

Problem A. Another GCD Problem

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 second
 Memory limit: 1024 mebibytes

Consider an infinite multiset s containing elements $a_1, a_2, \dots, a_n$, each in infinite quantity. Based on this, we define the function $f(k)$ as follows. For each sub-multiset s' of s with exactly k elements, calculate the sum of the elements in s' . The value of $f(k)$ is then the greatest common divisor (GCD) of all these sums. Formally,

$$f(k) = \gcd \left(\sum_{\substack{s' \subseteq s \\ |s'|=k}} x \right).$$

For example, for $a = [3, 6]$, we have

$$f(2) = \gcd(3 + 3, 3 + 6, 6 + 6) = 3.$$

Find the maximum value of $f(k)$ and the smallest k that achieves this maximum value. In particular, if the maximum value does not exist, print “infinite”.

Input

The first line of input contains an integer t ($1 \leq t \leq 4 \cdot 10^5$), the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 10^5$), representing the number of distinct numbers in S .

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^{18}$), representing the elements in S .

The sum of n over all test cases is at most $4 \cdot 10^5$.

Output

For each test case, if the maximum value of $f(k)$ exists, output a line with two integers, representing the maximum value of $f(k)$ and the smallest k that achieves this maximum value.

Otherwise, output “infinite” (without quotes).

Examples

<i>standard input</i>	<i>standard output</i>
2	3 1
2	infinite
3 6	
2	
2 2	
2	3 3
3	6 3
1 4 7	
4	
4 16 28 34	

Note

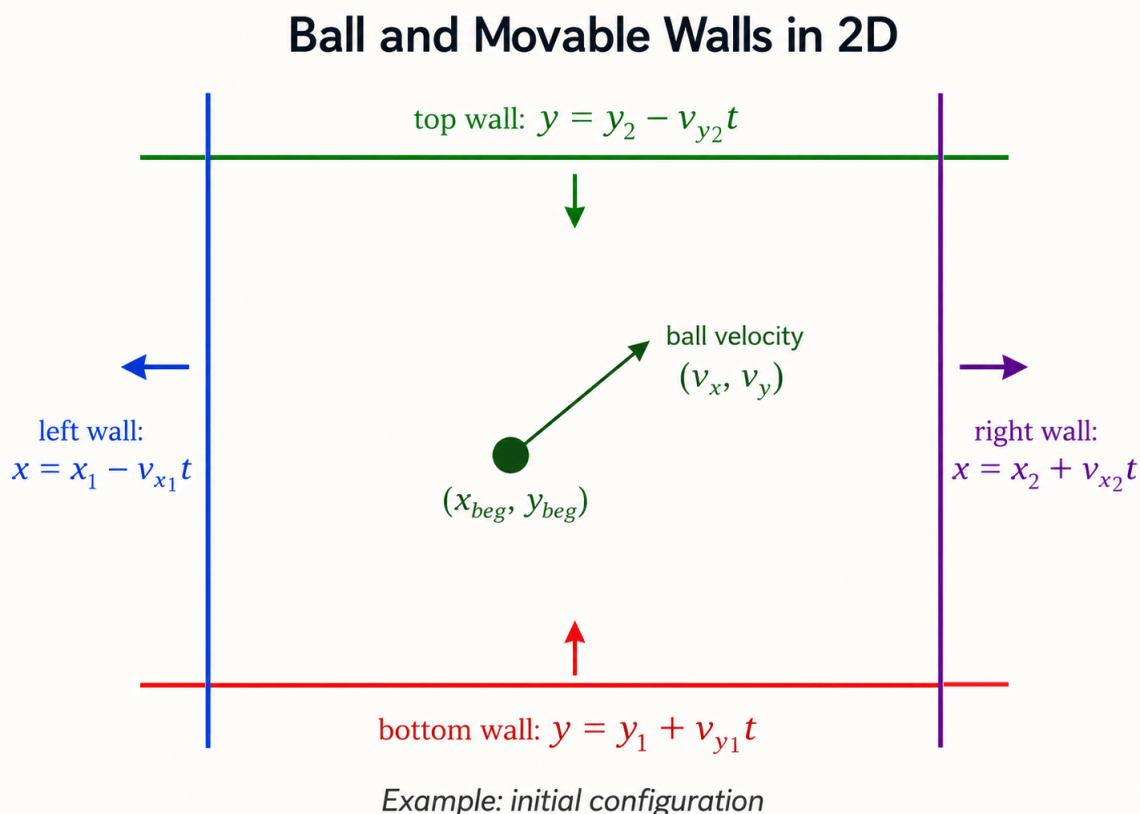
For the first test case in the first example, it can be observed that regardless of the value of k , $f(k) = 3$.
For the second test case in the first example, we have $f(k) = 2k$, thus f will grow infinitely and the maximum value does not exist.

Problem B. Ball And Movable Walls

Input file: *standard input*
Output file: *standard output*
Time limit: 1 seconds
Memory limit: 1024 mebibytes

In a two-dimensional experimental field, there are four infinitely long straight lines acting as movable “walls”: the top and bottom are horizontal lines, while the left and right are vertical lines. We place a negligibly small ball between the four lines, and it starts moving in a uniform straight line. Whenever the ball touches one of the walls, it rebounds in the direction normal to that wall (that is, only the sign of the velocity in that direction changes), while the velocity in the other cardinal direction remains unchanged. All the while, the left and right vertical lines move away from each other, whereas the top and bottom horizontal lines move towards each other, eventually converging at some moment to trap the ball.

Given the initial position and velocity of the ball, as well as the initial positions and velocities of the four lines, you need to determine the coordinates of the ball at the moment the top and bottom lines meet.



Here is a formal description of the experiment.

The initial position of the ball is denoted as (x_{beg}, y_{beg}) , and its initial velocity is denoted as (v_x, v_y) .

The bottom horizontal line and the top horizontal line are initially located at $y = y_1$ and $y = y_2$, respectively. In time t , they move to $y = y_1 + v_{y_1}t$ and $y = y_2 - v_{y_2}t$.

The left vertical line and the right vertical line are initially located at $x = x_1$ and $x = x_2$, respectively. In time t , they move to $x = x_1 - v_{x_1}t$ and $x = x_2 + v_{x_2}t$.

It is guaranteed that the ball is initially strictly inside the rectangle formed by the four lines, that is, $x_1 < x_{beg} < x_2$ and $y_1 < y_{beg} < y_2$.

Each collision occurs when the ball is moving faster than the wall in the corresponding direction. We guarantee that by ensuring that the ball's vertical velocity is always greater than the speeds of the two

horizontal lines, that is, $v_{y1} < |v_y|$ and $v_{y2} < |v_y|$.

To ensure that the top and bottom lines will definitely meet, we guarantee that the speeds of the two horizontal lines are not both 0.

The specific rules for collisions are as follows:

1. The ball moves in a uniform straight line when not in contact with any line.
2. When in contact with a horizontal line, only the vertical velocity is inverted (changing v_y to $-v_y$), while the horizontal velocity remains unchanged.
3. When in contact with a vertical line, only the horizontal velocity is inverted (changing v_x to $-v_x$), while the vertical velocity remains unchanged.
4. The determination of rebounds is independent of the translation speeds of the lines and only involves “inverting” the corresponding velocity components.

Input

The first line of input contains an integer t ($1 \leq t \leq 100$), the number of test cases. For each test case:

The first line contains four integers: x_{beg} , y_{beg} , v_x , and v_y ($-20 \leq v_x, v_y \leq 20$, $v_x \neq 0$, $v_y \neq 0$), representing the initial coordinates and velocity of the ball.

The second line contains four integers: y_1 , y_2 , v_{y1} , and v_{y2} ($0 \leq y_1 < y_{beg} < y_2 \leq 10^6$, $0 \leq v_{y1}, v_{y2} < |v_y|$, $v_{y1} + v_{y2} \neq 0$), representing the initial positions and velocities of the horizontal lines.

The third line contains four integers: x_1 , x_2 , v_{x1} , and v_{x2} ($0 \leq x_1 < x_{beg} < x_2 \leq 10^6$, $0 \leq v_{x1}, v_{x2} \leq 20$), representing the initial positions and velocities of the vertical lines.

Output

For each test case, output a line containing two space-separated real numbers x and y : the coordinates of the ball at the moment it stops.

Each of your answers will be considered correct if its absolute or relative error is at most 10^{-3} . In other words, if your answer is *out* and the true answer is *ans*, then your answer will be considered correct if

$$\frac{|out - ans|}{\max(1, |ans|)} \leq 10^{-3}.$$

Examples

<i>standard input</i>	<i>standard output</i>
1 8 5 3 7 0 20 2 3 0 30 1 1	20 8
2 108469 765432 20 20 4 999999 1 0 108464 108470 1 0 108470 765432 20 20 4 999999 1 0 108465 108471 1 0	-0.0789012981284893 999999 0.921098701871511 999999

Problem C. Circular Primes

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 1024 megabytes

A cyclic permutation of a given integer n is an operation in which its first digit (the leftmost one) is moved to the last position (the rightmost one).

For example, a cyclic permutation of the number 197 yields the number 971, and one more permutation yields the number 719.

Andrew noticed that for some numbers the following holds: they are prime numbers, and every cyclic permutation of them also yields a prime number. Such a number is called a *circular prime*.

For example, the number 197 is a circular prime.

In addition to being interested in these numbers, Martin also became interested in numbers that are **partially circular primes** — the number itself or at least one of its cyclic permutations is a prime number.

For example, the number 133 is a partially circular prime, because a cyclic permutation yields the number 331, which is prime, although 133 itself is not prime.

Andrew wants to know how many *circular primes* or *partially circular primes* lie in a given numerical interval from l to r inclusive. Since Andrew is very curious, he may ask several such queries at once.

Input

The first line of the input contains two integers n and c — the number of queries and the type of each query. ($1 \leq n \leq 1000$, $1 \leq c \leq 2$). $c = 1$ if the number of circular primes is required, and $c = 2$ if the number of partially circular primes is required.

Each of the following k lines contains two integers l and r ($1 \leq l \leq r < 10^6$) — the parameters of the next query (i.e., an answer is required for the interval from l to r inclusive).

Output

For each query, output a single integer on a new line — the answer to that query.

Examples

standard input	standard output
2 1 35 135 2 7	7 4
1 2 100 112	8

Note

In the first example, in the interval from 35 to 135, the number of circular primes is 7. These are the numbers 37, 7, 73, 79, 97, 113, and 131. In the interval from 2 to 7, their number is 4 (since all one-digit primes are circular).

In the second example, the partially circular primes are 101, 103, 104, 106, 107, 109, 110, and 112.

Problem D. Director and Excitement

Input file: *standard input*
Output file: *standard output*
Time limit: 3 seconds
Memory limit: 1024 mebibytes

The Cook Chicken Potato Contest is one of the most renowned events in the culinary world. The competition venue provides k cooking stations, and before the start, the participants will be divided into k equally sized teams by the Contest Director, known as Small Q.

To test the teamwork among participants and to add more excitement to the competition, Small Q wants to arrange the teams in such a way that the strength differences among participants in the same team are as large as possible. Let the strengths of the participants be denoted as $a_1, a_2, \dots, a_n$, and their respective team assignments as $t_1, t_2, \dots, t_n$. Small Q defines the excitement level of the competition as:

$$D = \min_{1 \leq i < j \leq n} \begin{cases} |a_i - a_j|, & t_i = t_j \\ +\infty, & t_i \neq t_j \end{cases}$$

Now, there are n potential participants. The possible strength of the i -th participant is described by an interval $[\ell_i, r_i]$, indicating that their actual strength during a certain competition could turn out to be any real number within that interval. The participants are sorted by strength in non-decreasing order, which means that, for all $1 \leq i < j \leq n$, we have $\ell_i \leq \ell_j$ and $r_i \leq r_j$.

Small Q has q competition plans, where the i -th plan is to invite participants with indices between L_i and R_i . You need to help Small Q determine whether he can arrange the teams in such a way that it would be possible for the excitement level of the competition to be at least D_i .

Input

The first line of input contains an integer t ($1 \leq t \leq 10^5$), the number of test cases. For each test case:

The first line contains two integers, n and k ($1 \leq n \leq 5 \cdot 10^5$, $1 \leq k \leq \min(5, n)$), representing the number of potential participants and the number of teams.

The next n lines contain two integers, ℓ_i and r_i ($0 \leq \ell_i \leq r_i \leq 10^{12}$) for the i -th participant, indicating the possible strength they can exhibit. For all $1 \leq i < n$, we have $\ell_i \leq \ell_{i+1}$ and $r_i \leq r_{i+1}$.

The next line contains an integer q ($1 \leq q \leq 10^5$), indicating the number of competition plans.

The next q lines contain three integers: L_i , R_i , and D_i ($1 \leq L_i \leq R_i \leq n$, $(R_i - L_i + 1)$ is divisible by k , and $0 \leq D_i \leq 10^{12}$), indicating that the i -th competition plan is to invite participants with indices from L_i to R_i , and Small Q expects the excitement level to be at least D_i .

The sum of n over all test cases is at most 10^6 . The sum of q over all test cases is at most 10^5 .

Output

For each test case, output q lines. For the i -th output, print “YES” or “NO” to indicate whether Small Q’s expectation for the i -th plan can possibly be met.

Example

<i>standard input</i>	<i>standard output</i>
2	YES
4 2	YES
1 1	YES
3 3	YES
4 4	NO
6 6	YES
3	NO
1 2 3	YES
3 4 2	NO
1 4 2	
5 1	
1 3	
2 3	
4 6	
7 10	
8 12	
6	
1 3 2	
1 3 3	
2 4 4	
2 4 5	
3 5 4	
3 5 5	

Problem E. Exposing The Foxes

Input file: **standard input**
Output file: **standard output**
Time limit: **1 second**
Memory limit: **1024 megabytes**

Photographer Gerald is planning to take a picture of a pack of foxes of a rare breed. The pack is on a clearing surrounded by coniferous forest on all sides. The camera is attached to a fir tree outside the clearing at the required height, after which it takes a single shot (if several shots are taken, there is a risk that the foxes will notice the presence). When setting up the camera, one can choose the orientation — vertical or horizontal — and the tree to which the camera is attached, so that the shooting direction coincides with a line chosen by Stierlitz, parallel to one of the coordinate axes.

There are n foxes on the clearing in total. Each of them can be represented as a segment. The shooting direction is represented by a line, that is, a fox will be captured in the picture if the line intersects the segment at least at one point (at an interior point, at one of the endpoints, or passes along the entire segment).

What is the maximum number of foxes that Gerald can capture in the picture if he chooses the shooting direction optimally?

Input

The first line of the input contains one integer n ($1 \leq n \leq 3 \cdot 10^5$). Each of the following n lines contains four integers x_a , y_a , x_b , and y_b — the coordinates of the beginning and the end of the segment corresponding to the next fox ($-10^8 \leq x_a, y_a, x_b, y_b \leq 10^8$). The segments may overlap, coincide, and may also have zero length.

Output

Print one integer — the maximum number of foxes that Gerald can capture in the picture.

Example

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

Problem F. Force Levels

Input file: **standard input**
Output file: **standard output**
Time limit: 2 seconds
Memory limit: 1024 megabytes

In a galaxy far, far away, a training camp for Jedi candidates is being planned.

Initially, the candidates are untrained, that is, their Force level is zero.

At the camp, the first half of each day is devoted to meditation. On some days, this leads to an increase in the Force level: on day d_i , the Jedi's Force level increases by p_i . The second half of the day is devoted to practicing combat skills: if at that moment the Jedi's Force level is positive, then the candidate practices one more combat skill, but the Force level decreases by 1. If the Force level is zero, then no skill practice takes place on that day.

On some day k , Master Yoda, who supervises the camp, announces that combat skills have been practiced enough, and starting from the next day, the second half of the day is devoted to rest.

Master Yoda knows on which days the Force level increases during the camp. He wants to develop q adapted versions of the training program, differing by different values of k . For each program, the Master is interested in the number of combat skills that the Jedi candidate will practice if combat skill practice stops after day k_j .

Input

The first line of the input contains two integers n and q — the number of days on which the Force level increases, and the number of program adaptations, respectively ($1 \leq n, q \leq 3 \cdot 10^5$).

Each of the next n lines contains two integers d_i and p_i — the day on which the Jedi receives a Force increase, and the value of that increase, respectively ($1 \leq d_1 < \dots < d_n \leq 10^{18}$, $1 \leq p_i \leq 10^9$).

Each of the next q lines contains one integer k_j — the last day of combat skill practice according to the j -th adaptation ($1 \leq k_1 < \dots < k_n \leq 10^{18}$).

Output

In the j -th of the q output lines, print one number — the answer to the problem for the j -th adaptation.

Examples

standard input	standard output
2 1 1 2 5 10 7	5
4 4 5 7 14 1000000000 15 1000000000 16 1000000000 3 8 11 1000000000000000000	0 4 7 3000000007

Problem G. Efficient Summation

Input file: *standard input*
 Output file: *standard output*
 Time limit: 13 seconds
 Memory limit: 1024 mebibytes

Let us denote the prime factorization of an integer y as:

$$y = \prod_{i=1}^k p_i^{\alpha_i}, \text{ where } p_1 < p_2 < \dots < p_k.$$

Now, given a sequence r of length m that does not contain duplicate elements, define $f(y)$ as:

$$f(y) = \prod_{i=1}^m p_{r_i} \cdot \alpha_{r_i}.$$

If $r_i > k$, consider $p_{r_i} \cdot \alpha_{r_i} = 1$.

You are given q queries. Each query provides an integer x in the form of its prime factorization. Your task is to compute the value of

$$\sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} f(i \cdot x) \bmod 2^{32}.$$

To reduce the output size, print the XOR sum of all answers.

Input

The first line of input contains three integers: n , m , and q ($1 \leq n \leq 7 \cdot 10^8$, $1 \leq m \leq 25$, $1 \leq q \leq 5 \cdot 10^5$), representing the upper limit for queries, the length of the sequence r , and the number of queries, respectively.

The second line contains m integers $r_1, r_2, \dots, r_m$ ($1 \leq r_i \leq 25$, all r_i are distinct).

The next q lines describe queries. Each query is given on a single line which starts with an integer ℓ indicating the number of terms in the prime factorization of x . It is followed by 2ℓ integers $p_1, \alpha_1, p_2, \alpha_2, \dots, p_\ell, \alpha_\ell$ (p_i are distinct primes in ascending order, $\alpha_i \geq 1$). This describes the value

$$x = \prod_{i=1}^{\ell} p_i^{\alpha_i}$$

such that $1 \leq x \leq n$. Specifically, $\ell = 0$ indicates that $x = 1$.

Output

Output a single line containing one integer: the XOR sum of all query answers.

Example

<i>standard input</i>	<i>standard output</i>
10 2 5 2 3 0 1 5 1 1 2 1 1 7 1 2 2 1 3 1	31

Problem H. How Many Rounds Are Expected?

Input file: *standard input*
Output file: *standard output*
Time limit: 1 seconds
Memory limit: 1024 mebibytes

There are n sets, and the i -th set contains a_i elements, with a total of $2m$ distinct elements ($\sum a_i = 2m$), where each element belongs to a unique set.

In each round of operations, we randomly match all elements into m pairs. For each matched pair, we randomly select one element, remove it from its original set, and place it into the set of the element it is matched with. The question is: what is the expected number of rounds until all elements belong to a single set?

Output the answer modulo 998 244 353.

Input

The first line of input contains an integer t ($1 \leq t \leq 100$), the number of test cases. For each test case:

The first line contains two integers n and m ($1 \leq n \leq 2m \leq 400$), representing the number of initial sets and the number of matches per round.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 2 \cdot m$, $\sum a_i = 2 \cdot m$), where a_i indicates that the i -th set has a_i elements.

The sum of n over all test cases is at most 800. The sum of m over all test cases is at most 400.

Output

For each test case, output a single line containing an integer: the answer modulo 998 244 353. Formally, the answer can be represented as a rational number p/q , and you have to print the value r such that $p \equiv q \cdot r \pmod{998\,244\,353}$.

Examples

<i>standard input</i>	<i>standard output</i>
4 2 3 2 4 2 3 3 3 3 3 1 2 3 4 3 1 1 2 2	5 499122182 748683271 7
3 6 5 4 2 1 1 1 1 7 7 1 4 2 3 2 1 1 8 10 6 3 1 4 2 2 1 1	341357698 460332729 534908689

Note

Explanation for the first example:

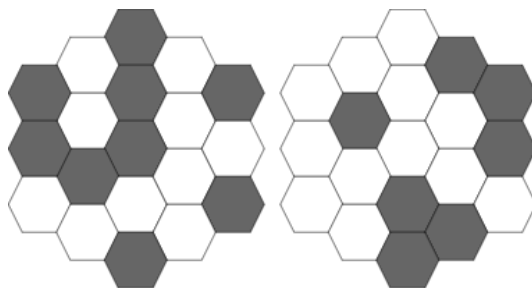
For each of the four test cases in this example, there are 6 elements. In one round, there are 15 possible matching ways, and for each matching there are $2^3 = 8$ ways to choose one element from every matched pair. Thus, there are 120 equally likely operations in one round. Let E_S denote the expected number of remaining rounds from state S .

- For the first test case: We represent the initial state as $[2, 4]$. Among the 120 operations, 24 operations result in the state $[6]$, while the remaining 96 operations keep the state $[2, 4]$. Therefore, $E_{[2,4]} = 1 + \frac{96}{120}E_{[2,4]} = 5$.
- For the second test case: We represent the initial state as $[3, 3]$. Among the 120 operations, 12 operations result in the state $[6]$, while the remaining 108 operations result in the state $[2, 4]$. Therefore, the expected value is $1 + \frac{108}{120} \cdot 5 = \frac{11}{2}$, which is 499 122 182 modulo 998 244 353.
- For the third test case: We represent the initial state as $[1, 2, 3]$. First, consider the auxiliary state $[2, 2, 2]$: among the 120 operations, 72 operations result in $[2, 4]$, while 48 operations keep the state $[2, 2, 2]$. Hence, $E_{[2,2,2]} = 1 + \frac{72}{120} \cdot 5 + \frac{48}{120}E_{[2,2,2]} = \frac{20}{3}$. For the state $[1, 2, 3]$, among the 120 operations, 6 operations result in $[6]$, 78 operations result in $[2, 4]$, and 36 operations result in $[2, 2, 2]$. Therefore, the expected value is $1 + \frac{78}{120} \cdot 5 + \frac{36}{120} \cdot \frac{20}{3} = \frac{25}{4}$, which is 748 683 271 modulo 998 244 353.
- For the fourth test case: We represent the initial state as $[1, 1, 2, 2]$. Among the 120 operations, 48 operations result in $[2, 4]$, while the remaining 72 operations result in $[2, 2, 2]$. Therefore, the expected value is $1 + \frac{48}{120} \cdot 5 + \frac{72}{120} \cdot \frac{20}{3} = 7$.

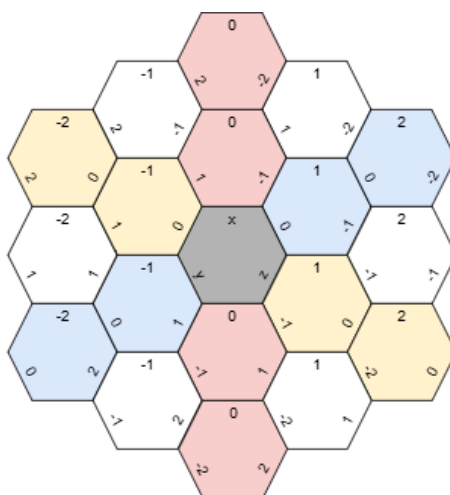
Problem I. Infinite Hexagonal Grids

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 seconds
 Memory limit: 1024 mebibytes

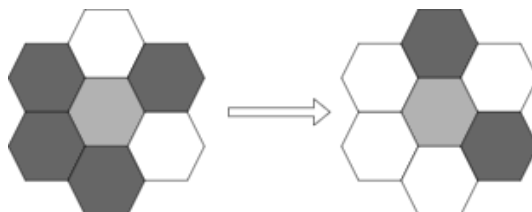
You are given two infinite hexagonal grids, where some grid cells are black and some grid cells are white. Two such grids are shown in the figure below (they correspond to the first example):



We describe each grid cell in the grid using a three-dimensional coordinate (x, y, z) such that $x, y, z \in \mathbb{Z}$ and $x + y + z = 0$, as illustrated in the figure below:



We can perform a flip operation: select a three-dimensional coordinate (x, y, z) such that $x, y, z \in \mathbb{Z}$ and $x + y + z = 0$, and then flip the colors of the grid cells surrounding that coordinate: black flips to white, and white flips to black. Specifically, we flip the colors of the grid cells $(x, y - 1, z + 1)$, $(x + 1, y - 1, z)$, $(x + 1, y, z - 1)$, $(x, y + 1, z - 1)$, $(x - 1, y + 1, z)$, and $(x - 1, y, z + 1)$. An example is shown in the figure below:



The question is whether it is possible to perform several flip operations on the first hexagonal grid so that the color of each grid cell matches that of the second hexagonal grid.

Input

The first line of input contains an integer t ($1 \leq t \leq 100$), the number of test cases. For each test case:

The first line contains two integers, n and m ($1 \leq n, m \leq 10^5$), representing the number of black grid cells on the two hexagonal grids.

The next n lines contain the coordinates of distinct black cells on the first hexagonal grid, where the i -th line contains three integers: x_i , y_i , and z_i ($-10^9 \leq x_i, y_i, z_i \leq 10^9$, $x_i + y_i + z_i = 0$).

The following m lines contain the coordinates of distinct black cells on the second hexagonal grid, where the i -th line contains three integers: u_i , v_i , and w_i ($-10^9 \leq u_i, v_i, w_i \leq 10^9$, $u_i + v_i + w_i = 0$).

The sum of n over all test cases is at most $2 \cdot 10^5$. The sum of m over all test cases is at most $2 \cdot 10^5$.

Output

For each test case, output “YES” if it is possible to perform several flip operations on the first hexagonal grid so that the color of each grid cell matches that of the second hexagonal grid; otherwise, output “NO”.

Examples

<i>standard input</i>	<i>standard output</i>
1 9 7 0 2 -2 -2 2 0 0 1 -1 2 0 -2 -1 0 1 2 -2 0 0 -2 2 0 0 0 -2 1 1 -1 1 0 1 1 -2 2 0 -2 2 -1 -1 0 -1 1 0 -2 2 1 -2 1	YES
2 4 2 -1 1 0 -1 0 1 0 -1 1 1 0 -1 0 1 -1 1 -1 0 4 3 -1 1 0 -1 0 1 0 -1 1 1 0 -1 0 0 0 0 1 -1 1 -1 0	YES NO

Problem J. Judge of Gensokyo

Input file: *standard input*
Output file: *standard output*
Time limit: 4 seconds
Memory limit: 1024 mebibytes



Image source: Bad Apple!! PV [Shadow Picture]

Shiki is the judge of Gensokyo and often needs to take notes with pen and paper. Shiki has discovered that many English letters, such as **ovw**, form ligatures when written. For example, two consecutive **vs** written together look like a **w**; consecutive **v** and **w** also connect, for instance, **wvwwvww** looks like a string of length 10 of **vs**.

Shiki considers a string to be good if and only if it is completely symmetrical when written on paper. For example, **wvowv** is symmetrical because it has three sharp angles, one round shape, and three sharp angles when written; however, **vowow** is not symmetrical because it has two sharp angles on the left and three on the right.

Now Shiki has given you a string s that she recorded, which consists only of the characters “**ovw**”. You need to find the longest substring of s such that the substring is good.

Input

The first line of input contains an integer t ($1 \leq t \leq 2 \cdot 10^6$), the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 10^7$), representing the length of the string s .

The second line contains a string s of length n , which consists only of the characters “**ovw**”.

The sum of n over all test cases is at most 10^7 .

Output

For each test case, output a line containing the longest good substring of s .

If there are multiple solutions, output any one of them.

Example

<i>standard input</i>	<i>standard output</i>
3	wwwowvvw
9	oooooooo
vwwowvvw	vvvvv
12	
vooooooooowvv	
12	
wwvovovvvvv	

Problem K. Knapsack And Greedy Algorithm

Input file: *standard input*
 Output file: *standard output*
 Time limit: 1 second
 Memory limit: 1024 mebibytes

The 01-knapsack problem is a classic combinatorial optimization problem in algorithmic competitions. Little W has learned a greedy algorithm to solve this problem. The 01-knapsack problem is defined as follows.

Consider n items: their weights are positive integers $w_1, w_2, \dots, w_n$, and their values are positive integers $v_1, v_2, \dots, v_n$. We have a knapsack with capacity c . We need to find $x_1, x_2, \dots, x_n$, each of them either 0 or 1, such that the sum of weights $\sum_{i=1}^n w_i \cdot x_i \leq c$ and the sum of values $v_t = \sum_{i=1}^n v_i \cdot x_i$ is the maximum possible.

Little W's greedy algorithm works as follows.

Sort the n items in descending order based on the value-to-weight ratio v_i/w_i ; when ratios are the same, sort by decreasing w_i . Initialize a variable c_0 to 0, and iterate i from 1 to n . If $c_0 + w_i \leq c$, set $x_i \leftarrow 1$ and $c_0 \leftarrow c_0 + w_i$; otherwise, set $x_i \leftarrow 0$. After iterating, you will obtain the desired $x_1, x_2, \dots, x_n$ and v_t .

You certainly know that this algorithm is incorrect, but Little W does not believe it. Even when you provide Little W with some counterexamples, Little W still believes that this algorithm can yield the optimal v_t for many different values of c . Therefore, provided with a capacity limit c_{lim} , you now wish to construct a list of weights $w_1, w_2, \dots, w_n$ and a list of values $v_1, v_2, \dots, v_n$ such that:

1. For all c such that $2 \leq c \leq c_{lim}$, Little W's algorithm cannot achieve the optimal v_t .
2. Under condition 1, n is as small as possible.
3. Under conditions 1 and 2, $\max(w_1, w_2, \dots, w_n)$ is as small as possible.
4. Under conditions 1, 2, and 3, $\max(v_1, v_2, \dots, v_n)$ is as small as possible.

Now you need to construct a set that satisfies the above conditions for the 01-knapsack to convince Little W. Can you do it? If there are multiple construction methods, you can output any one of them.

Input

The input consists of a single line containing an integer c_{lim} ($2 \leq c_{lim} \leq 5000$), the upper bound for c .

Output

The output should contain the following:

The first line should contain an integer n ($1 \leq n \leq 10^4$), the number of items in the constructed 01-knapsack.

The second line should contain n integers $w_1, w_2, \dots, w_n$ ($1 \leq w_i \leq c_{lim}$), the weights of the items.

The third line should contain n integers $v_1, v_2, \dots, v_n$ ($1 \leq v_i \leq 10^9$), the values of the items.

It can be proven that, given the problem and input conditions, a solution satisfying the above data ranges can always be found.

Example

<i>standard input</i>	<i>standard output</i>
2	2 1 2 2 3

Problem L. Let's Avoid Obstacles

Input file: *standard input*
Output file: *standard output*
Time limit: 1 second
Memory limit: 1024 mebibytes

Consider an $n \times m$ grid with row numbers ranging from 1 to n and column numbers ranging from 1 to m . Each column, except for the leftmost column 1 and the rightmost column m , can have at most one obstacle; the leftmost and rightmost columns are guaranteed to have no obstacles.

You start from any cell in the leftmost column (you can choose the row), and your goal is to reach any cell in the rightmost column (you can choose the row). Suppose you are currently at row i and column j . You can choose one of the following three operations at each step:

- Move right: from (i, j) to $(i, j + 1)$;
- Move up: from (i, j) to $(i - 1, j)$;
- Move down: from (i, j) to $(i + 1, j)$.

You are not allowed to move left, and at any time, you cannot enter an obstacle cell or move outside the grid.

There are k obstacles (numbered from $i = 0$ to $k - 1$ after sorting by column number). For each obstacle, you must choose to “go around from above” or “go around from below”.

If the i -th obstacle is at row r_i and column c_i :

- If you choose “above”, your row number in column c_i must always be less than r_i ;
- If you choose “below”, your row number in column c_i must always be greater than r_i .

The choices for different columns are independent, resulting in a total of 2^k possible schemes.

Your task: for each scheme, calculate the minimum number of steps from any row in the leftmost column to any row in the rightmost column while satisfying all the constraints of that scheme; if there is no feasible path under that scheme, output -1 .

Input

The first line of input contains an integer t ($1 \leq t \leq 50$), the number of test cases. For each test case:

The first line contains two integers, n and m ($1 \leq n \leq 100$, $2 \leq m \leq 100$), the number of rows and columns in the grid.

The second line contains an integer k ($0 \leq k \leq \min(m - 2, 10)$), the number of obstacles.

The next k lines describe obstacles. The i -th of them contains two integers, r_i and c_i ($1 \leq r_i \leq n$, $1 < c_1 < c_2 < \dots < c_k < m$), indicating that there is an obstacle at row r_i and column c_i .

The sum of $n \cdot m$ over all test cases is at most 5000. There is no such constraint on the sum of k over all test cases.

Output

For each test case, output a line containing 2^k integers, representing the minimum number of steps for each scheme numbered from 0 to $2^k - 1$. Adjacent numbers should be separated by a space. If a certain scheme has no solution, output -1 .

Scheme numbering explanation:

- Each scheme can be viewed as a binary number of length k .
- The binary digits, from the least significant bit to the most significant bit, correspond to the obstacles 0, 1, ..., $k - 1$.

- A value 0 means going around the obstacle from above; a value 1 means going around from below.

For example, if $k = 3$, there are $2^3 = 8$ schemes corresponding to the following table:

Scheme #	Binary Representation (Least → Most Significant Bit)	Bypass Choice (for obstacles 0, 1, 2)
0	000	Above, Above, Above
1	100	Below, Above, Above
2	010	Above, Below, Above
3	110	Below, Below, Above
4	001	Above, Above, Below
5	101	Below, Above, Below
6	011	Above, Below, Below
7	111	Below, Below, Below

The program should output the minimum number of steps for each scheme in ascending order of scheme numbers.

Example

<i>standard input</i>	<i>standard output</i>
3	5 -1 -1 -1
3 6	-1 3
2	-1 -1 5 -1
3 4	
2 5	
3 4	
1	
1 2	
3 6	
2	
3 2	
1 5	

Problem M. Managing Team of Robots

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 1024 megabytes

Before the robotic football game, n robots with heights $a_1, \dots, a_n$, where a_n are positive integers, lined up in a row. The team manager wants the robots to be arranged in nondecreasing order of height from left to right. But instead of rearranging the athletes, the coach may repeatedly perform the following operation:

- Choose an index i ($1 \leq i \leq n$) and modify robot i by multiplying its height by 2.

You need to find the minimum number of operations required to make the robots arranged by height.

Input

The first line of the input contains one integer n — the number of robots ($1 \leq n \leq 250\,000$). The second line contains n integers a_i — the height of the i -th robot from the left ($1 \leq a_i \leq 10^6$).

Output

Print one integer — the answer to the problem.

Example

standard input	standard output
6 3 1 4 1 5 9	4