Problem A. Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Little A has an array a of length n, which is indexed from 1 to n. A total of n numbers are stored in $a_1, a_2, \cdots, a_n$, and little A wants to sort them in non-decreasing order.

Little A can perform several rounds of operation. In the i-th round, Little A will choose v_i+1 non-negative numbers $b_{i,0}, b_{i,1}, \cdots, b_{i,v_i}$, and an array $p_i = (p_{i,1}, p_{i,2}, \cdots, p_{i,v_i})$ of length v_i, storing a permutation of 1 to v_i. Little A should ensure the following properties:

* $2 \le v_i \le k$, where k is a constant set by Little A.

* $\forall 1 \le j \le v_i, b_{i,j} > b_{i,j\text{-}1}$.

* $b_{i,0} = 0$ and $b_{i,v_i} = n$.

According to $(b_{i,0}, b_{i,1}, \cdots, b_{i,v_i})$, Little A will split the array a into v_i continuous segments, and the j-th one c_j of length $b_{i,j} - b_{i,j\text{-}1}$ consists of $a_{b_{i,j\text{-}1}+1}, a_{b_{i,j\text{-}1}+2}, \cdots, a_{b_{i,j}}$. After that, he will connect the continuous segments in the order of $c_{p_{i,1}}, c_{p_{i,2}}, \cdots, c_{p_{i,v_i}}$ to get a new array. Little A called the whole process above a round of operations. And the new array generated will be used as the array a in the next round.

Little A wonders whether the array a can be in non-decreasing order by finite rounds of operation. And if there is a solution to sort it, he wants the one that satisfies $\sum v_i \le 3n$, so that he can simply simulate the whole process.

Input

The input consists of multiple test cases.

The first line contains an integer T – the number of test cases.

For each test case:

In the first line, there are two integers n and k, which represent the length of the array and the constant that Little A set in this problem.

In the second line, there are n integers, and the i-th one represents a_i.

It is guaranteed that $1 \le T \le 1000, 1 \le k \le n \le 1000, 1 \le a_i \le 10^9$, and the sum of n in all test cases does not exceed 30000.

Output

For each test case:

If there is no way to sort the array in finite rounds of operations, output -2 in the first line.

If there exist some ways to sort the array in finite rounds of operations, but none of them satisfies $\sum v_i \le 3n$, output -1 in the first line.

If there exists a way to sort the array in finite rounds of operations, and $\sum v_i \le 3n$ holds, then:

The first line is a non-negative integer m, which represents the number of rounds in your solution. The following $3m$ lines contain the information about each round of operations.

For each round of operations:

The first line is a positive integer v_i $(1 \le v_i \le k)$, which represents the number of continuous segments you split in the operation.

The second line contains $v_i + 1$ non-negative numbers, and the j-th one represents $b_{i,j\text{-}1}$. You need to guarantee that b_i satisfies all the properties described in the statement.

The third line contains v_i positive numbers, and the j-th one represents $p_{i,j}$. You need to guarantee that

p_i is a permutation of 1 to v_i.

If there are multiple valid constructions, any of them is correct.

Example

standard input	standard output

5
1 1
1
2 1
2 1
3 2
3 1 2
5 3
1 2 3 4 5
5 5
5 4 3 2 1
	0
	-2
	1
	2
	0 1 3
	2 1
	0
	1
	5
	0 1 2 3 4 5
	5 4 3 2 1

Problem B. Mailman

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

Little A is the only mailman in city P, and he has to deliver lots of mails every day.

The traffic network in city P can be simply regarded as an undirected weighted grid graph with n rows and m columns. A vertex in the graph represents a crossroads, while an edge represents a road. For convenience, Little A calls the crossroads in the northeast corner $(1, 1)$ and the one in the southwest corner (n, m). Every road has its own length.

Every day Little A has to start delivering mails from the post office at crossroads $(1, 1)$, pass each road at least once, and return to the post office at crossroads $(1, 1)$. The distance travelled by Little A is the sum of the lengths of all the roads that he passes through. Note that Little A cannot change his moving direction in the middle of a road, and a road that is passed through multiple times is also counted multiple times in the distance.

Recently, there are Q underground food courts being built in city P. The i-th one connects the crossroads $(x1_i, y1_i)$ and $(x2_i, y2_i)$ and has a length of v_i. As a large number of citizens pour into the food courts, Little A can only pass through each underground food court exactly once when delivering mails every day.

After each underground food court is put into use, Little A wants to know the minimum distance he takes to deliver mails every day.

Input

The first line contains three positive integers n, m, Q, which represent the number of rows and columns in the traffic network and the number of the underground food courts.

Each of the following $n - 1$ lines contains m positive integers. The j-th integer in the i-th line represents the length of the road that connects crossroads (i, j) and $(i + 1, j)$.

Each of the following n lines contains $m - 1$ positive integers. The j-th integer in the i-th line represents the length of the road that connects crossroads (i, j) and $(i, j + 1)$.

Each of the following Q lines contains 5 positive numbers $x1_i, y1_i, x2_i, y2_i, v_i$, where the i-th line describes the i-th underground food court that is put into use.

It is guaranteed that $2 \le n \le 3, 2 \le m \le 5 \times 10^4, 1 \le Q \le 5 \times 10^4, 1 \le x1_i, x2_i \le n, 1 \le y1_i, y2_i \le m$. And the length of any roads and underground food courts is a positive number not exceeding 10^9.

Output

Output Q lines. The i-th line contains an integer, which represents the minimum distance he takes if the first i underground food courts are put into use while others are not.

Example

standard input

```
3 4 5
19 26 8 17
19 49 10 1
9 9 8
2 4 4
3 5 3
1 2 3 3 11
2 1 3 2 13
1 3 3 1 17
1 1 2 2 19
2 1 2 3 23
```

standard output

```
230
232
249
289
314
```

Problem C. Range Subsequence

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

The following describes four notations related to this problem.

- Sequence $s = [s_1, \dots, s_m]$ is a *subsequence* of sequence $t = [t_1, \dots, t_n]$ if there exists index sequence $[a_1, \dots, a_m]$ such that (1) $1 \le a_1 < a_2 < \cdots < a_m \le n$, and (2) $\forall i \in [1, m], s_i = t_{a_i}$.
- Sequence $s = [s_1, \dots, s_m]$ is a *range* if $\forall i \in [1, m], s_i = s_1 + i - 1$.
- Sequence s is a *range subsequence* of sequence t if (1) s is a subsequence of t, and (2) s is a range.
- Two sequences $s = [s_1, \dots, s_m]$ and $t = [t_1, \dots, t_n]$ are *different* if $n \ne m$ or there exists index $i \in [1, m]$ such that $s_i \ne t_i$.

Now, given a sequence $x = [x_1, \dots, x_n]$, your task is to answer m queries. In each query, an interval $[l, r]$ is provided, and you need to calculate the number of **different range subsequences** of $[x_l, \dots, x_r]$.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ – the length of sequence x and the number of queries.

The second line contains n integers $x_1, \dots, x_n$ $(1 \le x_i \le n)$.

Then m lines follows. Each line contains two integers l_i and r_i $(1 \le l_i, r_i \le n)$ representing a query.

Output

For each query, output a single line with a single integer, the corresponding answer.

Examples

standard input	standard output

5 5
1 2 3 2 4
1 3
2 4
2 5
1 5
3 5
	6
	3
	6
	10
	4

14 14
14 12 10 11 7 6 7 11 8 13 5 6 4 2
9 11
4 6
1 4
3 4
6 13
10 10
1 14
7 14
9 10
12 12
2 2
8 14
5 14
7 10
	3
	3
	5
	3
	11
	1
	17
	10
	2
	1
	1
	8
	12
	5

Problem D. Linear Algebra

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Baby Give is a sophomore. He has just learned about polynomials in the course "Discrete Mathematics", and he is currently learning more about polynomials in the course "Linear Algebra". Now he finds his favourite polynomial $f(x) \in \mathbb{F}_p[x]$ with degree n and leading coefficient 1, where p is a prime. (Please refer to "Problem I. Discrete Mathematics" for the definition of $\mathbb{F}_p[x]$.) Then he multiplies $f(x)$ with $f(x+1)$ to get a new polynomial $g(x)$ with degree $2n$, in other words, $g(x) = f(x)f(x+1)$. However, Baby Give forgets $f(x)$ after he has calculated $g(x)$. So he asks you to help him recover $f(x)$.

In this problem, $p = 998244353 = 2^{23} \times 7 \times 17 + 1$.

Input

The first line contains one integer n, meaning the degree of g.

The second line contains $2n+1$ integers $a_0, \dots, a_{2n}$, separated by a space, meaning the coefficients of $x^0, \dots, x^{2n}$.

It is guaranteed that $1 \le n \le 100000$, $0 \le a_i \le p-1$, and $a_{2n} = 1$.

Output

In the first line, output "Yes" (without quotes) if there exists a solution f, and output "No" otherwise.

If there exists a solution f, output $n+1$ integers separated by a space in the second line. They represent the the coefficients of $x^0, \dots, x^n$ in $f(x)$. There should be no extra spaces at the end of the line.

Examples

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

Problem E. Nearest Point

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

Nearest point is a classical problem in computational geometry. Given a set S of n points $p_1, \ldots, p_n$, p_j is the nearest point of p_i if (1) $j \neq i$, and (2) for any other point p_k $(k \notin \{i, j\}), dis(p_i, p_j) \leq dis(p_i, p_k)$, where $dis(p_1, p_2)$ is defined as $\sqrt{(p_1.x - p_2.x)^2 + (p_1.y - p_2.y)^2}$ in a 2-D space.

Calculating the neatest point is known to be difficult. Therefore, there are many heuristic methods proposed. The following describes one of these algorithms:

- **Step1:** A random angle α is uniformly selected from range $[-\pi, \pi)$.
- **Step2:** All points in S are rotated α counterclockwise centered on the origin $(0, 0)$.
- **Step3:** For each point p_i, the algorithm takes the point that is closest to p_i on the x-coordinate, i.e., the point p_j $(i \neq j)$ that minimizes $|p_i.x - p_j.x|$. If there are multiple such points, the point with the smallest index will be selected.

For example, suppose there are three points $p_1 = (1, 1)$, $p_2 = (3, 3)$, and $p_3 = (0, 2)$ in set S.

1. When α is selected as 0, the coordinates of these three points after the rotation are $(1, 1), (3, 3), (0, 2)$, respectively, and thus the nearest points to p_1, p_2, p_3 found by the algorithm are p_3, p_1, p_1, respectively.
2. When α is selected as $\frac{\pi}{4}$, the coordinates of these three points after the rotation are $(0, \sqrt{2}), (0, 3\sqrt{2}), (-\sqrt{2}, \sqrt{2})$, and thus the nearest points are p_2, p_1, p_1, respectively.

Now, given the n points $p_1, \ldots, p_n$ in S, your task is to output an $n \times n$ matrix w, where $w_{i,j}$ represents the probability for the algorithm to take p_j as the nearest point of p_i.

Input

The first line contains a single integer n $(2 \leq n \leq 50)$, the number of points in S.

Then n lines follow, each line contains two integers x_i and y_i $(|x_i|, |y_i| \leq 1000)$.

The input guarantees that each given point is different from the other.

Output

Output n lines, each with n float numbers. The j-th number in the i-th line represents the probability for p_j to be returned as the nearest point by the algorithm.

Note that for each i, the i-th number in the i-th line is always 0.

Your result is judged as correct if for each probability, either the relative error or the absolute error comparing with the standard output is at most 10^{-6}.

Note: The space at the end of a line will be ignored.

Examples

standard input	standard output

3
1 1
3 3
0 2
	0.00000000 0.29516721 0.70483279
	0.57797916 0.00000000 0.42202084
	0.75000004 0.24999996 0.00000000

3
1 1
2 2
3 3
	0.00000000 1.00000000 0.00000000
	1.00000000 0.00000000 0.00000000
	0.00000000 1.00000000 0.00000000

5
8 10
9 75
810 9
7 5
810 975
	Please see the online statement.

Problem F. Leapfrog

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Leapfrog is a new minion added to the tavern in the new version. Its mechanism is so imbalanced that all players have to employ it for a higher ranking score.

Rikka has bought n leapfrogs. Each leapfrog can be described by a pair (a, D), where a represents the attribute, and D is a **list** of deathrattles, where each deathrattle is described by a pair (b, t).

Initially, the i-th leapfrog is $(a_i, [(b_i, t_i)])$, representing that (1) its attribute is a_i, and (2) it has only one deathrattle, represented by (b_i, t_i).

Now, a battle begins. The battle proceeds for $n-1$ rounds. In each round, a leapfrog dies (which is selected randomly among alive ones) and its deathrattles are triggered **in order**. For each deathrattle (b, t), the following process proceeds for t times:

- A leapfrog is selected randomly among those alive ones. (In the i-th round, there are exactly $n - i$ alive leapfrogs).
- For the selected leapfrog, the attribute is increased by b, and deathrattle (b, t) is added to the end of the list of deathrattles.

For example, suppose there are 3 leapfrogs $(1, [(1, 2)]), (2, [(2, 2)]), (1, [(1, 3)])$ initially. The following describes one possible procedure of the battle.

- In the first round, the third leapfrog dies, and leapfrog indexed $1, 2, 1$ are selected in order when dealing with the deathrattle of the third leapfrog. At the end of this round, the first two leapfrogs become $(3, [(1, 2), (1, 3), (1, 3)])$ and $(3, [(2, 2), (1, 3)])$.
- In the second round, the second leapfrog dies. At this time, only the first leapfrog is alive and thus it is always selected while dealing with the deathrattles. At the end of this round, the first leapfrog becomes $(10, [(1, 2), (1, 3), (1, 3), (2, 2), (2, 2), (1, 3), (1, 3), (1, 3)])$.

Clearly, after $n-1$ rounds, there must be a single leapfrog remained. Your task is to calculate the maximum possible attribute of this leapfrog.

Input

The first line contains an integer t $(1 \le t \le 1000)$, representing the number of testcases.

For each testcase, the first line contains an integer n $(1 \le n \le 10^5)$, representing the number of leapfrogs.

Then n lines follow, each line contains 3 integers a_i, b_i, t_i $(1 \le a_i, b_i \le 10^4, t_i \in \{2, 3\})$, representing the initial status of the i-th leapfrog.

The input guarantees that the sum of n is no larger than 10^5.

Output

For each testcase, output a single line with a single integer, the answer module 998244353.

Example

standard input	standard output

3
3
1 1 2
2 2 2
1 1 3
3
1 1 2
2 2 2
10 1 3
3
1 1 2
2 10 2
1 1 3
	14
	20
	44

Problem G. Limit

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given $2n$ integers, $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$, and an integer t. You need to calculate:

$$\lim_{x \to 0} \frac{\sum_{i=1}^{n} a_i \cdot \ln(1 + b_i \cdot x)}{x^t}.$$

Input

The first line consists of two integers n, t.

In the following n lines, the i-th line consists of two integers a_i, b_i.

$1 \le n \le 100000, -100 \le a_i, b_i \le 100, 0 \le t \le 5.$

Output

Please output the result of this limit. If the result is ∞, please output "infinity"(without quotes). And if the result is an integer, please output this integer directly. Otherwise, the answer must be $\frac{a}{b}$, such that a and b are coprime and $b \ge 2$, please output "a/b".

Examples

standard input	standard output

2 2
1 1
1 -1
	-1

2 1
1 1
1 -1
	0

2 3
1 1
1 -1
	infinity

Problem H. Set

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

You are given a set $S = \{1, 2, 3, \ldots, 256\}$ with 256 elements and two numbers k, r such that $3 \le r \le k \le 26$.

And you need to construct k sets $I_1, I_2, I_3, \ldots, I_k$, such that:

- Every set is a subset of S, i.e., $I_1, I_2, \ldots I_k \subseteq S$.
- The union of any r subsets has at least 128 elements, i.e., for every $1 \le i_1 < i_2 < i_3 < \cdots < i_r \le k$, we have
$$|\bigcup_{u=1}^{r} I_{i_u}| \ge 128.$$
- For every $1 \le i \le k$, we have $|I_i| \le \lceil \frac{512}{r} \rceil$.

Input

The input consists of two integers, k, r. $3 \le r \le k \le 26$.

Output

The output consists of k lines.

In the i-th line, you need to output a 01 string with length 256. If $j \in I_i$, then you need to output 1 at the j-th position of this string; otherwise you need to output 0 at the j-th position.

If there is no solution, please output -1.

Note

Sample Input

```
3 3
```

Sample Output

(Due to the page width limit, the sample output with 3 lines are printed as 9 lines below. It may not be difficult to guess the actual locations of line breaks from the following layout.)

```
1110001011001011110110011111011100111111010011010111111110111011111010111110111111010011
1110101011110111010011000010110111111101000011011111011001101111101000111111100011010110
0111100111100001011110011110110010011111111100111111001111110111001101111101001
1010111110001111000101111010100110000101111101111111111100101111110101110111111111110011
1011111100101110111111001011111110110000001111010010011011111100111111111111100000110001
1111101101011101010000110001101101101001101100101000111111011011111111111011111
1111010101110100000111111011111101111011001111011011111000001011111101110011100000111111
1101111111101110110001111111111100011100111101001111111111111110110110111110000110101010
0011110100101011011110010011001001110111011110000101111110111111101010110110011 0
```

Problem I. Discrete Mathematics

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

Given a set F with two binary operations defined on F: $+$ and $\times$, we call F a field, if it satisfies the following properties:

- $\forall x, y \in F, x + y = y + x$
- $\forall x, y, z \in F, x + (y + z) = (x + y) + z$
- $\forall x, y \in F, x \times y = y \times x$
- $\forall x, y, z \in F, x \times (y \times z) = (x \times y) \times z$
- $\forall x, y, z \in F, x \times (y + z) = x \times y + x \times z$
- There exist two different elements 0 and 1 in F
- $\forall x \in F, x + 0 = x = 0 + x$
- $\forall x \in F, x \times 1 = x = 1 \times x$
- $\forall x \in F$, there exists y, such that $x + y = 0$
- $\forall x \in F \backslash \{0\}$, there exists y, such that $x \times y = 1$

An example of field is a finite field that consists of $\{0, 1, \cdots, p-1\}$, where p is a prime number. Denote this field as $\mathbb{F}_p$. The addition $\oplus$ and multiplication $\otimes$ in $\mathbb{F}_p$ are defined as follows:

- $\forall x, y \in \mathbb{F}_p, x \otimes y = (x \times y) \mod p$
- $\forall x, y \in \mathbb{F}_p, x \oplus y = (x + y) \mod p$

Define $\mathbb{F}_p[x]$ to be the set of polynomials over $\mathbb{F}_p$, and we can define addition and multiplication on $\mathbb{F}_p[x]$ similar to that on polynomials over real numbers.

A non-constant polynomial $f \in \mathbb{F}_p[x]$ is called **irreducible** if and only if it cannot be written as the product of two non-constant polynomials $g, h \in \mathbb{F}_p$. Then by definition, we can always factorize all polynomials in $\mathbb{F}_p[x]$ into the pattern of $c \times \prod g_j$, where c is a constant, and g_j are irreducible polynomials with leading coefficient 1. We can prove the unique factorization property on $\mathbb{F}_p[x]$, that is the factorization above is unique.

Little A is very curious about how many irreducible polynomials over $\mathbb{F}_p$ with degree n there are. Furthermore, he requires that the leading coefficient is 1, and the coefficient of x^{n-1} is k. Because the answer is very large, you only need to output the result module 998244353.

To help you solve this problem, little A tells you the following theorem: the number of irreducible polynomials over $\mathbb{F}_p$ with degree n, such that the coefficient of the highest term is 1, is $\frac{1}{n}\sum_{i|n}\mu(i)p^{n/i}$. Here μ represents the Mobius function, and it is defined as follows:

$$\mu(n) = \begin{cases} 0 & \text{if } n \text{ has a squared prime factor.} \\ 1 & \text{if } n \text{ is a square-free positive integer with an even number of prime factors.} \\ -1 & \text{if } n \text{ is a square-free positive integer with an odd number of prime factors.} \end{cases}$$

Input

The first line contains three integers n, k, and p, separated by a space.

It is guaranteed that $5 \le p \le n \le 2500$, $0 \le k < p$, and p is a prime.

Output

The answer to little A is the total number of irreducible polynomials with degree n in $\mathbb{F}_p[x]$, such that the leading coefficient is 1, and the coefficient of x^{n-1} is k.

Output a single integer, which represents the answer taking the module of 998244353.

Examples

standard input	standard output
6 2 5	516
11 1 11	361458985
646 8 17	187850719
2499 0 17	710706754

Problem J. Leaking Roof

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Grandpa Are lives in an old house. Now it is the heavy rain that occurs only once in a hundred years, but the roof of Grandpa Are's house leaks. His roof can be viewed as a $n \times n$ grid, and each square has a height. Denote the height of the square with coordinate (i, j) by $h_{i,j}$ meters. Now the rainfall is m millimeters, meaning that the incoming rain on every square is m millimeters.

Since the roof is not flat, the water flows, and the water on a square can only flow to an adjacent square with a strictly smaller height. Two squares are adjacent when they share a common edge. Also, since the water level is subtle compared with the difference in heights of the squares, water will **NOT** flow to adjacent squares with the same height due to surface tension. Furthermore, the water equally divides among all possible directions. Besides, the roof leaks. To be specific, if a square has a height of 0, it leaks.

Grandpa Are wants to know how much water flows to each leaking square after a long time. The water flows to a square includes the one from the incoming rain and the one from the adjacent squares. He asks you to calculate for him.

Input

The first line contains two integers n and m.

The next n lines contains n integers in each line. The i-th line contains integer $h_{i,1}, \dots, h_{i,n}$, separated by a space, meaning the height of the squares in the i-th row. There **is a space in the end of each line, and there is an eoln in the end of the n-th line.**

It is guaranteed that $1 \le n \le 500$ and $0 \le m, h_{i,j} \le 10000$.

Output

There are n lines in the output, and each line contains n real numbers. The i-th line contains $a_{i,1}, \dots, a_{i,n}$, separated by a space, meaning the height of water flows to the corresponding square at last, or 0 if the square is not leaking.

Your answer will be considered correct if and only if the absolute or relative error of your answer to the correct answer is less than or equal to 10^{-6}. Also, there is a space in the end of each line, and there is an eoln in the end of the n-th line.

Examples

standard input	standard output

3 1
0 1 2
1 2 3
2 3 4
	9.000000 0 0
	0 0 0
	0 0 0

3 1
0 1 0
1 1 1
0 1 0
	2.000000 0 2.000000
	0 0 0
	2.000000 0 2.000000

Note

In the first sample, all water flows to the leaking square (1,1).

In the second sample, notice that the water in the square (2,2) will NOT flow. Although it is not the case in real life, it is the case in this problem due to the imaginary surface tension.

Problem K. Meal

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

Time for meals has passed. The restaurant in T University has only n dishes remained, and each dish can only serve one hungry student.

n hungry students line up to get dishes, and everyone has their own preferences for each dish. For the i-th student lining up in the queue, his preference for the j-th dish is $a_{i,j}$.

Before getting dishes, every student will generate today's preference sequence according to their preference for different dishes. The preference sequence of the i-th student, denoted as p_i, will be generated in the following way:

1. Initially, there is a set S including all n dishes, and p_i is empty.
2. The i-th student randomly draws a dish x from S. For each dish j in S, the probability for it to be drawn is $\frac{a_{i,j}}{\sum_{k \in S} a_{i,k}}$.
3. x is inserted at the end of sequence p_i, and is deleted from set S.
4. If S is empty, then the generation procedure ends. Otherwise, it returns to the second step.

Students will get dishes one by one, and everyone will choose the dish that is not taken by other students and placed as high as possible in the preference sequence (which means the dish that has the smallest index in his preference sequence).

Little A has finished his lunch, and he is curious about the probability that the i-th student getting the j-th dish. Because it is such a complex answer, you should only tell the result taking the modulo of 998244353.

Input

The first line contains a positive number n.

In the following n lines, each line contains n positive numbers. The j-th number in the i-th line represents $a_{i,j}$.

It is guaranteed that $1 \le n \le 20$ and $1 \le a_{i,j} \le 100$.

Output

Output n lines, each of which contains n numbers separated by spaces. The j-th number in the i-th line represents the probability for the i-th student getting the j-th dish, taking the modulo of 998244353. There should be no extra spaces at the end of the line.

Examples

standard input	standard output

2
2 1
2 1
	665496236 332748118
	332748118 665496236

3
19 26 8
17 19 49
100 1 7
	772226764 583878773 640383170
	593962049 726647781 675878877
	630299894 685962153 680226660

Note

Suppose x and y are a pair of coprime integers, and y is not a multiple of 998244353. After taking the modulo of 998244353, $\frac{x}{y}$ becomes $x \times y^{998244351} \bmod 998244353$. Without dividing by zero, the result will be the same if you use operation after taking modulo.

Problem L. Euler Function

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

The Euler function, φ, is known to be important in math theory. Given a positive integer n, $\varphi(n)$ is defined as the number of integers in $[1, n]$ that are co-prime with n. For example, $\varphi(1) = 1, \varphi(10) = 4, \varphi(11) = 10$.

Now, given an array x with n positive integers $x_1, \ldots, x_n$, your task is to maintain m operations on x, which can be categorized into two types:

- $0\ l\ r\ w$, for each index $i \in [l, r]$, change x_i to $x_i \times w$.
- $1\ l\ r$, calculate and print $\left(\sum_{i=l}^{r} \varphi(x_i)\right) \mod 998244353$.

Input

The first line contains two inetegers n, m $(1 \le n, m \le 10^5)$.

The second line contains n integers $x_1, \ldots, x_n$ $(1 \le x_i \le 100)$.

Then m lines follow. The i-th line describes the i-th operation. The input guarantees that $1 \le l \le r \le n$ and $1 \le w \le 100$.

Output

For each operation of type 1, output a single line with a single integer, representing the answer.

Example

standard input	standard output

5 5
1 2 3 4 5
1 1 5
0 1 3 2
1 1 5
0 2 5 6
1 1 5
	10
	11
	37

Problem M. Addition

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

The numbers in the computer are usually stored in binary form. But in Little A's computer, the numbers are stored in a different way.

The computer uses n bits to store a number. The value it stores is $\sum_{i=0}^{n-1} v_i sgn_i 2^i$, where v is an array of length n containing only 0 and 1, and sgn is a predefined array of length n containing only -1 and 1. It is not difficult to find that every expressible integer has a unique expression.

Little A gives you the binary representation of a and b in his computer, and you should report the binary representation of $a + b$ in his computer. It is guaranteed that $max\{|a|, |b|\} \leq 10^8$, and all integers in $[-10^9, 10^9]$ can be expressed in his computer.

Input

The first line contains an integer n, which represents the number of bits used to store an integer.

The second line contains n integers, and the i-th of them represents sgn_{i-1}.

The third line contains n integers, and the i-th of them represents va_{i-1}. The value of a is $\sum_{i=0}^{n-1} va_i sgn_i 2^i$.

The fourth line contains n integers, and the i-th of them represents vb_{i-1}. The value of b is $\sum_{i=0}^{n-1} vb_i sgn_i 2^i$.

It is guaranteed that $32 \leq n \leq 60, sgn_i \in \{-1, 1\}, va_i, vb_i \in \{0, 1\}$, and $max\{|a|, |b|\} \leq 10^8$.

Output

Output one line containing n integers, separated by spaces. The i-th of them represents vc_{i-1}. There should be no extra spaces at the end of the line.

You should guarantee that $a + b = \sum_{i=0}^{n-1} vc_i sgn_i 2^i$.

Your output **must** be in a single line.

Example

standard input	standard output
`32`	`0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0`
`1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1`	`0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`
`1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1`	
`1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`	
`0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`	
`1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0`	
`0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0`	

Note

There is no extra line breaks in the test cases. It's just for the convience of displaying.The correct form can be found in the online statement.

Your output **must** be in a single line.