Problem A. WPC

RRH is an old driver of our BG. One day, she went out with the little yellow duck to play and came to the front of a grand building.

"Weather prediction center!" exclaimed RRH.

"WPC!" repeated the little yellow duck.

It turns out that our little yellow duckling can only repeat the first letter of each word in its capitalized form. Given a sentence consisting of n words, what will the little yellow duck say if he repeats the sentence?

Input

There are multiple test cases. The first line of the input contains an integer T (about 100), indicating the number of test cases. For each test case:

The first line contains an integer n ($n \geq 1$), indicating the number of words in the sentence.

The second line contains n words separated by a space, indicating the sentence. Each word will only consist of upper-case and lower-case English letters.

It's guaranteed that the total length of each sentence (including the spaces) will not exceed 128.

Output

For each test case output one line containing one string, indicating the first letter of each word in the capitalized form.

Example

standard input	standard output
4 3 Weather Prediction Center 3 weather Prediction center 3 Weather prediction Center 3 weather Prediction Center	WPC WPC WPC WPC

Problem B. Operation on Queue

There is a queue Q with n elements ($q_1, q_2, \ldots, q_n$ from front of the queue to the end of the queue) and a stack S with infinite capacity. You are allowed to perform one of the following operation several times (including zero times):

- pop an element from the front of Q and put it on the top of S.
- pop an element from the top of S and put it to the end of Q.

Now, given two queues $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, you are supposed to find out whether it is possible to make queue A become queue B using the above two operations. It's NOT necessary that the elements are pair-wise distinct in the queue.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^6$), indicating the number of elements in each queue.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), indicating the elements in queue A.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$), indicating the elements in queue B.

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

Output

For each test case, output "Yes" (without quotes) if it is possible to transform queue A to queue B, or "No" (without quotes) otherwise.

Example

standard input	standard output
2 3 1 2 3 3 2 1 3 1 1 2 2 2 1	Yes No

Problem C. Graduation

It is known that school idols of grade three of Aqours are to graduate soon, for which they are going to hold a graduation ceremony at the school hall. During their decoration of the hall, they come up with a problem.

They need to hang flags on a crossbeam of length n. They have square flags of 9 different colors, where each color symbolizes a member of Aquors. The number of flags with each positive integer side length of each color is infinite. They want to put one side of the square flag on the crossbeam such that:

- every place on the crossbeam is exactly covered by a flag, and
- the edge of each flag should not exceed the crossbeam.

The figure below shows an example. The black bar is the crossbeam, and the other three colorful squares are the flags.

But it's not an easy task for them to hang the flags, as there are m positions on the crossbeam which are already rotten. If the rotten position happens to be the boundary of two flags, the crossbeam will break. Therefore, these m positions can't be the boundary of flags.

The *beauty value* of a flag can be calculated by $ax^2 + bx$, where a and b are two given constants, and x is the side length of the flag. The *beauty value* of the crossbeam is the product of all beauty values of the flags on it.

Your task is to help them calculate the total sum of beauty values of all possible crossbeams. Print this sum modulo $(10^9 + 7)$.

Two crossbeams are considered different if there is a position covered by different flags. Two flags are different if they have different side lengths or different colors.

Input

There is only one test case.

The first line of the input contains four integers n $(2 \leq n \leq 10^9)$, m $(0 \leq m \leq 10^5)$, a and b $(0 \leq a, b \leq 10^9)$, indicating the length of the crossbeam, the number of rotten positions and two given constants.

The second line contains m distinct integers $p_1, p_2, \ldots, p_m$ $(1 \leq p_i \leq n-1)$, where p_i indicates that there is a rotten position at a distance of p_i from the left end of the crossbeam.

Output

Output one line containing one integer, indicating the sum of beauty values of all possible crossbeams modulo $(10^9 + 7)$.

Examples

standard input	standard output
3 1 1 0 2	405
3 1 1 1 2	1080
5 2 2 2 3 2	306396
1000000000 0 577 25252	591500023

Note

For the second sample test case, there are two different possible crossbeams:

- Put a flag with side length of 1 and then put a flag with side length of 2. The beauty value of the crossbeam is $(1 + 1) \times (4 + 2) = 12$;
- Put a flag with side length of 3. The beauty value of the crossbeam is $9 + 3 = 12$.

Note that as there is a rotten position at a distance of 2 from the left end of the crossbeam, it's invalid to first put a flag with side length of 2 and then put a flag with side length of 1.

Also note that for every flag we can select its color from 9 different colors, so the total sum of beauty values is $12 \times 9 \times 9 + 12 \times 9 = 1080$.

Problem D. One Cut

Given a convex polygon, calculate the minimum length of the segment and the maximum length of the segment which can divide the convex polygon into two parts with equal areas.

Input

There is only one test case.

The first line of the input contains an integer n ($3 \leq n \leq 10^3$), indicating the number of vertices of the polygon.

For the following n lines, the i-th line contains two integers x_i and y_i ($0 \leq x_i, y_i \leq 10^5$), indicating the x-coordinate and the y-coordinate of the i-th vertex of the polygon. The vertices are given in counter-clockwise order.

It's guaranteed that the polygon is simple and convex, and there are no three adjacent vertices lying on the same line.

Output

Output two lines. The first line contains a number indicating the minimum length of the segment, while the second line contains a number indicating the maximum length of the segment.

Your answer will be considered correct if your absolute or relative error does not exceed 10^{-6}.

Examples

standard input	standard output
4 0 0 10 0 10 10 0 10	10.000000000 14.142135624
3 0 0 6 0 3 10	4.242640687 10.000000000

Note

For the first sample test case, the end points of the segment of minimum length are (5, 0) and (5, 10), while the end points of the segment of maximum length are (0, 0) and (10, 10).

Problem E. Distinct Number

Given n intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$ and an integers x, you should find the size of the set $S = \{y | y = i \text{ AND } x, i \in [l_1, r_1] \cup [l_2, r_2] \cup \cdots \cup [l_n, r_n]\}$, where i AND x means the bitwise *and* operation of i and x.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and x ($1 \leq n \leq 5 \times 10^3, 0 \leq x \leq 10^{18}$).

Each of the next n lines contains two integers l_i and r_i ($0 \leq l_i \leq r_i \leq 10^{18}$).

It is guaranteed that the sum of n over all test cases does not exceed 5×10^3.

Output

For each test case, output an integer denoting the size of the S.

Example

standard input	standard output
3 2 1 1 2 343 34345 1 3 1 3 1 123242343 1 1000000000000000000	2 3 32768

Note

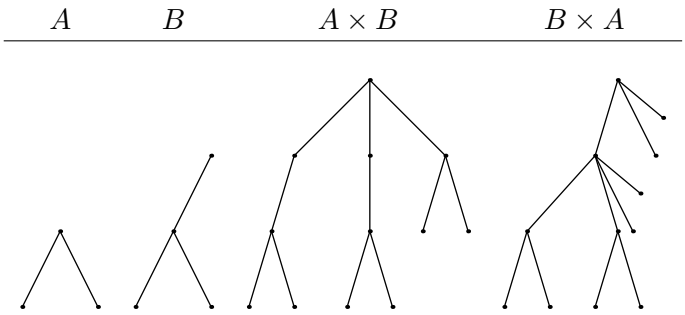
For the first sample test case, it's obvious that $S = \{0, 1\}$, so the answer is 2.

For the second sample test case, it's also obvious that $S = \{1, 2, 3\}$, so the answer is 3.

Problem F. Tree Product

Given n rooted trees $T_1, T_2, \ldots, T_n$, find two permutations $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_n$ such that the diameter of $T_{p_1} \times T_{p_2} \times \cdots \times T_{p_n}$ is maximum and the diameter of $T_{q_1} \times T_{q_2} \times \cdots \times T_{q_n}$ is minimum.

For two rooted tree A and B, $T = A \times B$ is built by merging the root of B with each vertex $x \in A$ so that all the subtrees of the root of B become new subtrees of x. See the table below more details:



Recall that

- A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. The parent of a vertex v is the last vertex different from v on the path from the root to v.
- The diameter of a rooted tree is the length of the longest simple path in the tree, where the length of a path is equal to the number of the edges in the path.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

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

Each of the next n lines starts from an integer m_i ($1 \le m_i \le 10^5$), indicating the number of vertices in the i-th rooted tree, and then m_i integers $p_{i,1}, p_{i,2}, \cdots, p_{i,m_i}$ ($0 \le p_{i,j} \le m_i$) follow, where the j-th of them denotes the parent of the j-th vertex and the root of the tree has 0 as parent.

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

Output

For each test case, output two integers denoting the length of the maximum and the minimum diameter.

Example

standard input	standard output
2 3 5 0 1 2 1 4 3 2 0 2 2 2 0 2 1 0 1 0	8 7 0 0

Note

For the first sample test case, $T_1 \times T_2 \times T_3$ will provide the maximum diameter, while $T_3 \times T_2 \times T_1$ will provide the minimum diameter.

Problem G. Another Strange Function

If a string s is composed of digits from 0 to 9, we call this string a "digit string". It's obvious that each substring of s can be considered as a non-negative integer (note that leading zeros are thrown away when parsing strings to integers, for example, we parse "00123" to 123, and "000" to 0). Let $\mathbb{S}$ be the set containing all these non-negative integers, define a function

$$f(s) = \min\{x | x \in \mathbb{N}, x \in \mathbb{S}\}$$

That is to say, $f(s)$ is the smallest non-negative integer which is in $\mathbb{S}$. For example, $f(\text{"1241"}) = 1$, $f(\text{"542"}) = 2$, $f(\text{"000321123"}) = 0$, and $f(\text{"00000"}) = 0$.

Given a digit string s, let $\text{substr}(i, j)$ be the substring of s which starts from the i-th digit and ends at the j-th digit, your task is to calculate

$$\sum_{i=1}^{|s|} \sum_{j=i}^{|s|} f(\text{substr}(i, j))$$

.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first and only line contains a digit string s ($1 \le |s| \le 5 \times 10^5$).

It's guaranteed that the sum of $|s|$ of all test cases will not exceed 5×10^6.

Output

For each test case output one line containing one integer, indicating the answer.

Example

standard input	standard output
4 1241 542 000321123 00000	15 19 29 0

Note

We explain the first sample test case below.

Substring	Value	Substring	Value
"1"	1	"24"	2
"2"	2	"41"	1
"4"	4	"124"	1
"1"	1	"241"	1
"12"	1	"1241"	1

So the answer is $1 + 2 + 4 + 1 + 1 + 2 + 1 + 1 + 1 + 1 = 15$.

Problem H. Attention Machanism

Human perception focuses selectively on partsof the scene to acquire information at specific places and times. In machine learning, this kind of process is referred to as attention mechanism, and has drawn increasing interest when dealing with languages, images and other data. Integrating attention can potentially improve performance.

Now you are coding for a new platform that does not support machine learning libraries due to some unknown hardware limitations, so some basic functions need to be re-implemented, and your job is to implement the attention module. The input data and parameters of the attention module are given in vectors, and output under attention defined as follows:

$$\alpha_{ts} = \frac{\exp\left(\langle \mathbf{h}_t, \mathbf{x}_s \rangle\right)}{\sum_{s'=1}^{n} \exp\left(\langle \mathbf{h}_t, \mathbf{x}_{s'} \rangle\right)} \quad \mathbf{c}_t = \sum_s \alpha_{ts} \mathbf{x}_s$$

In the above formula, vectors $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n} \in \mathbb{R}^d$ are called input features, while vectors $\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_m} \in \mathbb{R}^d$ are called parameters. Variable t ranges from 1 to m (both inclusive) and s ranges from 1 to n (both inclusive). The result of vector inner product $\langle \cdot, \cdot \rangle$ is a scalar: $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$, where x_i is the i-th element in vector $\mathbf{x}$ and y_i is the i-th element in vector $\mathbf{y}$.

Given the input features and the parameters, your task is to calculate the outputs $\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_m}$.

Input

There is only one test case.

The first line contains three integers n, m and d ($1 \le n, m, d \le 100$).

For the following n lines, each line contains d numbers as a vector of input features.

For the following m lines, each line contains d numbers as a vector of attention parameters.

It's guaranteed that all the elements is not NaN and the absolute values are not larger than 10.

Output

Output m lines. The i-th line contains d numbers separated by a single space, indicating the value of $\mathbf{c_i}$. It's guaranteed that NaN would never appear in the output.

Your answer will be considered correct if your absolute or relative error does not exceed 10^{-6}.

Example

standard input	standard output
3 3 2 1.0 0.8 2.0 -1.0 -1.0 0.2 1.0 -1.0 0.1 0.9 1.1 -0.2	1.901658 -0.881528 0.566279 0.399045 1.746857 -0.640958

Problem I. A + B Problem

Consider adding up two binary integers $A = a_p a_{p-1} \dots a_2 a_1$ and $B = b_q b_{q-1} \dots b_2 b_1$ in a circuit, where a_i indicates the i-th least significant bit of A and b_i indicates the i-th least significant bit of B. To complete the addition operation, an electronic component, called the *adder*, is needed.

An n-bit adder is composed of n *full adder*s, where the i-th full adder has three inputs a_i, b_i, c_i^{in} and two outputs s_i and c_i^{out}. Their definitions are given below. Note that in the definitions below, we use $\wedge$ to indicate the logical *and* operation, $\vee$ to indicate the logical *or* operation and $\oplus$ to indicate the logical *exclusive or* operation.

- a_i and b_i: The i-th least significant bits of the binary integers to be added. For completeness, $a_i = 0$ if $i > p$ and $b_i = 0$ if $i > q$.
- c_i^{in}: The bit carried from the previous full adder. That is to say, $c_i^{\text{in}} = c_{i-1}^{\text{out}}$ if $i > 1$, and $c_1^{\text{in}} = 0$.
- s_i: The i-th least significant bit of the result. $s_i = a_i \oplus b_i \oplus c_i^{\text{in}}$.
- c_i^{out}: The bit carried to the next full adder. $c_i^{\text{out}} = (a_i \wedge b_i) \vee (c_i^{\text{in}} \wedge (a_i \vee b_i))$.

After feeding the two binary integers A and B into the n-bit adder, we can easily get the addition result $S = s_n s_{n-1} \dots s_2 s_1$ as a binary integer, where s_i is the i-th least significant bit of S.

But unfortunately, k out of n full adders of the n-bit adder are broken and we don't know their exact indices! What we only know is that, for these k full adders, instead of giving out the correct c_i^{out}, their c_i^{out} will always be 0.

Given two binary integers A and B, what's the minimum and maximum possible result if we use this broken adder to calculate $A + B$?

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and k ($2 \le n \le 5 \times 10^5$, $0 \le k \le n$), indicating the size of the adder and the number of broken full adders.

The second line contains a binary integer $A = a_p a_{p-1} \dots a_2 a_1$ without leading zeros ($1 \le p < n$), indicating the first binary integer to be added.

The third line contains another binary integer $B = b_q b_{q-1} \dots b_2 b_1$ without leading zeros ($1 \le q < n$), indicating the second binary integer to be added.

It's guaranteed that the sum of n of all test cases will not exceed 5×10^6.

Output

For each test case output two lines. The first line contains a binary integer without leading zeros, indicating the minimum possible result if we use the broken adder to calculate $A+B$. The second line contains another binary integer without leading zeros, indicating the maximum possible result if we use the broken adder to calculate $A + B$.

Example

standard input	standard output
3 6 1 10110 110 5 4 1011 11 100 1 1 1	10100 11100 1000 1100 0 10

Note

We explain the sample test cases below.

For the first sample test case, if the 3rd full adder is broken, we can achieve the smallest result; If the 1st full adder is broken, we can achieve the largest result.

i	a_i	b_i	$c_i^{\mathbf{in}}$	$c_i^{\mathbf{out}}$	s_i
1	0	0	0	0	0
2	1	1	0	1	0
3	1	1	0	0 (broken)	1
4	0	0	0	0	0
5	1	0	0	0	1
6	0	0	0	0	0

i	a_i	b_i	$c_i^{\mathbf{in}}$	$c_i^{\mathbf{out}}$	s_i
1	0	0	0	0 (broken)	0
2	1	1	0	1	0
3	1	1	0	1	1
4	0	0	1	0	1
5	1	0	0	0	1
6	0	0	0	0	0

For the second sample test case, if the 1st, 2nd, 3rd and 4th full adder is broken, we can achieve the smallest result; If the 1st, 3rd, 4th and the 5th full adder is broken, we can achieve the largest result.

i	a_i	b_i	$c_i^{\mathbf{in}}$	$c_i^{\mathbf{out}}$	s_i
1	1	1	0	0 (broken)	0
2	1	1	0	0 (broken)	0
3	0	0	0	0 (broken)	0
4	1	0	0	0 (broken)	1
5	0	0	0	0	0

i	a_i	b_i	$c_i^{\mathbf{in}}$	$c_i^{\mathbf{out}}$	s_i
1	1	1	0	0 (broken)	0
2	1	1	0	1	0
3	0	0	1	0 (broken)	1
4	1	0	0	0 (broken)	1
5	0	0	0	0 (broken)	0

For the third sample test case, if the 1st full adder is broken, we can achieve the smallest result; If the 100th full adder is broken, we can achieve the largest result. Note that you should output the answers without leading zeros.

Problem J. Game of Life

The Game of Life, also known simply as Life, is a cellular automaton devised By the British mathematician John Horton Conway in 1970 – Wikipedia

In this problem, we will deal with *OR* game of life. Given a infinite board, each cell has an initial state *live* or *dead*. The game will last for n seconds. At the beginning of each second, each cell (say we're considering cell (x, y)) interacts with its four neighbors $(x+1, y)$, $(x, y+1)$, $(x-1, y)$ and $(x, y-1)$ using the following rules:

1. Any cell with at least one live neighbors becomes a live cell.
2. Any cell with no live neighbors becomes a dead cell.

As you can see, it is possible to predict the state at any moment by the initial state. Let $f(i)$ be the number of live cells at the end of the i-th second (for completeness, we define $f(0)$ as the number of live cells in the initial state), given an initial state with m live cells, you need to calculate $\sum_{i=0}^{n} f(i)$.

Input

There is only one test case.

The first line of the input contains two integers n ($1 \leq n \leq 10^9$) and m ($1 \leq m \leq 100$), indicating the number of seconds the game lasts and the number of initial live cells.

For the following m lines, the i-th line contains two integers x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$), indicating the coordinate of the i-th initial live cell.

It's guaranteed that $x_i + y_i$ is even for all initial live cells and the initial live cells are distinct.

Output

Output one line containing one integer, indicating the answer. As the answer may be very large, print it modulo 998244353.

Examples

standard input	standard output
5 3 0 0 0 2 2 2	153
100 4 0 0 2 2 4 4 6 6	379454