Problem A. A + B

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

This is an interactive problem. Your solution must strictly follow the interaction protocol described below.

Output the sum of the numbers a and b.

The numbers a and b are chosen by the interactor and are not revealed to you. To obtain any information about the numbers a and b, you can send any non-negative integer x to the interactor, after which the interactor will return a number equal to

$$(a \,\&\, b \,\&\, x) + (a \,|\, b \,|\, x) + (a \oplus b \oplus x)$$

Here, &, | and $\oplus$ denote the bitwise operations "and", "or" and "exclusive or", respectively. Thus, for each of your requests, the interactor will inform you of the sum of three quantities obtained from a, b and your number using each of these operations.

By making no more than 5 requests to the interactor, find and output the value of $a + b$.

Interaction Protocol

The interactor chooses two non-negative integers a and b ($0 \le a, b \le 10^9$).

Your program can send requests to the interactor in the format "`?` x" ($0 \le x \le 10^{12}$). In response to each request of this form, the interactor will inform your program of the number $(a \,\&\, b \,\&\, x) + (a \,|\, b \,|\, x) + (a \oplus b \oplus x)$.

If the value of x in the request does not fit within the specified constraints, or if the limit of 5 requests per test is exceeded, the interactor will output "`-1`" (without quotes) and terminate, and your solution will receive a verdict of `WA` (Wrong Answer). Upon reading "`-1`", your program should terminate, otherwise an incorrect verdict of `TL` (Time Limit Exceeded) may be obtained.

If your program is ready to answer the problem, it should output a request in the format "`!` x", where x is the presumed value of the sum $a + b$. Upon reading such a request, the interactor will issue a verdict of `WA` or `OK` depending on the correctness of the answer and terminate. Accordingly, after outputting such a request, your program should also terminate.

After each request, it is necessary to flush the output buffer (`cout.flush()` in C++, `sys.stdout.flush()` in Python) and output a newline. After each response to a request, the interactor also outputs a newline. The limit of 5 requests does not include requests of the second type, i.e. it is possible to make 5 requests in the format "`?` x" and 1 request in the format "`!` x".

Example

standard input	standard output
3	? 1 ! 2

Note

In the example from the statement, a request is made for $x = 1$, to which the interactor reports that $(a \,\&\, b \,\&\, 1) + (a \,|\, b \,|\, 1) + (a \oplus b \oplus 1) = 3$.

Based on this information, the solution assumes that $a = b = 1$. By simple enumeration of all a and b for which $a \,|\, b \,|\, 1 \leqslant 3$, it can be proven that other values of a and b cannot give the same sum, so the solution immediately outputs $1 + 1 = 2$ as the answer.

Problem B. Beautiful Dices

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

You are playing a game of "chaos dice". According to the rules of the game, players take turns rolling k-sided dice and write down the sequence of numbers a_i that appear on them (numbered from 1) until the sequence becomes of length n (n is odd).

Lets denote the central index of the sequence as c, i.e., $c = \frac{n+1}{2}$. You win only if the sequence satisfies the following three criteria:

1. the number k appears only once and is **exactly** in the middle of the sequence; more formally — $a_c = k$ and $a_i \neq k$ for all $i \neq c$;
2. the numbers on one side of the center at distances differing by a factor of two are the same; that is, for any d from $-\left\lfloor \frac{c-1}{2} \right\rfloor$ to -1 and from 1 to $\left\lfloor \frac{c-1}{2} \right\rfloor$, $a_{c+d} = a_{c+2d}$;
3. each of game master's m *favorite number pairs* (x_i, y_i) appears in the sequence as consecutive values no more than once.

In any other case, game master wins. It is worth noting that game master's favorite number pairs are ordered, so if game master likes the number pair (x_i, y_i), it is not guaranteed that he likes a pair (y_i, x_i).

Your task is to calculate the total number of sequences of numbers from 1 to k of length n that satisfy the described conditions. Calculate this number modulo $10^9 + 7$, as it may be too large.

Input

The first line of input contains three integers n, k, and m — the required number of dice rolls (length of the sequence), the number of sides on the die, and the number of game master's favorite number pairs ($1 \leq n \leq 53$; n is odd; $2 \leq k \leq 10$; $0 \leq m \leq 16$). It is guaranteed that $(k-1)^{n-1} \leq 10^{36}$.

In the i-th of the following m lines, a pair of numbers x_i and y_i is given — the i-th of game masters's favorite number pairs ($1 \leq x_i, y_i \leq k$). It is guaranteed that all pairs are distinct.

Output

Output a single integer — the number of sequences of length n that satisfy all the conditions.

Examples

standard input	standard output
3 6 1 6 6	25
5 8 1 1 2	49
7 5 2 1 2 2 1	254

Problem C. Historic Memories

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

Many decades have passed since the great war. Humans are not destined to live too long, so with the change of generations, even such events from the past are gradually forgotten.

You set out on a journey to visit the places where the team of heroes passed during their trip. The map of the continent is represented as a tree, meaning that there is a unique path between any two of the n cities. Traveling along each edge of the tree takes exactly one year.

In addition, each city is characterized by the value s_i — the level of *memory* of the events of those days. It is known that with each passing year, the level of memory in all cities decreases by 1. Sometimes events of the following types occur:

- "- t_i x_i" — at the beginning of year t_i, someone erects or restores a monument to the heroes and organizes an annual celebration in honor of the heroes' victory in city x_i. Then, starting from this year inclusive, the memory in this city stops decreasing.
- "+ t_i x_i" — at the beginning of year t_i, a monument is destroyed or the celebration is canceled in city x_i. Then, starting from this year inclusive, the level of memory in this city begins to decrease annually by 1 again.

The level of memory does not drop below 0 and can never increase. It is guaranteed that events of type '+' can occur only if before that the last event that occurred in the same city was of type '-'. Similarly, an event of type '-' cannot follow another event of type '-' in the same city.

You are interested in questions of the form "if at the beginning of year t'_i you set out on a journey from city x'_i, **and no more changes of type '+' or '-' will occur in the cities**, what is the maximum level of memory about the heroes can you encounter?". Find an answer for every such query.

Input

The first line of the input contains two integers n and q separated by a space — the number of cities on the continent and the number of queries ($1 \le n, q \le 10^5$).

The second line contains n integers s_i — the initial levels of memory in the cities at the beginning of year number 0 ($0 \le s_i \le 10^9$).

The next $n-1$ lines contain a description of the edges of the tree: the i-th line contains a pair of integers u_i and v_i — the numbers of cities connected by the i-th edge ($1 \le u_i, v_i \le n$). It is guaranteed that there is a unique path from any vertex to any other vertex.

The next q lines contain a description of the queries. Each query begins with the symbol '-', '+', or '?'. In the first two cases, two integers t_i and x_i follow the symbol, which mean "stop ('-') or resume ('+') the process of decreasing the level of memory in city x_i, starting from year t_i inclusive". Otherwise, two integers t'_i and x'_i follow the symbol, representing the query "what is the maximum level of memory that can be encountered by leaving city x'_i at the beginning of year t'_i?" ($0 \le t_i, t'_i \le 10^9$; $1 \le x_i, x'_i \le n$).

The queries are listed in chronological order. It is guaranteed that queries of types '-' and '+' with the same city alternate, and the first in this sequence must be '-'.

Output

For each query of type '?', output the answer to it on a separate line. **Note that when answering such a query, you should not take into account** subsequent queries of types '-' and '+'.

Examples

standard input	standard output
3 9 5 7 4 1 2 1 3 - 0 3 ? 0 1 ? 0 2 ? 0 3 + 3 3 - 4 1 ? 5 1 ? 5 2 ? 5 3	6 7 5 1 2 2
5 9 5 7 4 0 0 1 2 1 3 3 4 3 5 - 0 3 ? 0 1 ? 0 2 ? 0 3 + 4 3 - 5 1 ? 5 1 ? 5 2 ? 5 3	6 7 5 2 2 3

Problem D. Doctor

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

A famous doctor arrived in a rather interesting city: it has a total of n squares, some of which are connected by two-way roads, and it is also known that there is a unique path from any square to any other.

Since this doctor is a quite legendary figure, especially well-known in this city, as soon as he decides to stop at a square, people from all over the city will start gathering at this square to ask for life advice and treatment.

Of course, the doctor knows that every city resident will come to him, and he also found out that there are exactly a_i people living on square number i. Stopping at square x, he will will receive all the people living on this square immediately without queue, and all the other people will have time to line up — one queue on each road leading to this square. Each person will join the queue that will be on the way from their square to square x.

The doctor believes that he should choose square x so that the size of the largest queue to him is as small as possible. Help him find such a square.

Input

The first line of input contains an integer n — the number of squares in the city ($1 \le n \le 100\,000$).

The next $n - 1$ lines contain integers u_i and v_i — the numbers of the squares connected by the i-th road ($1 \le u_i, v_i \le n$). It is guaranteed that there is a unique path between any two squares.

Output

Output a single number x — the number of the square that the doctor should choose to minimize the size of the largest queue. If there are multiple optimal answers, output any of them.

Examples

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

Problem E. Secure Prison

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

You are tasked with building a high-security prison cell, which should consist of a room located inside another room. The laws allow for the construction of rooms of strictly one of n types, where the i-th type is a rectangle with dimensions a_i from west to east and b_i from north to south (but not vice versa). A room of size $a_i \times b_i$ can be placed inside a room of size $a_j \times b_j$ if and only if $a_i \le a_j$ and $b_i \le b_j$. For safety reasons, the outer and inner rooms must be of different types (but not necessarily different sizes).

Of course, it is not true for any two rooms that one can fit inside the other. For example, rooms of sizes 4×6 and 5×3 cannot be placed one inside the other, as $4 < 5$, but $6 > 3$. Therefore, if the warden insists that the inner room must be of type i and no other, you are allowed to make an exception and choose any type $j \ne i$ and "expand" the room $a_j \times b_j$ to $a \times b$ to make it an outer room (that is, choose arbitrary $a \ge a_j$, $b \ge b_j$) paying the fee of $(a + b) - (a_j + b_j)$ in order for the room of type i to fit inside it.

Clearly the warden wants to spend as little money as possible but he does not want to risk safety. Therefore, before making a final choice, he wants to estimate the minimum amount he will have to pay to the government if the inner room is of type i, for each i from 1 to n.

Input

The first line of input contains a single integer n — the number of types of rooms officially allowed in the city ($2 \le n \le 2 \cdot 10^5$).

In each of the following n lines, the integers a_i and b_i are given representing the dimensions of the i-th type of room ($1 \le a_i, b_i \le 10^9$). It is not guaranteed that all types correspond to different room sizes. If there are two types with the same dimensions, rooms of these types can fit inside each other.

Output

Output n integers separated by spaces, where the i-th integer is the minimum amount of money that must be paid to build a high-security cell with an inner room of type i.

Examples

standard input	standard output
3 1 3 2 2 3 1	1 1 1
5 4 4 6 3 3 3 2 5 1 1	1 2 0 1 0

Problem F. Mixing Drinks

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

After a tough contest, it can be helpful to drink some coffee. Real programmers have many different types of coffee that are gradually added to one cup. We will consider coffee with a specific feature: different types of coffee are layered in the cup and do not mix.

A drink made from such types of coffee can be described as follows: there are a total of n types of coffee in the cup, the i-th type is characterized by its *strength level* p_i and the *height of the layer* it occupies in the cup, h_i. If $i < j$, then the layer of the i-th type of coffee is below the coffee of the j-th type. It is also known that the height of the cup is equal to $\sum_{i=1}^{n} h_i$, meaning the top edge of the uppermost layer of coffee is exactly at the upper boundary of the cup.

Sometimes you want to create a drink with a specific total strength level. The total strength level is defined as the weighted average of the levels of the types of coffee poured into the cup, that is,

$$P = \frac{\sum_{i=1}^{n} p_i \cdot h_i}{\sum_{i=1}^{n} h_i}.$$

To change P, one can:

1. choose a straw of arbitrary height h;
2. perform the following one or more times: put the straw into the drink to any depth from 0 to h inclusive relative to the top edge of the cup (not relative to the current liquid level) and sip an arbitrary (not necessarily whole) amount of coffee from the level where the bottom end of the straw reaches.

When sipping some amount of coffee from one layer, the height of that layer decreases by the corresponding amount, and all upper layers drop down by the same amount.

Your task is to answer queries of the form: can the current drink be turned into a drink with strength t_i, and if so, what is the minimum height of the straw required for this? Since it may be difficult to calculate the exact necessary height of the straw, it is sufficient to determine the minimum number of upper layers of coffee needed so that by sipping some amount of coffee from some of them, it is possible to achieve a total strength level of t_i.

Input

The first line of input contains two integers n and q — the number of layers of coffee in the cup and the number of queries ($1 \le n, q \le 2 \cdot 10^5$).

The next n lines contain two integers p_i and h_i each — the strength level and height of the i-th layer of coffee **from the bottom** of the cup ($1 \le p_i, h_i \le 10^9$). It is guaranteed that the sum $p_i \cdot h_i$ over all i does not exceed 10^{18}.

In the i-th of the next q lines, there is a single integer t_i, defining the i-th query ($1 \le t_i \le 10^9$).

Output

Output q lines, in the i-th of which there is a single integer from 0 to n — the answer to the i-th query. If for some query the answer is such that the required strength level cannot be achieved, output -1 as the answer to that query.

Example

standard input	standard output
3 4 1 1 3 7 2 4 1 2 3 4	2 2 3 -1

Note

For the example from the statement:

1. In the first query, to obtain a drink with strength 1, it is sufficient to sip the top two layers of coffee.
2. In the second query, it is sufficient to sip some coffee from the second layer from the top.
3. In the third query, it will be necessary to sip from the first and third layers of coffee.
4. In the fourth query, it is impossible to achieve a strength level of 4.

Problem G. Crazy Arrangements

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You have a tree where m simple paths are chosen: (u_1, v_1), (u_2, v_2), ..., (u_m, v_m) — each path is determined by two vertices u_i and v_i, the start and the end. All paths have non-zero length, meaning $u_i \neq v_i$.

Some jokester wants to assign weights to edges each being either 0 or 1. Let's consider s_i a sum of all weights of the edges along the i-th path modulo 2 (in other words, xor of all weights along this path). The jokester calls an arrangement of weights on the edges *crazy* if the following inequality is correct: $s_i \leq s_{i+1}$ for all $1 \leq i < m$.

Your task is to calculate the number of *crazy* arrangements of weighs. Because the arrangements are crazy, you have to find the answer by modulo 998 244 353.

Input

The first line contains two integers n and m — the number of vertices in the tree and the number of chosen paths ($2 \leq n, m \leq 250\,000$).

The second line contains $n-1$ integers p_i denoting that there is an edge in the tree connecting the vertices p_i and $i+1$ ($1 \leq p_i < i+1$).

The next m lines contain two integers u_i and v_i each — the start and the end of the i-th path ($1 \leq u_i < v_i \leq n$).

Output

Output one number — the number of crazy arrangements of weights on the edges by modulo 998 244 353.

Examples

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

Problem H. Mood Balance

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

Autumn is a time of changeable moods. Some people are sad about the departing summer, some are happy about the slowly approaching winter, while others simply stroll peacefully and enjoy the autumn freshness and coolness.

When you walk through the autumn alley, you are often visited by deep thoughts about the meaning of life. Every minute, a new thought comes to you, and and you can choose whether it will be a negative thought or an ambiguous one (there are no positive thoughts). If your current mood is represented by the number x and it is the i-th minute of the walk, then

- a negative thought decreases your mood by 1, making it $x - 1$;
- an ambiguous thought first changes Kostya's mood by $i-2$, and then doubles it, making it $2(x+i-2)$.

Since you enjoy tranquility you want your mood at the end of the walk to be the same as it was at the beginning — zero. You are also not very fond of negative thoughts, so you would like to plan your walk in such a way that there are as few of them as possible.

Plan your thoughts during the walk so that by the end of the journey your mood is 0, while minimizing the number of negative thoughts. Your initial mood is $x = 0$, and the minutes of the walk are numbered from 1 to n.

Input

The first and only line contains an integer n — the duration of the walk in minutes ($4 \le n \le 10^{18}$).

Output

In the first line, output an integer k — the minimum number of negative thoughts.

In the next line, output k integers from 1 to n in increasing order — the minutes at which you should choose negative thoughts.

Examples

standard input	standard output
5	2 1 4
8	1 4

Problem I. Delivery Robot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You have to operate a delivery robot.

The area where the robot will deliver orders is a plane. We will introduce coordinate axes on it: the OX axis is directed from left to right, and the OY axis is directed from bottom to top. A point with coordinates x and y will be denoted as (x, y).

Two radio towers are placed on the plane: the first radio tower is located at point $(0, 0)$ and the second radio tower is at point $(1, 0)$.

The delivery robot can only execute four commands.

1. Move from the current point p **to the first radio tower**, reach it, turn 90 degrees **to the left**, and move in the new direction a distance equal to the distance from p to the first radio tower. After executing this command, the robot will be at point q, which is point p rotated 90 degrees **clockwise** around point $(0, 0)$.

2. Move from the current point p **to the first radio tower**, reach it, turn 90 degrees **to the right**, and move in the new direction a distance equal to the distance from p to the first radio tower. After executing this command, the robot will be at point q, which is point p rotated 90 degrees **counterclockwise** around point $(0, 0)$.

3. Move from the current point p **to the second radio tower**, reach it, turn 90 degrees **to the left**, and move in the new direction a distance equal to the distance from p to the second radio tower. After executing this command, the robot will be at point q, which is point p rotated 90 degrees **clockwise** around point $(1, 0)$.

4. Move from the current point p **to the second radio tower**, reach it, turn 90 degrees **to the right**, and move in the new direction a distance equal to the distance from p to the second radio tower. After executing this command, the robot will be at point q, which is point p rotated 90 degrees **counterclockwise** around point $(1, 0)$.

Currently, the robot is at point (x_1, y_1), and you want to send it to point (x_2, y_2). He suspects that this may not always be possible, and sometimes it may take too long. Help Sam construct a sequence of commands for the robot, with a length of no more than 10^6, after which the robot will move from point (x_1, y_1) to point (x_2, y_2). If such a sequence of commands with a length of no more than 10^6 does not exist, report that.

Input

The first line contains two integers x_1 and y_1 — the coordinates of the robot's starting position ($-100,000 \le x_1, y_1 \le 100\,000$).

The next line contains two integers x_2 and y_2 — the coordinates of the point where the robot must end up after executing the commands ($-100\,000 \le x_2, y_2 \le 100\,000$). It is guaranteed that the starting point does not coincide with the destination point.

Output

If there is no sequence of commands with a length of no more than 10^6, output "-1". Otherwise, in the first line, output a positive integer k — the number of commands in your sequence ($k \le 10^6$). In the next line, output a string s, consisting of k digits 1, 2, 3, and 4 — the numbers of the commands in the order they are executed.

Note that you do not need to minimize the length of the command sequence. If there are multiple answers, you can output any.

Examples

standard input	standard output
0 1 1 -2	2 24
0 1 1 1	-1

Note

In the first example, after the first operation, the robot will be at point $(-1, 0)$, and after the second at point $(1, -2)$.

In the second example, the robot cannot reach the destination point.

Problem J. Pizza

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are going to bake a pizza for your m friends. There are n additional ingredients at your disposal, each of them can either be put into pizza or not. You may use all ingredients or even prepare a pizza without additional ingredients at all. Thus, there are 2^n possible pizza recipes.

Not just any pizza will make your friends happy, though. Every friend has prepared a wish list of the form "ingredient t should be included into the pizza" or "ingredient t shouldn't be included into the pizza". Your friends aren't too needy: any pizza which has at least one of friend's wishes satisfied will make that friend happy.

Calculate the number of ways you can bake the pizza to make all your friends happy. Since this number may be too large, output it modulo 998 244 353.

Input

The first line of the input contains two integers n and m — the number of ingredients and the number of your friends, respectively ($1 \le n \le 1000$; $1 \le m \le 20$).

Each of the next m lines corresponds to one of your friend and contains an integer a_i — the number of his wishes on the wish list, followed by a_i integers $b_{i,j}$ — the description of wishes on the list ($1 \le a_i \le 100$; $-n \le b_{i,j} \le n$; $b_{i,j} \ne 0$). If $b_{i,j}$ is positive, the i-th friend has a wish "ingredient $b_{i,j}$ should be included into the pizza"; if it's negative, the i-th friend has a wish "ingredient $-b_{i,j}$ shouldn't be included into the pizza".

Every ingredient occurs at most once in every list.

Output

Output the number of different pizzas making all friends happy, modulo 998 244 353.

Examples

standard input	standard output
4 3 2 1 3 3 2 -4 1 1 -2	5
68 1 1 -42	468704809

Note

In the first example, the following sets of ingredients will make all friends happy: (1), (3), (1, 3), (1, 4), (1, 3, 4).

In the second example, ingredient 42 shouldn't be included into the pizza, while all the other ingredients may be either included or not. The answer is equal to 2^{67} modulo 998 244 353.

Problem K. String Mutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

You have string p of length n, each of its characters is from '`a`' to '`z`', i.e. a lowercase Latin letter.

In one action you can:

- choose any number i from 1 to n and change the value of p_i to any other c from '`a`' to '`z`';
- choose two possible values c_1 and c_2 from '`a`' to '`z`' and replace the value of **all** characters equal to c_1 with c_2;
- check if strings formed by the characters of the string on positions from l_1 to r_1 and from l_2 to r_2 (where $l_1 \le r_1$ and $l_2 \le r_2$) are equal; i.e. that the following holds:

$$\begin{cases} r_1 - l_1 = r_2 - l_2 \\ p_{l_1+i} = p_{l_2+i} & \text{for all } 0 \le i \le r_1 - l_1 \end{cases}.$$

Determine for each query of the third type whether the corresponding strings are equal or not.

Input

The first line of input contains a string p of length n, consisting of lowercase Latin letters — the initial characters ($1 \le n \le 2 \cdot 10^5$).

The next line contains a single integer q — the number of actions that will be performed ($1 \le q \le 2 \cdot 10^5$).

In the following q lines, the descriptions of the queries are listed in the form:

- "`1` i c" — assign the value c to the i-th character of the string ($1 \le i \le n$; '`a`' $\le c \le$ '`z`');
- "`2` c_1 c_2" — replace all values of all characters in the string equal to c_1 with c_2 ('`a`' $\le c_1, c_2 \le$ '`z`');
- "`3` l_1 r_1 l_2 r_2" — check that substrings formed by segments $[l_1, r_1]$ and $[l_2, r_2]$ are equal ($1 \le l_1, r_1, l_2, r_2 \le n$; $l_1 \le r_1$; $l_2 \le r_2$).

Output

For each query of the third type, output on a separate line a string "`YES`" (without quotes) if corresponding substrings are equal, and "`NO`" otherwise.

Examples

standard input	standard output
aaabc 6 3 1 2 2 3 1 4 c 2 c a 3 1 4 2 5 1 3 x 3 1 4 2 5	YES YES NO
abracadabra 7 3 1 4 8 11 1 7 r 2 a c 2 b c 3 1 4 8 11 3 1 4 5 8 3 2 4 6 8	YES YES YES YES

Problem L. Magical Clock

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

Scientists have acquired very mysterious magical clock. The strange movement of the hands led them to believe that important information might be encoded in that clock.

The clock consists of a dial with $60 \cdot 12 \cdot t$ divisions, numbered from 0 to $720t - 1$ clockwise, as well as the minute and hour hands. Every $\frac{1}{t}$-th minute (let's call this time interval a *tick*), the minute hand moves 12 divisions, while the hour hand moves 1 division. Thus, on a regular clock with this mechanism, in one hour (that is, in 60 minutes) the minute hand completes a full circle, while the hour hand completes $\frac{1}{12}$ of a circle.

The peculiarity of this clock is that when at the next tick the minute hand is supposed to overtake or catch up with the hour hand, it teleports back to the start. Therefore, if the minute hand points to division number m, and the hour hand points to h, and the distance between them is $d = (h - m) \bmod (720t)$ divisions, then if $0 < d < 12$, after the next *tick*, the minute hand will be at the zero division (while the hour hand continues its movement).

Scientists have proposed q theories regarding the nature and capabilities of such clock, and to verify the i-th theory, one needs to learn how long it takes for the hands to transition from state $s_{i,1}$ to state $s_{i,2}$. Find the answer for each such query.

Input

The first line of input contains two integers t and q — the $\frac{1}{720}$-th number of divisions on the clocks and the number of queries ($1 \le t \le 1500$; $1 \le q \le 5 \cdot 10^5$).

The i-th of the following q lines contains the description of the i-th query which consists of four integers h_1, m_1, h_2, and m_2 — the numbers of the divisions to which the hour and minute hands point in the initial and final states, respectively ($0 \le h_{1,2}, m_{1,2} < 720t$).

Output

Output q lines — the answers to all of the scientists' queries, each on its own line.

As an answer to the query, output the minimum number of *ticks* after which the clock will transition from state (h_1, m_1) to state (h_2, m_2), or "`-1`" (without quotes) if the clock will never reach the second state starting from the specified initial state.

Examples

standard input	standard output
1 4 0 0 1 12 0 0 60 0 11 0 12 0 12 0 13 0	1 60 1 -1
2 5 1201 1317 588 396 658 1196 102 84 442 8 1246 1200 79 0 739 0 355 286 98 72	827 -1 -1 660 5503