Problem A. Ancient Maps, Hidden Danger

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

Behold, adventurer! Tread not the abyss without scrutiny.

Legend has it that there will always be a minotaur in a labyrinth. But what defines a labyrinth? Numerous grievous incidents have occurred, and the great warrior Alan decided to take matters into his own hands.

Being one of the most prestigious fighters in the town, one day, he came out with some conjectures based on his previous experiences. When you were only allowed to stay outside the structure, the area where the eyes could not see might be the actual source of danger. He rallied his friends at the bar to discuss this new thought. After that, they came to the conclusion of how to scientifically measure this property.

Picture a 2D map of a structure in your head. The structure consists of multiple walls, which can be described by polygons. These walls block movements and view. That is, you cannot walk, nor see through the wall. There are no holes in the walls (Alan is a warrior; not a mage), and therefore, a point is reachable from outside if it is not in a wall. The main insight of the discussion was, that if some reachable spot is visible outside the structure, then for any distance, it should be visible at some place with that distance to the spot for some angle. Mathematically speaking, for any radius r, we can pick a point on the circle centered at the spot, such that the segment between the point and the spot does not intersect with any wall. Any points that are not visible outside are said to be *hidden*, and they are interested in the reachable but hidden area.

Figure A.1: The labyrinth map of the Sample Input 1. The walls are shaded in brown, and the reachable but hidden area is shaded in grey.

Later that night, the folks stared at the maps perplexed, with no progress at all. None of them had a good method to measure the area other than going to the actual place. Certainly, it was too risky and impractical as Alan could not guard all the fellows at the same time. Perhaps you can help? Help them find the hidden reachable area of the map to verify their conjecture.

Input

The first line contains a number n ($1 \le n \le 30$), which is the number of the polygons describing the walls in the structure. Each of the following n lines contains several integers. The first number of each line is the number of vertices k_i ($3 \le k_i \le 90$) of the i-th polygon. Following this number, there are $2k_i$ integers, $x_1, y_1, x_2, y_2 \ldots, x_{k_i}, y_{k_i}$ ($-10^4 \le x_j, y_j \le 10^4$), which are the coordinates of the vertices of the polygon, in the clockwise or counterclockwise order. There will not be duplicated vertices in the input. The given polygons are simple, i.e. they have no hole and never intersect with themselves. Two polygons will not overlap at any point. The total number of vertices will not exceed 90.

Output

Output a floating point number indicating the reachable hidden area in the map. Your answer should

have an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
2 8 0 0 7 0 7 7 6 7 6 1 1 1 1 7 0 7 4 2 7 5 7 5 6 2 6	2.25
2 4 2 4 5 8 9 4 5 7 4 4 5 5 1 6 5 5 4	1.4

Problem B. Basic Graph Algorithm

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

Aqua is learning graph theory. After learning about Depth-First Search, he wrote the following pseudo-code:

```
function dfs(u)
    mark u as visited
    output u
    for v in vertices adjacent to u
        if v is not visited, then
            dfs(v)
        Endif
    Endfor
Endfunction

function run_dfs()
    for v in all vertices
        if v is not visited, then
            dfs(v)
        Endif
    Endfor
Endfunction
```

Aqua noticed that there are some non-deterministic behaviors in line 4 and line 12. Specifically, the order of the traversed vertices might not be fixed!

Therefore, Aqua asks you the following problem: Given an undirected graph with n vertices and m edges and a permutation $p_1, p_2, \ldots, p_n$, what is the minimum number of edges you need to add to **make it possible** for the output order to become the permutation p? Moreover, you need to provide the added edges for Aqua to check your answer.

It can be shown that there always exists an answer to Aqua's problem.

Input

The first line of the input contains two integers n, m ($1 \leq n \leq 3 \times 10^5, 0 \leq m \leq 5 \times 10^5$), indicating the number of vertices and the number of edges.

m lines follow, and the i-th of which contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$), indicating that the i-th edge connects vertex u_i and vertex v_i.

The last line contains n integer $p_1, p_2, \ldots, p_n$, indicating the given permutation.

Output

Output an integer k, the minimum number of edges you need to add, in the first line.

k lines follow, and the i-th of which contains two integers a_i, b_i, indicating the i-th edge you want to add which connects $a_i, b_i (1 \leq a_i, b_i \leq n, a_i \neq b_i)$.

If there are multiple answers, you can output any.

Examples

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

Problem C. Conquer the Multiples

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

Alice and Bob are two brilliant strategists who stumble upon a mystical blackboard in an ancient temple, where a sequence of integers $l, l+1, \ldots, r$ is written. Next to the blackboard is a special number x, initially set to l.

The rules of the game are simple but challenging: Alice and Bob take turns erasing exactly one multiple of x from the blackboard, with Alice always making the first move. After each move, the value of x is increased by 1, and the game continues. If a player cannot erase a multiple of the current x, they lose the game.

Both players, eager to prove their tactical brilliance, will play optimally to secure victory. Can you determine which player will ultimately win this battle of multiples?

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \le T \le 10^5$), denoting the number of test cases.

For each test case, there is only one line of input. This line contains two integers l, r ($1 \le l \le r \le 10^9$), denoting the range of numbers $(l, l+1, \ldots, r)$ initially on the blackboard.

Output

For each test case, output one line. If Alice will win, print "Alice" in this line. Otherwise, print "Bob".

Example

standard input	standard output
3 2 4 4 4 6 7	Alice Alice Bob

Problem D. Decrease and Swap

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

David Liu has n boxes in a row, denote box i to be the i-th box from the left, indexed from 1 to n. Initially every box is empty.

Frank has a binary string s of length n. He will put 10^{18} stones in box i for each i such that $s_i = \mathsf{1}$, and he will put 0 stones in box i for each i such that $s_i = \mathsf{0}$.

After that, David Liu will try to remove every stone in all the boxes. However, he can't just take them out, and he can only achieve this by doing the following operation zero or more times:

- Choose an integer i such that $1 \le i \le n-2$ and there is at least one stone in box i. Then he must take one stone from box i **and** swap the contents of box $i+1$ and box $i+2$.

Since there are a lot of stones, before trying himself, David Liu wants to ask you if it is possible to remove every stone in all the boxes.

Recall that a binary string is a string which only contains `0` and `1`.

Input

The first line contains an integer T ($1 \le T \le 10^6$), representing the number of test cases.

There are two lines of input for each test case.

The first line of each test case contains an integer n ($3 \le n \le 10^6$), denoting the number of boxes.

The second line of each test case contains a binary string of length n, denoting whether Frank will put stones in each box.

It is guaranteed that the sum of n across all test cases does not exceed 10^6.

Output

For each test case, output `Yes` if David Liu can remove all stones, output `No` otherwise.

Example

standard input	standard output
3 3 101 4 1010 5 00000	No Yes Yes

Problem E. Equal Measure

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

A friend bought a piece of fabric, green with white squares. "Ah, it looks like a chessboard!" "To me, it looks more like manuscript paper." "It really reminds me of mung bean cakes." The same fabric, yet each of us viewed it differently. Everyone's perspective on beauty varies, while there's no harm in people having different views. What truly matters is to tolerate and respect one another's perspectives.

If one can enjoy the beauty of a sunrise through the door, why insist they walk to the window to listen to the birds sing? You enjoy your birdsong, they admire their sunrise, and both will experience beauty in equal measure.

You decide to leave for a botanical garden and take a trip to enjoy the birdsong. There are n areas and m trails in the botanical garden, and the i-th trail connecting the u_i-th and the v_i-th area.

A route in the botanical garden is defined as a sequence of areas $a_1, a_2, \ldots, a_k$ with length k, which satisfies the following conditions:

- $k \geq 3$.
- $\forall i \neq j$, $a_i \neq a_j$.
- $\forall 1 \leq i < k$, there is a trail connecting the a_i-th and the a_{i+1}-th area.
- There is a trail connecting the a_k-th and the a_1-th area.

To ensure that every visitor experiences beauty in equal measure, you decided to write a program to check whether all routes in the botanical garden have the same length. If no route exists in the botanical garden, we consider all routes to be the same length.

Input

The first line contains an integer T ($1 \leq T \leq 5 \times 10^5$), representing the number of test cases.

In each test case, the first line contains two integers n, m ($1 \leq n \leq 5 \times 10^5$, $0 \leq m \leq 5 \times 10^5$), representing the number of areas and trails respectively. It is guaranteed that among all T test cases, $\sum n,\ \sum m \leq 5 \times 10^5$.

m lines follow. The i-th line of them contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), indicating the i-th trail connects the u_i-th and the v_i-th area. There is at most one trail connecting the same pair of areas. From any area, it is possible to reach any other area through the trails.

Output

For each testcase, output `Yes` if all routes have the same length, or `No` otherwise.

Examples

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

Problem F. Fast Bogosort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Have you heard of Bogosort? Bogosort is a very interesting random sorting algorithm, which can sort an array of n elements in the best-case time complexity of $O(n)$. The algorithm can be described as follows:

- Uniformly shuffle the entire array.
- Check if the array is sorted.

Yes, after reading the algorithm above, you will realize that the expected time complexity of Bogosort is actually $O(n \cdot n!)$!

Bogosort is so fascinating, but it runs so slowly, which is quite unfortunate. Therefore, let's help improve it by optimizing it using the following algorithm, Fast Bogosort, to sort a permutation of numbers from 1 to n:

- If the array is already sorted, stop.
- Otherwise, divide the array into as many segments as possible, such that each segment corresponds to the interval $[l_i, r_i]$ and contains exactly the permutation of numbers $[l_i, r_i]$. Note that the segments must not overlap and their union must be the entire range $[1, n]$.
- For each segment $[l, r]$ that has been divided, **if** $l < r$, call `shuffle(l, r)` to uniformly shuffle the numbers within that segment.

If we only care about the number of calls to `shuffle(l, r)`, it is clear that the original Bogosort has an expected number of calls of $n!$. Now, given a permutation of numbers from 1 to n, please calculate the expected number of times Fast Bogosort will call `shuffle(l, r)`.

Input

The first line of the input contains one integer n ($1 \le n \le 10^5$), indicating the length of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$, indicating the content of the array. It is guaranteed that $a_1, a_2, \ldots, a_n$ form a permutation of $1 \sim n$.

Output

Output the expected number of times Fast Bogosort will call `shuffle(l, r)`. It can be proved that the expected number is always a rational number. Additionally, under the constraints of this problem, it can also be proved that when that value is represented as an irreducible fraction $\frac{P}{Q}$, we have $Q \not\equiv 0 \pmod{998244353}$. Thus, there is a unique integer R such that $R \times Q \equiv P \pmod{998244353}, 0 \le R < 998244353$. Report this R.

Examples

standard input	standard output
5 2 1 5 3 4	332748123
10 4 2 3 1 6 5 9 7 10 8	453747445

Problem G. Geometry Task

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

There are n red lines and n blue lines in a two-dimensional plane. The equation of the i-th red line is $y = a_i x + b_i$, and the equation of the i-th blue line is $x = c_i$.

Define the value of pairing a red line with a blue line as the y-coordinate of their intersection point. You want to pair each red line with exactly one blue line, resulting in n values. Determine the maximum possible median of these n values.

The median of an array of length n is the $\lceil \frac{n}{2} \rceil$-th **largest** element in the array. For example, the median of array $[3, 4, 2]$ is 3 and the median of array $[1, 1, 4, 5, 1, 4]$ is 4.

Input

The first line contains the number of test cases T ($1 \le T \le 10^5$). The description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 10^5$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($-10^{18} \le b_i \le 10^{18}$).

The fourth line of each test case contains n integers $c_1, c_2, \ldots, c_n$ ($-10^9 \le c_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output a single integer — the maximum possible median value.

Example

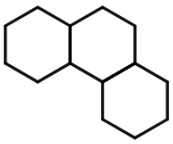
standard input	standard output
3 5 0 5 -2 1 2 9 -4 0 10 5 -4 -1 4 -2 4 10 -6 3 1 0 6 -2 -4 3 0 10 22 65 11 1 -34 -1 -39 -28 25 24 10 9 1 -2 -5 8 -7 -10 -7 -7 1 101 48763 651	9 25 114514

Problem H. Hexagon Puzzle

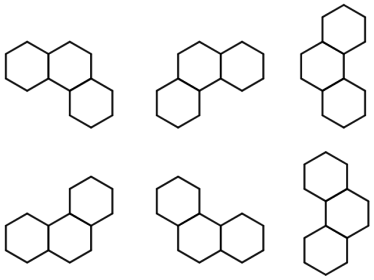
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Kotoha loves to solve puzzles. This year, her friends Saki and Yui gave her a special puzzle as her birthday gift: A hexagon puzzle and many colorful V-shaped pieces. The puzzle contains n rows of regular hexagons, where the i-th row contains exactly i regular hexagons. For every $i > 1$, the j-th regular hexagon in the i-th row shares an edge with the j-th regular hexagon and the $(j-1)$-th regular hexagon in the previous row, if they exist. Below are examples of the puzzle for $n = 1, 2, 3, 4$ respectively:

A V-shaped piece occupies 3 hexagons, as shown below. The color of each piece can be one of the 26 colors, represented by `ABC...XYZ`:



The piece can be rotated by a multiple of 60 degrees. More specifically, there are 6 possible rotations for a V-shaped piece, as shown below:



Now Kotoha wants to maximize the number of V-shaped pieces she used, while satisfying the following conditions:

- Each V-shaped piece she used must be placed inside the puzzle and occupies exactly 3 hexagons.
- No two V-shaped pieces occupy the same hexagon.
- No two adjacent V-shaped pieces share the same color. Two V-shaped pieces are considered as adjacent if the two pieces share at least one common edge.

Can you help her find the maximum number of V-shaped pieces and a way to place that?

Input

The input only contains one line with one integer n ($1 \le n \le 1024$), indicating the size of the puzzle.

Output

Output n lines. The i-th line contains $n+i-1$ characters. Let $s_{i,j}$ be the j-th character in the i-th line. Your output must satisfy the following format:

- If $j \leq n-i$, $s_{i,j}$ is a space (ASCII code 32).
- If $j-(n-i)$ is an even integer, $s_{i,j}$ is a space.
- If the $(\frac{j-(n-i)+1}{2})$-th hexagon in i-th row is not covered by any V-shaped pieces, $s_{i,j}$ is a dot (ASCII code 46).
- Otherwise, $s_{i,j}$ is used to represent the color of the V-shaped piece that covers the $(\frac{j-(n-i)+1}{2})$-th hexagon in the i-th row, which will be one of the uppercase English letters.

Your output will be considered correct if all the following conditions are satisfied:

- It satisfies all the format described above.
- All the pieces you placed are V-shaped.
- No two adjacent V-shaped pieces have the same color.
- The number of pieces you placed is maximum.

Please DO NOT output extra spaces at the end of each line, or your solution might be considered incorrect.

Examples

standard input	standard output
2	` .` `. .`

3	`  W`
	` W .`
	`. W .`

4	`   .`
	`  . R`
	` . B R`
	`B B R .`

Problem I. In Search of the Ultimate Artifact

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

Justin, a skilled mage, has spent years amassing a collection of n mystical artifacts. Each artifact is imbued with a certain amount of power, represented by a non-negative integer. Recently, Justin stumbled upon a legendary spell called the "k-fusion," a technique said to enhance his artifacts to unprecedented levels.

With the "k-fusion" spell, Justin can take exactly k artifacts and fuse them into a single artifact. The power of this newly formed artifact is the product of the powers of the k consumed artifacts. However, this process is irreversible, as the original k artifacts are forever lost in the fusion.

Justin is determined to use this spell to craft the most powerful artifact possible. He performs some (possibly zero) number of k-fusions which maximizes the maximum power of his remaining artifacts. Once finished, Justin challenges you to calculate the maximum power of the remaining artifacts.

But there's a twist! The result could be an astronomically large number, and Justin doesn't want to deal with such huge values. Instead, he asks you to compute the remainder of the maximum power when divided by 998244353 – a prime number of great significance in his magical research.

Can you help Justin determine the remainder of the maximum possible power of the remaining artifact after his sequence of k-fusions?

Input

This problem contains multiple test cases. The first line of input contains an integer T ($1 \le T \le 1000$), denoting the number of test cases.

For each test case, there are two lines of input. The first line of input contains two integers n, k ($2 \le n \le 2 \times 10^5, 2 \le k \le n$), denoting the initial number of artifacts and the parameter of k-fusion.

The second line of input contains n non-negative integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le 10^9$), where p_i is the power of the i-th artifact.

It is guaranteed that the sum of n over all test cases will not exceed 2×10^5.

Output

For each test case, output an integer in one line, denoting the answer to Justin's challenge.

Example

standard input	standard output
3 8 3 44 5 2018 8 8 2024 8 28 5 4 4 5 5 1 0 5 2 0 0 0 0 0	923923948 100 0

Problem J. Just-in-Time Render Analysis

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Websites contain various elements, including visible UI elements and invisible grouping frames. As you can see on the DOMjudge team's interface, it can be divided into multiple components. These regions are organized in a hierarchical tree structure, called the DOM tree, although they are not directly related.

You are a developer of a brand-new browser project. In the project, the elements are modeled as rectangles placed on the Cartesian plane, in such a way that they are properly nested. That is, each pair of rectangles is either strictly contained one another or completely disjoint.

The current algorithm runs in iterations: the browser renders the outermost elements, which are not contained by any others, in a single iteration. These elements are marked as completed and ignored in subsequent iterations. This process repeats until all elements are rendered. Thus, the browser computes *render depth* of an element as one more than the maximum render depth of all the other elements strictly *containing* it, or zero if there is no such element.

However, modern websites often feature more animations, overwhelming the current rendering algorithm. To identify the performance bottleneck, your team needs a tool to analyze the website's structure on the fly. Each element has an animation switch, controlling whether it shows its animation. To get a better view of the page, a key property of an element is whether any portion of it displays its animation. Therefore, an element is called *animated* if it contains some element (including itself) of which the animation switch is on.

To measure how slow an iteration of the rendering process can potentially be, you are given the task of monitoring the resource usage of each iteration. More specifically, you need to handle the toggling of the animation switches, and answer the queries that ask for the number of *animated* elements with some given render depth.

Input

The first line of input contains two integers n ($1 \le n \le 5 \times 10^5$) and q ($1 \le q \le 5 \times 10^5$), where n is the number of elements and q is the total number of upcoming events.

Each of the next n line contains four integers x_1, y_1, x_2, y_2 ($1 \le x_1 < x_2 \le 10^9, 1 \le y_1 < y_2 \le 10^9$), indicating the element is a rectangle with bottom-left corner (x_1, y_1) and upper-right corner (x_2, y_2). The boundaries of the rectangles never intersect with each other, not even at a point.

Following this are q lines, indicating the events in chronological order. Each line is either in the format of `^` i, or `?` k. The first format means the animation switch of the i-th element ($1 \le i \le n$), in the order of input, should be toggled, i.e. on to off and vice versa. The second format means it is a query of the number of animated elements with render depth k ($0 \le k$).

Initially, all animation switches are off, and there is at least one event of the second format. The render depth specified in the queries will not exceed the maximum render depth of all elements.

Output

For each query, output the number of animated elements with the given render depth in a line.

Example

standard input	standard output
8 7 1 1 15 11 2 2 7 4 2 5 7 10 3 6 4 9 5 6 6 7 8 2 14 10 9 5 13 9 11 6 12 7 ^ 4 ^ 5 ^ 8 ? 0 ? 1 ? 2 ? 3	1 2 3 1

Problem K. Knights of Night

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 megabytes

On the sixteenth night of the phantasmagoria, the knights gather around, prepared for a fight.

Two groups of knights are set to fight, each group consists of n knights. The i-th knight in the first group has a strength of a_i, and the j-th knight in the second group has a strength of b_j.

Your task is to organize the fights between the knights for tonight. Each fight involves one knight from each group, and each knight can participate in at most one fight. The outcome of the fights is not important—what truly matters is how wild, or lunatic, the fights become.

The *lunaticus* of a fight is defined as the sum of the strengths of the two knights involved. However, there's a special condition: if the lunaticus of a fight reaches 998244353, it overflows and resets to zero. Specifically, the fight between the i-th knight from the first group and the j-th knight from the second group has a lunaticus of $(a_i + b_j) \bmod 998244353$.

The special condition applies only to the lunaticus of individual fights. When calculating the sum of lunaticus for multiple fights, the sum of lunaticus can exceed 998244353.

Since the number of fights has not been decided, you would like to calculate that, for each integer x from 1 to k, what is the maximum possible sum of lunaticus for exactly x fights held tonight. In addition, there are m pairs of specific fights that cannot be organized. Please maximize the sum of lunaticus while avoiding these m restrictions.

Input

The first line contains three integers n, m, k ($1 \le n \le 10^5, 0 \le m \le 3 \times 10^5, 1 \le k \le \min(n, 200)$), representing the number of knights per group, the number of additional pairs of fights that cannot be organized, and the number of questions you need to answer.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 998244353$), representing the knights' strength in the first group.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i < 998244353$), representing the knights' strength in the second group.

Followed by m lines, the i-th of which contains two integers u_i, v_i ($1 \le u_i, v_i \le n$), representing that you cannot organize a fight by the pair of the u_i-th knight in the first group, and the v_i-th knight in the second group. It is guaranteed that all given pairs are distinct.

Output

Output k numbers in one line, the i-th of which represents the maximum possible sum of lunaticus for exactly i fights held tonight. For the i-th number, if it is impossible to organize i fights, output -1 instead.

Examples

standard input	standard output
3 4 3 10 998244352 5 998244352 8 6 1 2 1 3 2 2 3 3	998244351 998244364 27
6 10 5 749208241 448597025 773867529 808779198 281439035 850615248 796547348 327547103 1 4 5 3 3 4 2 1 5 6 4 3 6 2 6 3 2 6 2 4	882168541 1667618296 2328768389 2900938913 3354309143 554621438 19807774 765641981 702406342
1 1 1 0 0 1 1	-1

Problem L. Lazy Susan

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

A lazy Susan is a rotating round tray placed on a dining table to hold food or utensils, making it easier for everyone seated around the table to share and serve themselves.

There are n people sitting around a round table enjoying dinner, with each person numbered from 0 to $n-1$ in clockwise order, and the distance between each of the n people is the same. Note that the person next to the $(n-1)$-th person in the clockwise direction is the 0-th person.

The dinner consists of n dishes, each numbered from 0 to $n-1$ in the order they are served. The waiter will start from the x-th person and serve each dish on the lazy susan in front of each person in clockwise order. Specifically, the i-th dish will be placed in front of the $[(x+i) \bmod n]$-th person.

Each time the lazy susan is rotated, all dishes can be rotated either clockwise or counterclockwise to the next person's position. More precisely, the dish originally in front of the i-th person will move to the $[(i+1) \bmod n]$-th person's position after one clockwise rotation, and to the $[(i+n-1) \bmod n]$-th person's position after one counterclockwise rotation. Each operation takes 1 second.

Each person has a reach distance r_i. The i-th person can enjoy the dish that is currently in front of the j-th person if there exist an integer k that satisfies the following conditions:

- $-r_i \le k \le r_i$
- $(i+k+n) \bmod n = j$

The person can enjoy the dish instantaneously once the conditions above are satisfied.

There are m preferences among the n people. The i-th preference is that the p_i-th person wants to enjoy the d_i-th dish before t_i seconds after all dishes are served. Please determine whether all preferences can be satisfied if the waiter starts serving from the x-th person for all x between 0 and $n-1$.

Input

The first line contains the number of test cases T ($1 \le T \le 2500$). The description of the test cases follows.

The first line of each test case contains two integers n, m ($2 \le n \le 5000, 0 \le m \le \min(n^2, 10^5)$), indicating the number of people and the number of preferences.

The second line of each test case contains n non-negative integer $r_0, r_1, \ldots, r_{n-1}$ ($0 \le r_i \le n$), indicating the reach distance of the i-th person.

The next m lines of each test case contains three integers p_i, d_i, t_i ($0 \le p_i < n, 0 \le d_i < n, 1 \le t_i \le 10^9$), indicating the information of the i-th preference. It is guaranteed that for every $1 \le i < j \le m$, $p_i \ne p_j$ or $d_i \ne d_j$.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

It is guaranteed that the sum of m over all test cases does not exceed 10^5.

Output

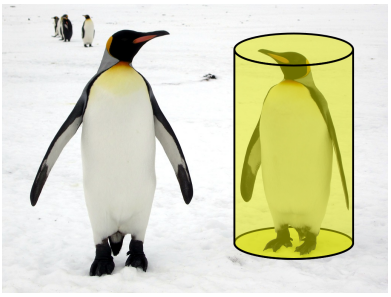
For each test case, output a 01 string s of length n. If the waiter starts can serve from the x-th person, $s_x = 1$. Otherwise $s_x = 0$.

Example

standard input	standard output
4 5 6 0 0 1 2 1 3 2 2 0 0 2 4 4 4 1 1 3 4 3 5 0 3 2 8 10 2 2 2 0 3 0 0 1 4 3 1 1 0 1 0 7 6 4 4 1 1 7 5 0 4 2 5 3 4 4 6 1 7 7 3 0 2 4 4 4 0 0 0 0 1 0 2 2 0 2 0 0 2 3 0 2 13 0 1 1 4 5 1 4 1 9 1 9 8 1 0	10100 11111010 0000 1111111111111

Problem M. Machine Learning with Penguins

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



Approximated penguin, original image by David Stanley, used under license CC BY 2.0.

Penguins are marine flightless birds which live almost only in the southern hemisphere. These great swimmers are always one of the most popular species in zoos, but unfortunately, such cute creatures are jeopardized by climate change. That's why the International Conservation of Penguins Center (ICPC) launched a project to investigate how to save them. They have launched state-of-the-art autonomous vehicles — plenty of rovers (with camouflage) going around their site collecting images for research. By combining multiple pictures from different points of view, it was possible to study these avians under 3D models, in a more systematic way. However, the drawback was such computation was clearly not possible through human efforts, which was where machine learning came in. Unfortunately, it did not work out. The machine was marking and creating complete nonsense.

Maybe something in the model was wrong. Nevertheless, ICPC needed a complete overhaul of its training methods. Numerous proposals were made, and most of them were easy to implement. However, there was one particular problem they found especially challenging: they wanted to penalize the model for marking shapes that do not resemble penguins, but what exactly makes a shape look like a penguin?

Turns out, physicists have known this for a couple of years: it's a circular cylinder!

How hard could it be to check if there is a circular cylinder of which the surface goes over all the points marked by the algorithm? The truth is, there has not been an effective result during the months. As a consequence, they lowered their short-term target, deciding to only feed in pictures of penguins standing straight. That is, the cylinder is restricted to a right circular cylinder (a cylinder with its cylindrical surface perpendicular to the circle base) with the base of it lying on the x-y plane. Of course, the penguin is neither imaginary nor infinite. The cylinders with zero or infinite volume should be excluded. Could you, renowned computer scientists and the master of geometry, solve this task?

Input

The first line contains a number n ($1 \le n \le 10^5$), which is the number of points the model has marked. Each of the following n lines contains three integers x, y, and z ($-10^9 \le x, y \le 10^9, 0 \le z \le 10^9$), indicating that there is a point in the three dimensional space with coordinate (x, y, z). There will not be duplicated points in the input.

Output

If there is a cylinder satisfying all the constraints, output "`probably`" on the first line. Otherwise, output "`not a penguin`" in a line.

Examples

standard input	standard output
5 0 0 0 3 4 1 -3 -4 2 -3 4 3 3 -4 4	probably
4 0 6 2 3 4 1 -3 -4 1 -3 4 1	not a penguin
2 0 0 0 1 1 0	probably