

The first line contains three space-separated integers n, a, b ($1 \leq n \leq 10^5$; $1 \leq a, b \leq 10^9$). The next line contains n space-separated distinct integers $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq 10^9$).

Output

If there is a way to divide the numbers into two sets, then print "`YES`" in the first line. Then print n integers: $b_1, b_2, \ldots, b_n$ (b_i equals either 0, or 1), describing the division. If b_i equals 0, then p_i belongs to set A, otherwise it belongs to set B.

If it is impossible, print "`NO`" (without the quotes).

Examples

twosets.in	twosets.out

`4 5 9`
`2 3 4 5`
	`YES`
	`0 0 1 1`

`3 3 4`
`1 2 4`
	`NO`

Note

It's OK if all the numbers are in the same set, and the other one is empty.

Problem C. Context Advertising

Input file:	`context.in`
Output file:	`context.in`
Time limit:	2 seconds
Memory limit:	256 megabytes

Advertising has become part of our routine. And now, in the era of progressive technologies, we need your ideas to make advertising better!

In this problem we'll look at a simplified version of context advertising. You've got a text consisting of exactly n words. A standard advertising banner has exactly r lines, each line can contain at most c characters. The potential customer always likes it when they can see lots of advertising, so you should determine which maximum number of consecutive words from the text can be written on the banner. Single words in one line of the banner should be separated by spaces. You are allowed to insert more than one space at once. Note that you are not allowed to break the words, that is, each word in the text must occupy exactly one line in the banner. Besides, you cannot change the word order, that is, if you read the banner text consecutively, from top to bottom and from left to right, you should get some consecutive part of the advertisement text.

More formally, the statement can be written like that. Let's say that all words are indexed from 1 to n in the order in which they occur in the advertisement text. Then you have to choose all words, starting from some i-th one and ending with some j-th one ($1 \le i \le j \le n$), so that all of them could be written on the banner. There must be as many words as possible. See the samples for clarifications.

Input

The first input line contains three integers n, r, c ($1 \le n, r, c \le 10^6$; $r \times c \le 10^6$). The next line contains a text, consisting of n words. The words consist only of lowercase English letters and are not empty. The words in the lines are separated by single spaces. The total number of characters in all words doesn't exceed $5 \cdot 10^6$.

Output

Print at most r lines, in each line print at most c characters — the optimal advertisement banner. If there are multiple advertisement banners, print any of them.

Note that some lines of the banner **can be empty**. You are allowed not to print such lines.

Examples

context.in	context.in

`9 4 12`
`this is a sample text`
`for croc final round`
	`this is a`
	`sample text`
	`for croc`
	`final round`

`9 1 9`
`this is a sample text`
`for croc final round`
	`this is a`

`6 2 3`
`croc a a a croc a`
	`a a`
	`a`

`2 2 5`
`first second`
	`first`

Problem D. RMQ Inverse Problem

Input file:	`rmq.in`
Output file:	`rmq.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider an array of n elements. Let $Q(i, j)$ be the response to a query for finding the minimum among elements of the array from i to j inclusive. Your task is to restore an array given some queries and their responses.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 100\,000$) — the number of the elements in the array and the number of queries, respectively. Each of the following m lines contains a description of a query: three integers i, j and q denoting that $Q(i, j) = q$ ($1 \le i \le j \le n$, $-2^{31} \le q \le 2^{31} - 1$).

Output

If required array does not exist, output single line which contains the single word "`inconsistent`". In the other case the first line must contain the single word "`consistent`". The second line must contain n integers — elements of the required array. All integers must be in the interval from -2^{31} to $2^{31} - 1$ inclusive. If there are different solutions, output any of them.

Example

rmq.in	rmq.out

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

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

Problem E. Queue

Input file:	`queue.in`
Output file:	`queue.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

On a cold winter evening our hero Vasya stood in a railway queue to buy a ticket for Codeforces championship final. As it usually happens, the cashier said he was going to be away for 5 minutes and left for an hour. Then Vasya, not to get bored, started to analyze such a mechanism as a queue. The findings astonished Vasya.

Every man is characterized by two numbers: a_i, which is the importance of his current task (the greater the number is, the more important the task is) and number c_i, which is a picture of his conscience. Numbers a_i form the permutation of numbers from 1 to n.

Let the queue consist of $n-1$ people at the moment. Let's look at the way the person who came number n behaves. First, he stands at the end of the queue and the does the following: if importance of the task a_i of the man in front of him is less than a_n, they swap their places (it looks like this: the man number n asks the one before him: "Erm... Excuse me please but it's very important for me... could you please let me move up the queue?"), then he again poses the question to the man in front of him and so on. But in case when a_i is greater than a_n, moving up the queue stops. However, the man number n can perform the operation no more than c_n times.

In our task let us suppose that by the moment when the man number n joins the queue, the process of swaps between $n-1$ will have stopped. If the swap is possible it necessarily takes place.

Your task is to help Vasya model the described process and find the order in which the people will stand in queue when all the swaps stops.

Input

The first input line contains an integer n which is the number of people who has joined the queue ($1 \le n \le 10^5$). In the next n lines descriptions of the people are given in order of their coming — space-separated integers a_i and c_i ($1 \le a_i \le n$, $0 \le c_i \le n$). Every description is located on s single line. All the a_i's are different.

Output

Output the permutation of numbers from 1 to n, which signifies the queue formed according to the above described rules, starting from the beginning to the end. In this succession the i-th number stands for the number of a person who will stand in line on the place number i after the swaps ends. People are numbered starting with 1 in the order in which they were given in the input. Separate numbers by a space.

Examples

queue.in	queue.out

2
1 0
2 1
	2 1

3
1 3
2 3
3 3
	3 2 1

5
2 3
1 4
4 3
3 1
5 2
	3 1 5 4 2

Problem F. Reverse

Input file:	`reverse.in`
Output file:	`reverse.out`
Time limit:	5 seconds
Memory limit:	256 megabytes

Physical education teacher has learned to count the total height of all students staying in a row on the positions from l to r a long time ago. But children play a trick on him. At some moment in time children on position from l to r are swapped. The teacher noticed that they always "reverse" the segment. That means l-th student swaps with r-th student, $(l + 1)$-th student swaps with $(r - 1)$-th student and so on. He decided to not to scold them, but to count the total height on all scheduled intervals.

Input

The first line contains two integers n and m $(1 \le n, m \le 200\,000)$ — the number of children and the number of events. The second line contains n natural numbers — the heights of children in the order in the row. The height of every child does not exceed $2 \cdot 10^5$. Each of the following m line contains a description of an event: three numbers q, l, r $(0 \le q \le 1, 1 \le l \le r \le n)$. The number q shows a type of the event: 0 corresponds to a query to calculate the total height of children with numbers from l to r, 1 shows that children with numbers from l to r "reverse" their interval. All numbers in the input file are integers.

Output

For every event of kind 0 output a single number on the separate line — an answer for this query.

Example

reverse.in	reverse.out

5 6
1 2 3 4 5
0 1 5
0 2 4
1 2 4
0 1 3
0 4 5
0 3 5
	15
	9
	8
	7
	10

Problem G. Move To Front!

Input file:	`movetofront.in`
Output file:	`movetofront.out`
Time limit:	6 seconds
Memory limit:	256 megabytes

Corporal Ducar enjoys to give commands to his troop. His favorite command is "Move to front!". Corporal aligns his soldiers in a row and gives some commands, and each of them is formulated as "Soldiers from l to r, move to front!". Before Ducar gave the first command, the soldiers had been numbered from 1 to n, from left to right. When soldiers hear a command "Soldiers from l to r, move to front!", soldiers with numbers from l to r move to the beginning of the row in the same order in which they are.

For example, if at some moment in time the soldiers stand in an order $1, 3, 6, 2, 5, 4$, then after the command "Soldiers from 2 to 3, move to front!" the new order is $3, 6, 1, 2, 5, 4$. If soldiers from 3 to 4 are moved to front by next command of corporal, than new order is already so $1, 2, 3, 6, 5, 4$.

Given the sequence of the commands of the corporal, your task is to find the order of soldiers after all commands.

Input

The first line contains two integers n and m $(1 \le n \le 100\,000$, $1 \le m \le 100\,000)$ — the number of the soldiers and the number of commands, respectively. Each of the following m line contains description of the commands: two integers l_i and r_i $(1 \le l_i \le r_i \le n)$.

Output

The single line must contain n integers — the order of the soldiers after the execution of all commands.

Example

movetofront.in	movetofront.out

6 3
2 4
3 5
2 2
	1 4 5 2 3 6

Problem H. Bus

Input file:	`taxibus.in`
Output file:	`taxibus.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Usually the residents of Flatland use buses to travel in the city. Unfortunately, there is a problem — buses are designed in the way that passengers feel discomfort passing each other on the way to free seats.

Spiridon works as a bus driver for a long time. And he knows, that all passengers ride on a bus day by day. Therefore he decided to arrange places among passangers, in the way to minimize the number of times, when one passenger passes another to get to his place.

It is known when each passenger enters and leaves bus. All places are arranged in a row and are numbered from 1 to n starting from the closest seat to entry. When passenger sitting on the i-th place goes to the exit, he passes all passengers who are sitting on the seats with numbers less than i.

During the ride, passengers do not change their place during the ride.

Input

The first line contains single integer n ($1 \le n \le 100\,000$) — the number of passengers. Each of the following n lines contains two integers a_i, b_i — the indices of bus stops on which the i-th passenger enters and leaves bus, respectively. At each bus stop at most one person enters or leaves the bus.

Output

The first line must contain the single integer — the mimimal number of passes of passengers past each other. The second line must contain n integers — for each passenger output the place on which passenger should sit.

Example

taxibus.in	taxibus.out

2
1 4
2 3
	0
	2 1

5
1 8
3 6
2 4
9 10
5 7
	2
	10 2 4 1 1

Problem I. XOR on Segment

Input file:	`xor.in`
Output file:	`xor.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

You've got an array a, consisting of n integers $a_1, a_2, \ldots, a_n$. You are allowed to perform two operations on this array:

1. Calculate the sum of current array elements on the segment $[l, r]$, that is, count value $a_l + a_{l+1} + \cdots + a_r$.
2. Apply the xor operation with a given number x to each array element on the segment $[l, r]$, that is, execute $a_l = a_l \oplus x, a_{l+1} = a_{l+1} \oplus x, \ldots, a_r = a_r \oplus x$. This operation changes exactly $r - l + 1$ array elements.

Expression $x \oplus y$ means applying bitwise xor operation to numbers x and y. The given operation exists in all modern programming languages, for example in language *C++* and *Java* it is marked as "`^`", in *Pascal* — as "`xor`".

You've got a list of m operations of the indicated type. Your task is to perform all given operations, for each sum query you should print the result you get.

Input

The first line contains integer n ($1 \le n \le 10^5$) — the size of the array. The second line contains space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^6$) — the original array.

The third line contains integer m ($1 \le m \le 5 \cdot 10^4$) — the number of operations with the array. The i-th of the following m lines first contains an integer t_i ($1 \le t_i \le 2$) — the type of the i-th query. If $t_i = 1$, then this is the query of the sum, if $t_i = 2$, then this is the query to change array elements. If the i-th operation is of type 1, then next follow two integers l_i, r_i ($1 \le l_i \le r_i \le n$). If the i-th operation is of type 2, then next follow three integers l_i, r_i, x_i ($1 \le l_i \le r_i \le n, 1 \le x_i \le 10^6$). The numbers on the lines are separated by single spaces.

Output

For each query of type 1 print in a single line the sum of numbers on the given segment. Print the answers to the queries in the order in which the queries go in the input.

Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams, or the `%I64d` specifier.

Examples

xor.in	xor.out

5
4 10 3 13 7
8
1 2 4
2 1 3 3
1 2 4
1 3 3
2 2 5 5
1 1 5
2 1 2 10
1 2 3
	26
	22
	0
	34
	11

6
4 7 4 0 7 3
5
2 2 3 8
1 1 5
2 3 5 1
2 4 5 6
1 2 3
	38
	28

Problem J. Two Captains

Input file:	`twocaptains.in`
Output file:	`twocaptains.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

As it is known, Jack Sparrow and Barbossa are captains of The Black Pearl. The ship carries exactly n cannons, located in a row. During the battle, both captains give orders to their sailors simultaneously each minite. There are three different kinds of orders:

- `send` $l\,r$ — send his sailors to shoot from the cannons with numbers from l to r inclusive;
- `back` $l\,r$ — recall all his sailors from the cannons with numbers from l to r inclusive. If there are no sailors obeying this captain on some cannons from this segment, nothing happens;
- `rum` — bring one more bottle of rum.

Every order is executed immediately and battle continues one more minute until the next order. If at any moment of time sailors obeing different captains stay on the same cannon, they will fight and kill each other. This situation isn't suitable for both captains, so they ask you to help them with solution of this problem.

Before the start of another battle, captain Jack Sparrow and Barbossa make plans of their actions. The plan of captain Jack Sparrow consists of m_1 orders and the plan of Barbossa consists of m_2 orders. At the beginning of the i-th minute of battle, each captain gives his sailors the i-th order from his plan if the plan consists of at least i orders. Your task is to fix plans in the way all sailors will stay alive. The only available modification for you is to insert some orders `rum` in any places of plans. Since captains do not enjoy change plans, the total number of additional orders should be minimal.

Input

The first line contains single integer n ($1 \le n \le 10^9$) — the number of cannons on the ship.

The second line contains an integer m_1 ($1 \le m_1 \le 3\,000$) — the number of orders in the plan of Captain Jack Sparrow. Each of the following m_1 lines contains description of this orders. The orders are given as described above. For all instructions that used l, r $1 \le l \le r \le n$. It is guaranteed that the last order is `back` $1\,n$.

The next line contains an integer m_2 ($1 \le m_2 \le 3\,000$) — the number of orders in the plan of Barbossa. Each of the following m_2 lines contains description of this orders. The orders are given as described above. For all orders that used l, r $1 \le l \le r \le n$. It is guaranteed that the last order is `back` $1\,n$.

Output

The single line must contain the single integer — the minimal number of additional orders.

Example

twocaptains.in	twocaptains.out
3 4 send 1 1 send 2 2 back 1 1 back 1 3 5 send 2 3 send 1 1 back 2 2 rum back 1 3	3

Problem K. Rectangles

Input file:	`rects.in`
Output file:	`rects.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider n rectangles on the plane such that any two of them do not have common points. A rectangle B is *farther* than a rectangle A, if B's top left corner lies strictly below and strictly right than bottom right corner of A.

A sequence of rectangles $R_1, R_2, \ldots, R_k$ is called a *chain*, if for all i the rectangle R_i is farther than the rectangle R_{i-1}. *Weight* of chain is the sum of the numbers writen in rectangles of this chain.

Your task is to find the chain of rectangles with maximal possible weight.

Input

The first line contains single integer n ($1 \le n \le 100\,000$) — the number of the rectangles. Let x-axis goes from left to right and y-axis goes from down to up. Each of the following n lines contains five integers — coordinates $x_{i,1}, y_{i,1}$ of left bottom corner, coordinates $x_{i,2}, y_{i,2}$ top right corner and the number a_i writen in the i-th rectangle. All coordinates do not exceed 10^9 in absolute value. Numbers writen inside rectangles are positive and do not exceed 10^9. Every rectangle do not lie inside another rectangle.

Output

The first line must contain a single integer — the maximal possible weight of chain of rectangles. The second line must contain numbers of rectangles forming such chain in the corresponding order. If there are several optimal solutions, output any of them.

Example

rects.in	rects.out

4
1 1 2 2 6
3 1 4 2 5
0 3 1 4 5
5 1 6 2 4
	10
	3 2

Problem L. Windows 2

Input file:	`windows2.in`
Output file:	`windows2.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider rectangular windows on the screen with sides parallel to the coordinate axis. Some of them can overlap. Your task is to find area covered by this windows.

Input

The first line contains single integer n ($1 \le n \le 50\,000$) — the number of windows. Each of the following n lines contains coordinates of windows $x_{(1,i)}, y_{(1,i)}, x_{(2,i)}, y_{(2,i)}$, where $(x_{(1,i)}, y_{(1,i)})$ are coordinates of the left top corner, and $(x_{(2,i)}, y_{(2,i)})$ are coordinates of the right bottom corner of the i-th window (on the computer screen x coordinate grows down and y coordinate grows from left to right). All coordinates are integers and their absolute values do not exceed 10^9.

Output

The single line must contain the single integer — the area covered by windows.

Example

windows2.in	windows2.out

2
0 0 3 3
1 1 4 4
	14

Problem M. Lucky Array

Input file:	`lucky.in`
Output file:	`lucky.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

*Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

Petya has an array consisting of n numbers. He wants to perform m operations of two types:

- *add l r d* — add an integer d to all elements whose indexes belong to the interval from l to r, inclusive ($1 \le l \le r \le n, 1 \le d \le 10^4$);
- *count l r* — find and print on the screen how many lucky numbers there are among elements with indexes that belong to the interval from l to r inclusive ($1 \le l \le r \le n$). Each

lucky number should be counted as many times as it appears in the interval.

Petya has a list of all operations. The operations are such that after all additions the array won't have numbers that would exceed 10^4. Help Petya write a program that would perform these operations.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 10^5$) — the number of numbers in the array and the number of operations correspondingly. The second line contains n positive integers, none of which exceeds 10^4 — those are the array numbers. Next m lines contain operations, one per line. They correspond to the description given in the statement.

It is guaranteed that after all operations are fulfilled each number in the array will not exceed 10^4.

Output

For each operation of the second type print the single number on the single line — the number of lucky numbers in the corresponding interval.

Examples

lucky.in	lucky.out
3 6 2 3 4 count 1 3 count 1 2 add 1 3 2 count 1 3 add 2 3 3 count 1 3	1 0 1 1
4 5 4 4 4 4 count 1 4 add 1 4 3 count 1 4 add 2 3 40 count 1 4	4 4 4

Note

In the first sample after the first addition the array will look in the following manner:

```
4 5 6
```

After the second addition:

```
4 8 9
```

The second sample after the first addition:

```
7 7 7 7
```

After the second addition:

```
7 47 47 7
```

Problem N. Shooting Gallery

Input file:	`shooting.in`
Output file:	`shooting.out`
Time limit:	5 seconds
Memory limit:	256 megabytes

Berland amusement park shooting gallery is rightly acknowledged as one of the best in the world. Every day the country's best shooters master their skills there and the many visitors compete in clay pigeon shooting to win decent prizes. And the head of the park has recently decided to make an online version of the shooting gallery. During the elaboration process it turned out that the program that imitates the process of shooting effectively, is needed. To formulate the requirements to the program, the shooting gallery was formally described. A 3D Cartesian system of coordinates was introduced, where the X axis ran across the gallery floor along the line, along which the shooters are located, the Y axis ran vertically along the gallery wall and the positive direction of the Z axis matched the shooting direction. Let's call the XOY plane a shooting plane and let's assume that all the bullets are out of the muzzles at the points of this area and fly parallel to the Z axis. Every clay pigeon can be represented as a rectangle whose sides are parallel to X and Y axes, and it has a positive z-coordinate. The distance between a clay pigeon and the shooting plane is always different for every target. The bullet hits the target if it goes through the inner area or border of the rectangle corresponding to it. When the bullet hits the target, the target falls down vertically into the crawl-space of the shooting gallery and cannot be shot at any more. The targets are tough enough, that's why a bullet can not pierce a target all the way through and if a bullet hits a target it can't fly on. In input the simulator program is given the arrangement of all the targets and also of all the shots in the order of their appearance. The program should determine which target was hit by which shot. If you haven't guessed it yet, you are the one who is to write such a program.

Input

The first line contains an integer n ($1 \leq n \leq 10^5$) — the number of targets. Each of the subsequent n lines contains the description of a target. The target is described by five integers x_l, x_r, y_l, y_r, z, that determine it's location in space ($0 \leq x_l < x_r \leq 10^7, 0 \leq y_l < y_r \leq 10^7, 0 < z \leq 10^7$). The next line contains an integer m ($1 \leq m \leq 10^5$), determining the number of shots. Then in m lines shots are described. Every shot is determined by the coordinates of a bullet on the shooting plane (x, y) ($0 \leq x, y \leq 10^7$, the coordinates of bullets are integers). The shots are given in the order of their firing. The intervals between shots are large enough, and a target falls very quickly, that's why assume that a falling target can not be an obstruction for all the shots following the one that hit it.

Output

For every shot in the single line print the number of the target which the shot has hit, or 0, if the bullet did not hit any target. The targets are numbered starting from 1 in the order in which they were given in the input data.

Examples

shooting.in	shooting.out

2
1 4 1 4 1
2 5 2 6 2
4
0 0
3 3
4 5
3 5
	0
	1
	2
	0

Problem O. Edges in MST

Input file:	`mst.in`
Output file:	`mst.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given a connected weighted undirected graph without any loops and multiple edges.

Let us remind you that a graph's *spanning tree* is defined as an acyclic connected subgraph of the given graph that includes all of the graph's vertexes. The *weight* of a tree is defined as the sum of weights of the edges that the given tree contains. The *minimum spanning tree* (**MST**) of a graph is defined as the graph's spanning tree having the minimum possible weight. For any connected graph obviously exists the minimum spanning tree, but in the general case, a graph's minimum spanning tree is not unique.

Your task is to determine the following for each edge of the given graph: whether it is either included in **any** MST, or included **at least in one** MST, or **not included in any** MST.

Input

The first line contains two integers n and m ($2 \le n \le 10^5$, $n-1 \le m \le min(10^5, \frac{n(n-1)}{2})$) — the number of the graph's vertexes and edges, correspondingly. Then follow m lines, each of them contains three integers — the description of the graph's edges as "a_i b_i w_i" ($1 \le a_i, b_i \le n, 1 \le w_i \le 10^6, a_i \ne b_i$), where a_i and b_i are the numbers of vertexes connected by the i-th edge, w_i is the edge's weight. It is guaranteed that the graph is connected and doesn't contain loops or multiple edges.

Output

Print m lines — the answers for all edges. If the i-th edge is included in any MST, print "`any`"; if the i-th edge is included at least in one MST, print "`at least one`"; if the i-th edge isn't included in any MST, print "`none`". Print the answers for the edges in the order in which the edges are specified in the input.

Examples

mst.in	mst.out

4 5
1 2 101
1 3 100
2 3 2
2 4 2
3 4 1
	none
	any
	at least one
	at least one
	any

3 3
1 2 1
2 3 1
1 3 2
	any
	any
	none

3 3
1 2 1
2 3 1
1 3 1
	at least one
	at least one
	at least one

Note

In the second sample the MST is unique for the given graph: it contains two first edges.

In the third sample any two edges form the MST for the given graph. That means that each edge is included at least in one MST.