Problem A. Task Computing

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

As it says, *Time is Money, Efficiency is Life*. A client has a computing task waiting for uploading to the cloud servers. However, due to the resource limits and many other tasks, a task usually cannot be worked on a single server but multiple servers, one after another. You, as an employee of a cloud server provider who specializes in task allocation, need to select several servers from a limited number of cloud servers to perform calculations. We know that even the same servers will perform differently in different environments.

There are n cloud servers available in the vicinity of the client's region numbered from 1 to n. The i-th cloud server has two parameters: w_i and p_i, which indicate the computing power and transmission efficiency of that server, respectively. The supervisor requires that each task must be computed by **exactly** m of these servers. The client's computational task is not parallelizable, so you must sequence the work of the m servers to maximize the total computational efficiency. We define the total computational efficiency as follows:

$$\sum_{i=1}^{m} w_{a_i} \prod_{j=0}^{i-1} p_{a_j}$$

where $a_1, a_2, \ldots, a_m$ (pairwise distinct, $1 \le a_i \le n$) are the servers you selected. For convenience, $a_0 = 0, p_0 = 1$.

Input

The first line contains two integers n, m ($1 \le n \le 10^5$, $1 \le m \le \min(n, 20)$), denoting the number of cloud servers and servers that you have to select.

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 10^9$), denoting the servers' computing power.

The third line contains n integers $q_1, q_2, \ldots, q_n$ ($8000 \le q_i \le 12000$), where $p_i = \frac{q_i}{10000}$ denotes the i-th server's transmission efficiency.

Output

Output a float number denoting the maximal total computational efficiency. Your answer is considered correct if the relative or absolute error between yours and the standard solution is not greater than 10^{-6}.

Example

standard input	standard output
5 2 1 2 3 4 5 12000 11000 10000 9000 8000	8.5000000000000000

Problem B. 2D Internet Angel

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

Welcome To My Religion.

Internet Angel descends! She is aimed at saving us, humans. The people waiting to be redeemed are enclosed in what can be seen as a convex polygon of n vertices on a 2D plane. This convex polygon is somehow special: there exists a circle with its center at (0,0) that is tangent to every side of the polygon. We denote the circle by `Circle A`.

Angle Chan does not descend directly into the crowd but additionally draws a large circle with the center at (0,0) that can surround the whole crowd. We denote the circle by `Circle B`. Next, she will choose a location between the circle and the crowd uniformly randomly, and then for all the tangent points of each side of the polygon, she will choose a point nearest to her and follow the shortest path to the crowd.

For convenience, when inputting this convex polygon we input `Circle A` and n angles indicating the position of the tangent point of each side of the polygon on `Circle A`. You can check the input format and the sample to get a better understanding.

Now, Angle Chan wants to know what is the expectation of the **squared** distance of this shortest path.

Input

The first line contains an integer $T(1 \le T \le 20)$, denoting that there are T test cases.

For each test case, the first line contains three integers n, R_1, R_2 ($3 \le n \le 10^4$, $1 \le R_1 < R_2 \le 10^4$), indicating the number of points of the convex polygon, the radius of `Circle A` and `Circle B` respectively.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^4 \le a_1 < a_2 < \ldots < a_n < 10^4$), indicating that the tangent point of the i-th side lies at $(R_1 \cos \frac{a_i \cdot \pi}{10000}, R_1 \sin \frac{a_i \cdot \pi}{10000})$.

It is guaranteed that all the tangent lines can form a convex polygon, and all points inside or on the sides of the convex polygon are strictly inside `Circle B`.

Output

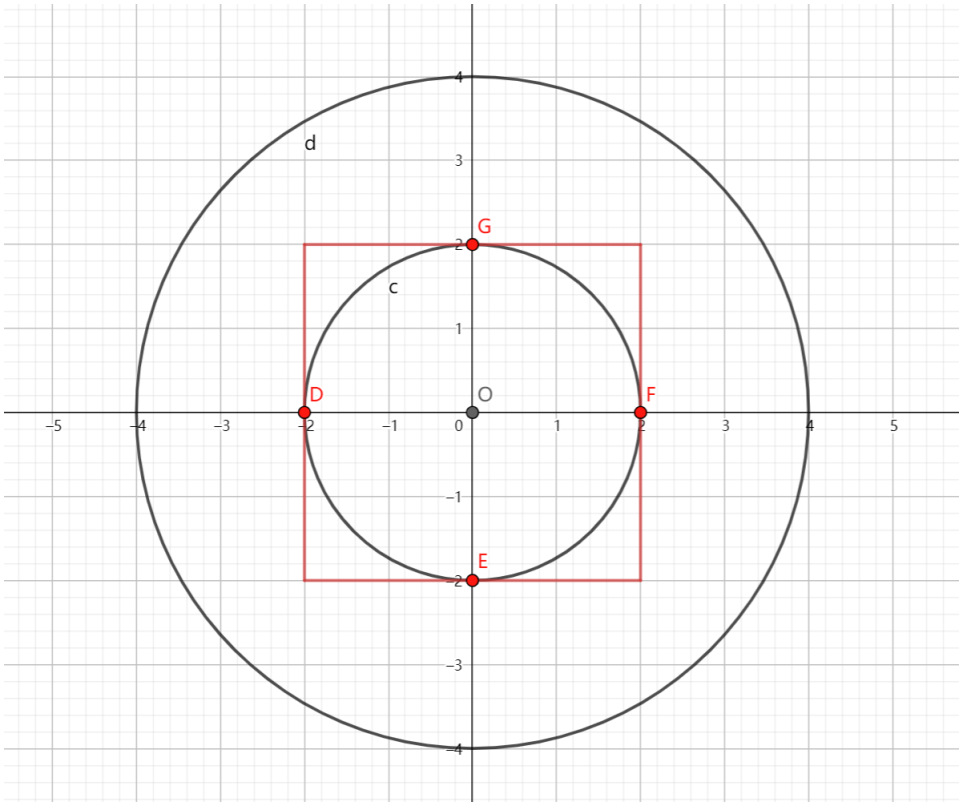
For each test case, print the expected **squared** distance in a line. Your answer will be considered correct if the absolute or relative error between yours and the standard answer is no more than 10^{-6}.

Example

standard input	standard output
1 4 2 4 -10000 -5000 0 5000	2.893122167156

Note

The figure of the example:



where D, E, F, G are the tangent points given in the input respectively.

Problem C. Easy Counting Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

Yukikaze is learning stringology. She found a series of interesting problems at the end of her textbook as follows.

We consider a string to be good if it satisfies the following conditions:

1. It consists of decimal digits less than w.
2. Digit i appears at least c_i times.

We consider two strings to be different if they are different in length or there exists an integer i such that the letters in their i-th position are different. The problem in the textbook asks for the number of different good strings of length n.

Input

The first line contains a single integer w ($2 \le w \le 10$), indicating that a good string consists of decimal digits less than w.

The second line contains w integers $c_0, c_1, \ldots, c_{w-1}$ ($1 \le c_i \le 50000$, $c_0 + c_1 + \ldots + c_{w-1} \le 50000$), indicating that the decimal digit i appears at least c_i times in a good string.

The third line contains a single integer q ($1 \le q \le 300$), indicating the number of queries.

Each of the next q lines contains a single integer n ($1 \le n \le 10^7$), indicating the length of the string.

Output

For each query, output an integer in a single line indicating the number of different good strings of length n. As the answer may be large, please output the answer modulo 998244353.

Examples

standard input	standard output
2 1 1 3 3 5 7	6 30 126
6 11 45 14 11 45 14 3 114514 1919 810	519265932 6372511 784491879

Note

If we set $n = 3$, $w = 2$, $c_0 = c_1 = 1$, then the answer is 6, including `001`, `010`, `011`, `100`, `101`, `110`.

Problem D. Jobs (Easy Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Note: The only difference between the easy and the hard version is the constraints on n and m_i.

Since Little Horse gets a golden prize in a programming contest, many companies send offers to him. There are n companies, and the i-th company has m_i available jobs. To get an offer from a company, you need to be qualified for the job.

How to be qualified? The company will test your "three quotients": IQ (Intelligence Quotient), EQ (Emotional Quotient), and AQ (Adversity Quotient). Each job has three lower limits of the three quotients respectively. If all of your IQ, EQ, and AQ are no less than the corresponding lower limits, you are qualified for the job. As long as you are qualified for at least one job in a company, the company will send an offer to you.

Little Horse's three quotients are all high enough, so all the companies send offers to him. But Little Horse has q friends that worry about their jobs. Given the value of IQ, EQ and AQ of each friend, Little Horse wants to know how many companies will send offers to each friend.

Input

The first line of input contains two integers n, q ($1 \le n \le 10$, $1 \le q \le 2 \times 10^6$), indicating the number of companies and Little Horse's friends.

Then in the next n lines, the first integer of the i-th line is m_i ($1 \le m_i \le 10^5$) — the number of jobs in the i-th company. Then there follow $3m_i$ integers. The j-th triple integers a_j, b_j, c_j ($1 \le a_j, b_j, c_j \le 400$) indicate the lower limits of IQ, EQ, and AQ for the j-th job in the i-th company.

The next line contains one integer seed ($10^8 \le \text{seed} < 2^{31}$). Then the IQ, EQ, and AQ of these q friends are generated as follows.

First, register an `mt19937` random engine `rng` with the seed and a uniform integer distribution object.

```
#include <random>

std::mt19937 rng(seed);
std::uniform_int_distribution<> u(1,400);
```

Then, the value of IQ, EQ, and AQ are generated as follows.

```
int lastans=0;
for (int i=1;i<=q;i++)
{
    int IQ=(u(rng)^lastans)%400+1;  // The IQ of the i-th friend
    int EQ=(u(rng)^lastans)%400+1;  // The EQ of the i-th friend
    int AQ=(u(rng)^lastans)%400+1;  // The AQ of the i-th friend
    lastans=solve(IQ,EQ,AQ);  // The answer to the i-th friend
}
```

Output

Let's denote the answer to the i-th friend as ans_i. You should output:

$$\sum_{i=1}^{q} \text{ans}_i \cdot \text{seed}^{q-i} \bmod 998244353$$

in a single line.

Example

standard input	standard output
3 5 2 1 1 2 2 1 2 3 1 3 2 2 3 1 2 3 3 2 2 3 1 3 2 1 191415411	34417749

Note

The value of IQ, EQ, and AQ of the 5 friends in example are shown as follows.

92 108 303
116 36 265
255 132 185
360 219 272
8 115 254

The answers to each friend are all 3.

Problem E. Jobs (Hard Version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Note: The only difference between the easy and the hard version is the constraints on n and m_i.

Since Little Horse gets a golden prize in a programming contest, many companies send offers to him. There are n companies, and the i-th company has m_i available jobs. To get an offer from a company, you need to be qualified for the job.

How to be qualified? The company will test your "three quotients": IQ (Intelligence Quotient), EQ (Emotional Quotient), and AQ (Adversity Quotient). Each job has three lower limits of the three quotients respectively. If all of your IQ, EQ, and AQ are no less than the corresponding lower limits, you are qualified for the job. As long as you are qualified for at least one job in a company, the company will send an offer to you.

Little Horse's three quotients are all high enough, so all the companies send offers to him. But Little Horse has q friends that worry about their jobs. Given the value of IQ, EQ and AQ of each friend, Little Horse wants to know how many companies will send offers to each friend.

Input

The first line of input contains two integers n, q ($1 \le n \le 10^6$, $1 \le q \le 2 \times 10^6$), indicating the number of companies and Little Horse's friends.

Then in the next n lines, the first integer of the i-th line is m_i ($1 \le m_i \le 10^6, \sum m_i \le 10^6$) — the number of jobs in the i-th company. Then there follow $3m_i$ integers. The j-th triple integers a_j, b_j, c_j ($1 \le a_j, b_j, c_j \le 400$) indicate the lower limits of IQ, EQ, and AQ for the j-th job in the i-th company.

The next line contains one integer seed ($10^8 \le \text{seed} < 2^{31}$). Then the IQ, EQ, and AQ of these q friends are generated as follows.

First, register an `mt19937` random engine `rng` with the seed and a uniform integer distribution object.

```
#include <random>

std::mt19937 rng(seed);
std::uniform_int_distribution<> u(1,400);
```

Then, the value of IQ, EQ, and AQ are generated as follows.

```
int lastans=0;
for (int i=1;i<=q;i++)
{
    int IQ=(u(rng)^lastans)%400+1;  // The IQ of the i-th friend
    int EQ=(u(rng)^lastans)%400+1;  // The EQ of the i-th friend
    int AQ=(u(rng)^lastans)%400+1;  // The AQ of the i-th friend
    lastans=solve(IQ,EQ,AQ);  // The answer to the i-th friend
}
```

Output

Let's denote the answer to the i-th friend as ans_i. You should output:

$$\sum_{i=1}^{q} \text{ans}_i \cdot \text{seed}^{q-i} \bmod 998244353$$

in a single line.

Example

standard input	standard output
3 5 2 1 1 2 2 1 2 3 1 3 2 2 3 1 2 3 3 2 2 3 1 3 2 1 191415411	34417749

Note

The value of IQ, EQ, and AQ of the 5 friends in example are shown as follows.

92 108 303
116 36 265
255 132 185
360 219 272
8 115 254

The answers to each friend are all 3.

Problem F. Palindromic Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

You are given a tree with n nodes. A character `0` or `1` is written on each node. You are also given q queries. For each query, you need to calculate the number of different palindromic substrings of the string obtained from the simple path from node u to node v.

We say two strings s and t are different if their lengths are different or there exists at least one position i satisfying $s_i \neq t_i$. A palindromic string is a string that reads the same forward or backward.

Input

The first line of the input contains an integer T ($1 \leq T \leq 10^4$), indicating the number of test cases.

For each test case, the first line contains two integers n, q ($1 \leq n, q \leq 10^5$), indicating the number of nodes and the number of queries.

The second line contains a string of length n consisting of `0` or `1`, indicating the characters written on nodes from 1 to n.

Each of the following $n-1$ lines contains two integers u, v ($1 \leq u, v \leq n$), indicating an edge connecting nodes u and v. It is guaranteed that these edges form a tree.

Each of the following q lines contains two integers u, v ($1 \leq u, v \leq n$), indicating a query with nodes u and v.

It is guaranteed that $\sum n \leq 10^5$ and $\sum q \leq 10^5$ over all test cases.

Output

For each query, output the answer in a single line.

Example

standard input	standard output
1 5 5 01110 1 2 1 3 2 4 2 5 1 1 1 2 3 4 3 5 4 5	1 2 4 4 3

Problem G. Wall Builder I

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

As a wealthy man, NIO buys an empty house that can be viewed as a rectangle of length a and width b. As the builder employed by NIO, you need to divide the space of his house into several rooms.

NIO selects n points on the four borders of the house, and you should build the walls according to the following steps:

1. Select a point on the border that has not been selected.
2. Start building a wall vertical to the border from this point.
3. Stop building when it meets the wall that has been built or another border of the house.
4. End if all the points have been selected. Otherwise, go back to step 1.

After that, the space of the house will be divided into several rooms, the largest of which will serve as the living room. NIO wants his living room to be as large as possible. As you can decide the order of selection of the points, please tell him the maximum area of the living room. The thickness of the walls is negligible.

Input

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

The first line of each test case contains three integers n, a, b ($1 \le n \le 10^5$, $1 \le a, b \le 10^9$), indicating the number of points on the border, the length of the house and the width of the house.

Each of the following n lines contains two integers x, y ($0 \le x \le a$, $0 \le y \le b$), indicating the coordinate of the point. You can consider the house as a rectangle on the Cartesian coordinate system whose edges are parallel to the x or y axis, with one vertex at $(0, 0)$ and another at (a, b). For a point (x, y) on the border, it is guaranteed that one of the following conditions is met:

- $x = 0$, $0 < y < b$
- $x = a$, $0 < y < b$
- $0 < x < a$, $y = 0$
- $0 < x < a$, $y = b$

For any two different points (x_i, y_i) and (x_j, y_j), it is also guaranteed that:

- $(x_i, y_i) \neq (x_j, y_j)$.
- if $|x_i - x_j| = a$, then $y_i \neq y_j$.
- if $|y_i - y_j| = b$, then $x_i \neq x_j$.

It is guaranteed that the sum of n over all test cases won't exceed 10^5.

Output

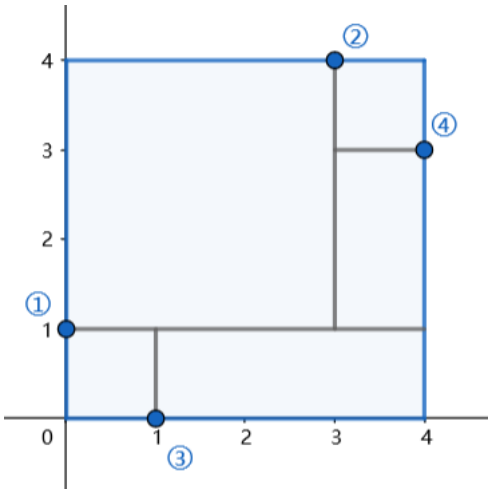
For each test case, output one integer indicating the maximum area of the living room in a single line.

Example

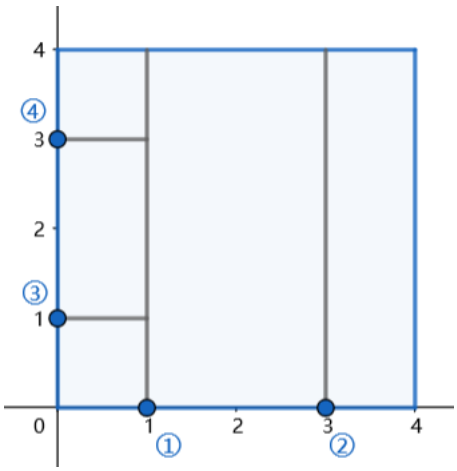
standard input	standard output
2 4 4 4 0 1 1 0 4 3 3 4 4 4 4 0 1 0 3 1 0 3 0	9 8

Note

A possible solution for the first test case in the sample:



A possible solution for the second test case in the sample:



Problem H. Wall Builder II

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

As the builder employed by NIO, you need to build more walls for him. Now you have a number of bricks of height 1 but with different lengths. More precisely, you have n bricks of length 1, $n-1$ bricks of length 2, ..., two bricks of length $n-1$, and one brick of length n. The width of the bricks is negligible. You need to build a wall with these bricks whose shape must be strictly rectangular. When building the wall, you must keep the height of the bricks at 1, which means you cannot rotate them.

Of course, you can definitely build a wall with a height of 1 and a large length, but it is not a beautiful wall. In NIO's opinion, a beautiful wall should have the minimum possible circumference. Please tell him the minimum circumference of the wall, and how to build it.

Input

The first line contains an integer T ($1 \le T \le 100$), indicating the number of test cases.

Each test case contains an integer n ($1 \le n \le 100$) in a single line. It is guaranteed that the sum of n over all test cases won't exceed 200.

Output

For each test case, output an integer in a single line, indicating the minimum circumference of the wall. Then in the next $\frac{n(n+1)}{2}$ lines, each line contains four integers x_1, y_1, x_2, y_2, indicating that the coordinate of the lower left of the brick is (x_1, y_1), and the upper right is (x_2, y_2).

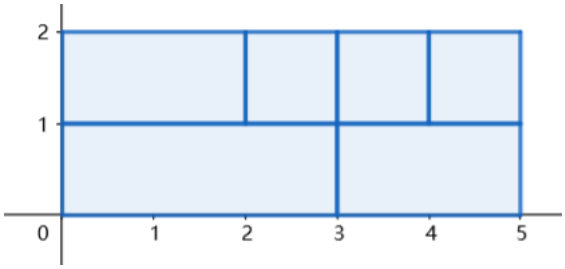
You can consider the wall as a rectangular in the Cartesian coordinate system, where the x-axis represents the length and the y-axis represents the height. The lower left of the wall must be located at $(0, 0)$. If there are multiple solutions, output any.

Example

standard input	standard output
4 1 2 3 4	4 0 0 1 1 8 0 1 1 2 1 1 2 2 0 0 2 1 14 2 1 3 2 3 1 4 2 4 1 5 2 3 0 5 1 0 1 2 2 0 0 3 1 18 4 0 5 1 2 3 3 4 3 3 4 4 4 3 5 4 3 1 5 2 3 2 5 3 0 3 2 4 0 1 3 2 0 2 3 3 0 0 4 1

Note

A possible solution for the third test case in the sample:



Problem I. Three Body

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

"The fish who dried the sea went onto land before they did this. They move from one dark forest to another dark forest."

You are now in a K-dimensional universe, which could be seen as an $n_1 \times n_2 \times ... \times n_K$ hyperrectangle. To be specific, let's divide the universe into $n_1 \times n_2 \times ... \times n_K$ hypercubes sized $1 \times 1 \times ... \times 1$ to describe different regions. The position of each region can be denoted by a K-dimensional integer coordinate $(a_1, a_2, ..., a_K)$ where $0 \le a_i < n_i$. Each region owns a *Dimensional Value* and the initial value is K. However, due to the space war and the abuse of dimensionality weapons, the *Dimensional Values* of some regions in the universe has decayed to less than K.

Your spaceship is an $m_1 \times m_2 \times ... \times m_K$ hyperrectangle which can also be viewed as $m_1 \times m_2 \times ... \times m_K$ parts of hypercubes sized $1 \times 1 \times ... \times 1$. The position of each part can also be denoted by a K-dimensional integer coordinate $(b_1, b_2, ..., b_K)$ where $0 \le b_i < m_i$. Each part of your spaceship also owns a *Dimensional Value* which is initially K. Thanks to advanced technology, some parts of your spaceship are reformed and their *Dimensional Values* become less than K, which can make them enter the regions with lower *Dimensional Values*.

We define the spaceship to be in a valid position if the following conditions are met:

- Each part of the spaceship is located exactly in one region.
- Each part of the spaceship cannot have a *Dimensional Value* strictly higher than the *Dimensional Value* of the region.

Please calculate the number of different valid positions for your spaceship. Please note that you can't rotate the spaceship.

Input

The first line contains an integer K ($2 \le K \le 5$).

The second line contains K integers $n_1, n_2, ..., n_K$ ($1 \le \prod_{i=1}^{K} n_i \le 3 \times 10^5$), indicating the size of the universe.

The third line is an integer c_u ($0 \le c_u \le \prod_{i=1}^{K} n_i$), indicating the number of regions whose *Dimensional Value* has decayed.

Each of the following c_u lines contains $K + 1$ integers $a_1, a_2, ..., a_K$ and d ($0 \le a_i < n_i$, $1 \le d < K$), indicating that the *Dimensional Value* of region $(a_1, a_2, ..., a_K)$ has decayed to d. It is guaranteed that the c_u regions are pairwise different.

The next line contains K integers $m_1, m_2, ..., m_K$ ($1 \le m_i < n_i$), indicating the size of the spaceship.

The next line contains an integer c_s ($0 \le c_s \le \prod_{i=1}^{K} m_i$), indicating the number of parts of the spaceship are reformed.

Each of the following c_s lines contains $K + 1$ integers $b_1, b_2, ..., b_K$ and d ($0 \le b_i < m_i$, $1 \le d < K$), indicating that the *Dimensional Value* of part $(b_1, b_2, ..., b_K)$ becomes d. It is guaranteed that the c_s parts are pairwise different.

Output

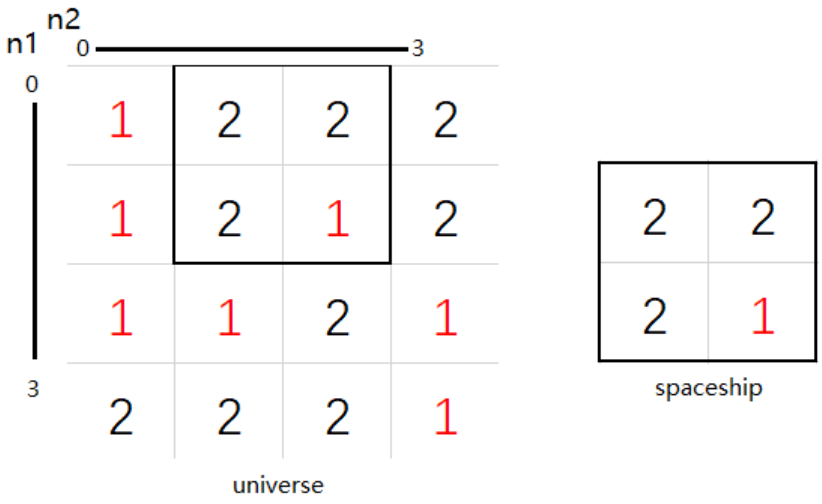
Output one integer indicating the number of different valid positions for your spaceship.

Examples

standard input	standard output
2 4 4 7 0 0 1 1 0 1 1 2 1 2 0 1 2 1 1 2 3 1 3 3 1 2 2 1 1 1 1	1
3 4 4 4 4 0 1 0 1 1 2 2 2 3 2 3 2 3 3 0 1 2 2 2 1 1 0 0 2	15

Note

For the first sample, the universe is two-dimensional. The *Dimensional Values* of the universe and spaceship are as follows.



Problem J. Counting Fish Again

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Many years later, as he faced the Nowcoder Multi-University Training Contest, ACMer NIO was to remember that distant afternoon when his coach took him to count fish.

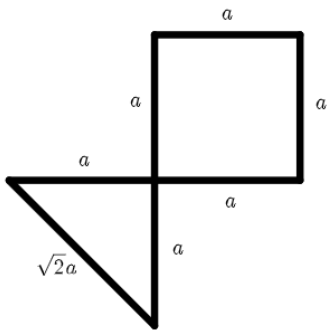
In a famous fish-counting problem in 2019, OIer NIO was asked to count the number of fish in 2D space. Unfortunately, the characteristics of fish were too complex, making counting them an almost impossible task. So let's look at this simplified fish-counting problem.

In this problem, NIO is given an infinite grid graph. And the god of fish will perform m operations on it. Assuming the coordinates of the bottom left corner of the grid graph are $(0, 0)$, there are two types of operations:

- $1\ x_1\ y_1\ x_2\ y_2$: Draw a segment between (x_1, y_1) and (x_2, y_2). It is guaranteed that $x_1+y_1 = x_2+y_2$, and the drawn segments do not overlap with the existing ones.
- $2\ x_1\ y_1\ x_2\ y_2$: Erase a segment between (x_1, y_1) and (x_2, y_2). Still, $x_1 + y_1 = x_2 + y_2$, and the segment alreay exists in the grid graph.

After each operation, NIO needs to calculate the number of fish in this grid. In this problem, a fish is defined as a shape that satisfies the following rules.

- It consists of a square and an equilateral right triangle, and the triangle's right-angle vertex coincides with one vertex of the square.
- The length of the right-angle side of the triangle is equal to the square's side length.



Note that the original segments of the grid graph can also be part of a fish.

Input

The first line contains a single integer m ($1 \le m \le 10^6$), indicating the number of operations.

Each of the next m lines contains five integers op, x_1, y_1, x_2, y_2 ($1 \le op \le 2$, $0 \le\ x_1,\ x_2,\ y_1,\ y_2 \le 10^6$, $x_1 \ne x_2$, $x_1 + y_1 = x_2 + y_2$), indicating the operation and the segment between (x_1, y_1) and (x_2, y_2).

Output

For each query, output an integer in a single line indicating the number of fish in the graph.

Example

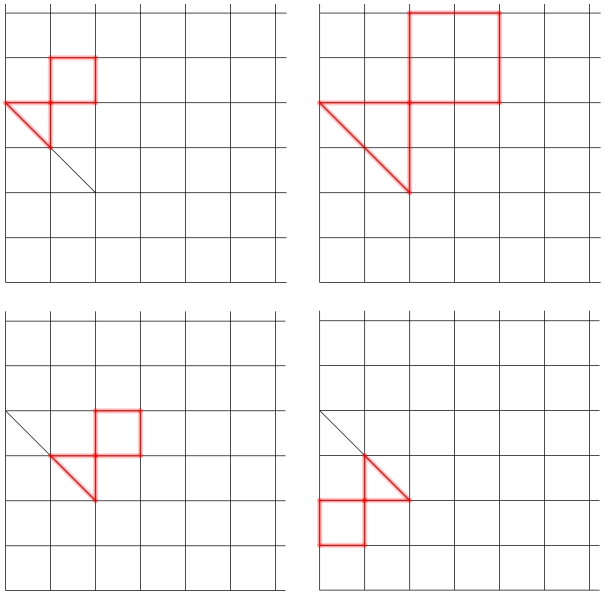
standard input	standard output
3 1 0 4 2 2 1 2 2 4 0 2 1 3 4 0	4 12 1

Note

Here is the explanation of the example.

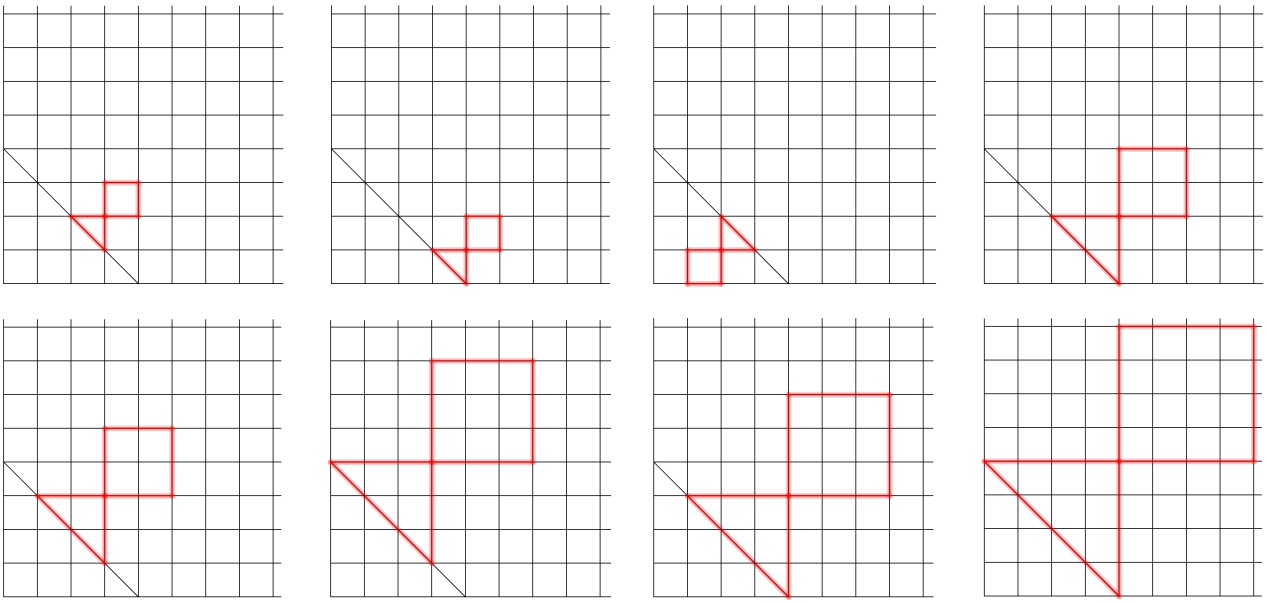
First, the god of fish draws a segment between $(0, 4)$ and $(2, 2)$.

As highlighted in the picture, there are 4 fish in the graph.



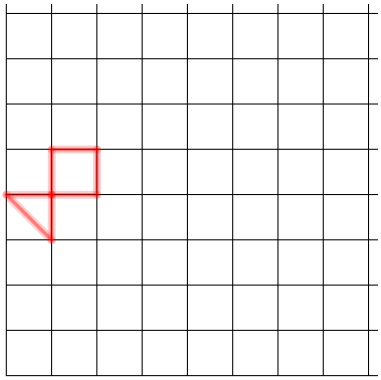
Then, the god of fish draws a segment between $(2, 2)$ and $(4, 0)$.

As highlighted in the picture, there are 8 **new** fish in the graph, so there are 12 fish in total.



Finally, the god of fish erases the segment between $(1, 3)$ and $(4, 0)$.

As highlighted in the picture, there is only 1 fish left in the graph.



Problem K. NIO's Sword

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

NIO is playing a new game. In this game, NIO has a sword with attack power represented by an integer A. Initially, $A = 0$. There are also n enemies in the game. NIO has to kill them in order, which means NIO must kill the $(i-1)$-th enemy before killing the i-th enemy for each i $(2 \le i \le n)$. For the i-th enemy, NIO can kill it if and only if $A \equiv i \pmod{n}$.

Fortunately, NIO can upgrade his sword at any moment for any number of times. Each time NIO upgrades his sword, he first chooses an integer x $(0 \le x \le 9)$, then changes the attack power of the sword from A to $10 \times A + x$.

NIO wants to minimize the number of times to upgrade the sword so that he can win the game as fast as possible. Can you help him to find the minimum times?

Input

The first line contains one integer n $(1 \le n \le 10^6)$, indicating the number of enemies.

Output

Output one integer indicating the minimum times to upgrade the sword.

Example

standard input	standard output
4	4

Note

Here is a possible solution for the example:

- The attack power of the sword when killing the first enemy can be $10 \times 0 + 1 = 1$.
- The attack power of the sword when killing the second enemy can be $10 \times 1 + 4 = 14$.
- The attack power of the sword when killing the third enemy can be $10 \times 14 + 7 = 147$.
- The attack power of the sword when killing the fourth enemy can be $10 \times 147 + 2 = 1472$.

So he needs to upgrade the sword at least four times.

Problem L. Black Hole

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

Equestria is a place full of friendship, magic, and mysteries. Thus, when Twilight Sparkle finds non-spherical black holes in the sky, she is curious rather than surprised.

Twilight Sparkle finds that each of the black holes is a convex regular polyhedron with n faces (a convex polyhedron whose n faces are identical regular polygons), and the length of its edges is a.

Every day the black hole will shrink **exactly once**. If a black hole shrinks, it will become the convex hull of n geometry centers of the original black hole's n faces. And whenever the black hole isn't a regular polyhedron, it will disappear forever.

Twilight Sparkle has made T records for the black holes she has observed. Each of the records can be described as a tuple (n, a, k), indicating that a convex regular polyhedron black hole with edges of length a and n faces initially has shrunk k times. However, naughty Rainbow Dash replaced some of the records with impossible ones. Can you find out the impossible records and calculate the final n' and a' after k times of shrinking for those possible records?

Input

The first line contains an integer T ($1 \le T \le 100$), denoting the number of the records.

In the following T lines, each line describes a record and contains three integers n, a, k ($1 \le n \le 200$, $1 \le a \le 1000$, $1 \le k \le 20$).

Output

For each of the records, if the record is impossible, output `impossible` in a single line. Otherwise, output `possible` followed by an integer n' and a real number a' in a single line, separated by spaces. a' is considered correct if the relative or absolute error between yours and the standard solution is not greater than 10^{-6}.

Example

standard input	standard output
3 2 10 1 4 10 1 6 10 1	impossible possible 4 3.333333333333 possible 8 7.071067811865

Note

For the first record, it's impossible to find a convex regular polyhedron with 2 faces.

For the second record, a 4-face convex regular polyhedron is a tetrahedron. After taking all its faces' geometry centers to make a convex hull, we get a smaller tetrahedron.

For the third record, a 6-face convex regular polyhedron is a cube. After taking all its faces' geometry centers to make a convex hull, we get an octahedron, a.k.a. an 8-face convex regular polyhedron.

Problem M. Monotone Chain

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

In computational geometry, polygon triangulation is the decomposition of a polygonal area into a set of triangles. Over time, a number of algorithms have been proposed to triangulate a polygon. One of the special cases is the triangulation of the **monotone polygon**. It is proved that a monotone polygon can be triangulated in linear time.

A polygon P is called monotone with respect to a straight line L, if every line orthogonal to L intersects the boundary of P at most twice. For many practical purposes, this definition may be extended to allow cases when some edges of P are orthogonal to L.

A monotone polygon can be split into two **monotone polygonal chains**. A polygonal chain is a connected series of line segments, which can be specified by a sequence of points $\{A_1, A_2, \ldots, A_n\}$. Similarly, a polygonal chain C is called monotone with respect to a straight line L, if every line orthogonal to L intersects C at most once, or some line segments of C are orthogonal to L.

In this problem, we denote a **directed line** $L_{\mathbf{a},\mathbf{b}}$ by a point $\mathbf{a}$ and a direction vector $\mathbf{b}$ where $L_{\mathbf{a},\mathbf{b}} = \{\mathbf{a} + \lambda\mathbf{b} | \lambda \in \mathbb{R}\}$. We define a polygonal chain $C = \{A_1, A_2, \ldots, A_n\}$ to be monotone with respect to a directed line $L_{\mathbf{a},\mathbf{b}}$ if the following condition is satisfied:

- For any $1 \le i < j \le n$, $(\mathbf{OA_i} - \mathbf{a}) \cdot \mathbf{b} \le (\mathbf{OA_j} - \mathbf{a}) \cdot \mathbf{b}$.

Here, $\mathbf{OA_i}$ means the coordinate of the point A_i, and "$\cdot$" means the dot product of vectors.

Now given the n points in a polygonal chain C, please determine whether there exists a directed line $L_{\mathbf{a},\mathbf{b}}$ such that C is monotone with respect to $L_{\mathbf{a},\mathbf{b}}$.

Input

The first line contains an integer n ($1 \le n \le 10^5$), indicating the number of points in the polygonal chain.

Each of the next n lines contains two integers x, y ($-10^5 \le x, y \le 10^5$), indicating the coordinates of the points in the polygonal chain. Please note that there may be identical points, even two adjacent points.

Output

If there exists a directed line $L_{\mathbf{a},\mathbf{b}}$ such that the polygonal chain is monotone with respect to $L_{\mathbf{a},\mathbf{b}}$, output `YES` in the first line. Then output four integers x_1, y_1, x_2, y_2 ($-10^9 \le x_1, y_1, x_2, y_2 \le 10^9$, $(x_2, y_2) \ne (0, 0)$) in the second line, indicating that $\mathbf{a} = (x_1, y_1)$ and $\mathbf{b} = (x_2, y_2)$. If there are multiple answers, output any.

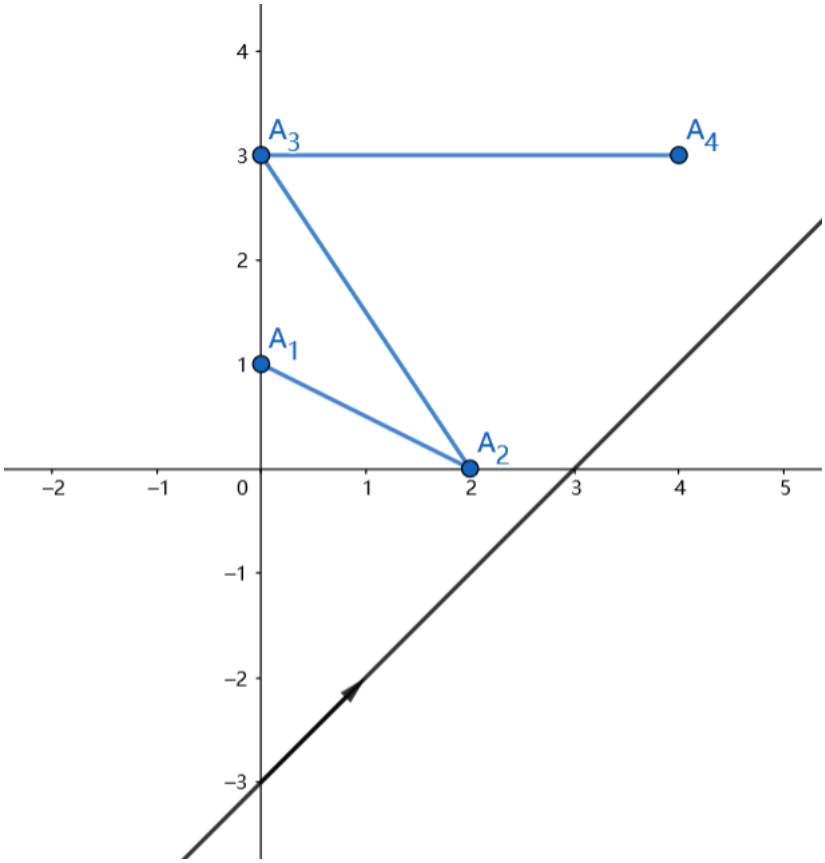
Otherwise, output `NO` in a single line.

Examples

standard input	standard output
4 0 1 2 0 0 3 4 3	YES 0 -3 1 1
5 0 0 0 1 1 1 1 0 0 0	NO

Note

A possible solution for the first sample:



Problem N. Particle Arts

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

In a confined NIO space, there are n NIO particles, the i-th of which has a_i joule energy. The NIO particles are very special as they keep colliding with each other randomly. When one particle carrying energy a joule collides with another particle carrying energy b joule, they will be annihilated and produce two new particles carrying the energy of a `AND` b and a `OR` b respectively. Here `AND` and `OR` mean bitwise AND and OR operation respectively.

The variance of the energy of these particles is obviously not decreasing, but unfortunately, the space here is too small for the author to write down his proof. After enough time the variance of the energy of these particles converges to a stable value. Can you find this value?

The variance of n numbers is defined as follows.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

$$\text{where } \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Input

The first line contains an integer n ($2 \le n \le 10^5$), indicating the number of particles.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{15}$), indicating the enegery of the particles.

Output

Output a irreducible fraction $\frac{a}{b}$ ($b > 0$) in the form of `a/b` that represents the answer. You should ensure that $\gcd(a, b) = 1$ or when $a = 0$, b should be 1.

Example

standard input	standard output
5 1 2 3 4 5	54/5

Note

Warm tip: Please note the use of data types.