Problem A. Abstract Circular Cover

Input file:	*standard input*
Output file:	*standard output*
Time limit:	20 seconds
Memory limit:	512 mebibytes

The time limit is a bit strict.

There are n distinct points on a circle, numbered from 0 to $n-1$ inclusive in the clockwise order. A *circular segment* of length ℓ ($1 \le \ell \le n$) with start at i ($0 \le i \le n-1$) is a tuple of ℓ consecutive points in the clockwise order, starting with i (in other words, a tuple of points with numbers $i, (i+1) \bmod n, (i+2) \bmod n, \ldots, (i+\ell-1) \bmod n$). Circular segments of length n with starts at $0, 1, \ldots, n-1$ are considered to be pairwise different, despite containing the same set of points.

An integer cost $c_{i,\ell}$ is assigned to each circular segment. For each k from 1 to n, find the minimum total cost of exactly k circular segments, such that each of the n points is contained in exactly one of them.

Note that there are **no** properties that values $c_{i,\ell}$ satisfy, except being comparatively small positive integers. That is, any $n \times n$ array of integers between 1 and 10^6 is a valid test for this problem.

Input

The first line contains an integer n ($1 \le n \le 850$), the number of points on the circle. The $(i+1)$-st ($0 \le i \le n-1$) of the following n lines contains n space-separated integers $c_{i,1}, c_{i,2}, \ldots, c_{i,n}$ ($1 \le c_{i,\ell} \le 10^6$ for $1 \le \ell \le n$).

Output

Output n space-separated integers: k-th of them should be the minimum total cost of k circular segments that cover every point exactly once.

Examples

standard input	standard output

3
10 12 23
7 4 11
8 5 3
	3 12 25

1 15	15

Problem B. Biggest Set Ever

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

A set of nonnegative integers is *fine* if and only if all numbers in the set are less than T and their sum is equivalent to rem modulo n. Your task is to find the number of different fine sets.

Input

The first line of the input contains space-separated integers n and rem ($0 \le rem < n \le 10^4$). The second line of the input contains a single integer T ($1 \le T \le 10^{100\,000} - 1$).

Output

Print the number of different fine sets. As this number can be **really** large, you should print it modulo prime number 998 244 353.

Examples

standard input	standard output
3 2 5	8
1 0 20	1048576

Note

In the first sample, we can include or exclude numbers 0 and 3 freely, it doesn't change the remainder. From numbers $\{1, 2, 4\}$ there are two *fine* sets: $\{2\}$ and $\{1, 4\}$. So the answer is $2 \cdot 2 \cdot 2 = 8$.

In the second sample, any subset of $\{0, 1, \ldots, 19\}$ is fine, hence, the answer is $2^{20} = 1\,048\,576$.

Problem C. Convex Sets On Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A set of vertices of an undirected graph is *convex* if, for any two distinct vertices from the set, any simple path between them lies entirely within that set. Your task is to compute the number of convex subsets of vertices of a given graph.

Input

The first line of the input contains two integers n and m ($1 \le n \le 3 \cdot 10^5$; $n - 1 \le m \le 3 \cdot 10^5$): the number of vertices and the number of edges correspondingly. The i-th of the following m lines contains two space-separated integers x and y: the endpoints of the i-th edge ($1 \le x, y \le n$; $x \ne y$). It is guaranteed that the graph is connected and does not contain multiple edges.

Output

Print one number: the answer modulo prime number 998 244 353.

Examples

standard input	standard output

3 2
1 2
2 3
	7

3 3
1 2
2 3
3 1
	5

Problem D. Delete Two Vertices Again

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

Several years ago, the same author has proposed the following problem for a school competition: delete two adjacent vertices from a given connected undirected graph with min-degree at least 2 in a way that preserves connectivity. But that problem is too simple for a student competition in 2020, so here we go.

You are given a connected undirected graph with at least 3 vertices and without self-loops and multiple edges. For each edge, determine whether deleting both its endpoints preserves connectivity of the graph.

Notice that, in this problem, the graph **may** contain vertices of degree 1, but not vertices of degree 0 (since it is connected and has at least 3 vertices).

Input

The first line of the input contains two integers n and m ($3 \le n \le 3 \cdot 10^5$; $n - 1 \le m \le 3 \cdot 10^5$): the number of vertices and the number of edges correspondingly. The i-th of the following m lines contains two space-separated integers x and y: the endpoints of the i-th edge ($1 \le x, y \le n$; $x \ne y$). It is guaranteed that the graph is connected and does not contain multiple edges.

Output

Output a single binary string of length m: i-th character of the answer should be “0” (without quotes) if deleting endpoints of the i-th edge makes the graph disconnected and “1” otherwise.

Example

standard input	standard output

4 4
1 2
2 3
3 1
4 1
	0101

Note

Please notice that not only the edge itself is deleted, both its endpoints are too. Therefore, the task in question is different from finding bridges.

For example, in the sample, the edge 1–2 is not a bridge, but deleting vertices 1 and 2 makes the graph disconnected: the resulting graph consists of two isolated vertices 3 and 4.

On the other hand, the edge 1–4 is a bridge, but deleting vertices 1 and 4 makes the graph connected: the resulting graph consists of two vertices 2 and 3, connected by the edge 2–3.

Problem E. Easy One

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You have a sequence of digits 1 and 2. In one step you can:

1. Insert 1 in a place which is to the right of every other 1 (or anywhere if there are no 1s).
2. Transform any 2 into 1, if there are no 1s to the right of this 2.
3. Delete the rightmost 1 (note that this operation is inverse to the operation 1).
4. Transform the rightmost 1 into 2 (note that this operation is inverse to the operation 2).

For example, you can obtain the following sequences in one step from the sequence 11212122:

- With operation 1: $11212112 2, 112121212, 11212122\mathbf{1}$.
- With operation 2: $112121\mathbf{1}2, 1121212\mathbf{1}$.
- With operation 3: 1121222.
- With operation 4: $11212\mathbf{2}22$.

Your task is to calculate the number of ways to transform a sequence of exactly a digits 2 to a sequence of exactly b digits 2, using exactly t operations.

Input

The only line of the input contains three integers a, b, and t ($0 \le a, b, t \le 10^6$).

Output

Output the number of ways to obtain a sequence of b digits 2 from a sequence of a digits 2 in exactly t steps. As this number can be very large, output it modulo prime number 998 244 353.

Examples

standard input	standard output
0 0 4	3
1 4 6	60

Note

In the first sample you should obtain an empty sequence from an empty sequence in 4 steps. Ways to do this are (ε stands for empty sequence):

$$\varepsilon \to 1 \to \varepsilon \to 1 \to \varepsilon$$
$$\varepsilon \to 1 \to 11 \to 1 \to \varepsilon$$
$$\varepsilon \to 1 \to 2 \to 1 \to \varepsilon$$

Problem F. Football

Input file:	*standard input*
Output file:	*standard output*
Time limit:	0.5 seconds
Memory limit:	512 mebibytes

You are standing at position $(0, 0)$ on the plane and want to pass the ball to your teammate standing at (x_t, y_t). Unfortunately, there is a player from the opposing team in position (x_o, y_o) who wants to intercept this pass.

A player gets the ball at a certain moment if their positions on the plane coincide at that moment. Both players can move with speed s_p, and you can make the ball fly with constant speed s_b. Note that, at some moment, the ball will touch the ground and the friction will stop it, so there is no case when players can't catch the ball at all, but for simplicity we assume that it will happen far far away.

Your task is to determine whether you can make a pass to your teammate, that is, throw the ball so that your teammate gets to it faster than the opponent. Output "+" (without quotes) if you can make such a pass that your teammate will be the first to get the ball, "0" if you can't, but you can make such a pass that both players will get the ball simultaneously, and "-" otherwise.

Input

The first line contains an integer n ($n \le 10^5$): the number of test cases. Each of the following n lines describes a separate test case and contains six integers $x_t, y_t, x_o, y_o, s_b, s_p$ ($-75 \le x_t, y_t, x_o, y_o \le 75$; $1 \le s_b, s_p \le 75$): coordinates of the teammate, coordinates of the opponent, and speeds of the ball and players, respectively. Your position, the teammate's position, and the opponent's position are pairwise different.

Output

Output n answers on a single line without spaces.

Example

standard input	standard output

3
1 0 2 0 1 1
0 3 0 -1 1 2
75 0 2 0 1 10
	+0-

Problem G. Game On Board

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Little Semyon plays with a board. Initially, he has a big board with n rows and m columns, with some cells painted black. Then, he starts to paint other cells black using the following rule: if three corners of some rectangle are already black, he can paint the remaining corner black as well. However, not every initial board satisfies Semyon: he wants to be able to paint the whole board black, and the initial board should contain the least possible number of black cells. Your task is to find the number of initial boards with given size which can satisfy Semyon.

Input

The only input line contains two integers n and m ($1 \le n, m \le 10^9$), the numbers of rows and columns respectively.

Output

You should print the number of boards which can satisfy Semyon. As this number can be very large, print it modulo prime number 998 244 353.

Examples

standard input	standard output
1 1	1
2 2	4

Note

In the second sample, the satisfying boards are exactly the boards with 3 black cells.

Problem H. Hardcore String Counting 2

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

What happened to "Hardcore String Counting 1"? That's a secret!

A non-empty word v over some alphabet is a *square* if it can be represented as $v = ww$ for some word w.

A word is *square-free* if all its non-empty substrings are not squares.

Your task is to compute the number of square-free words of length ℓ over the alphabet $\{\texttt{a}, \texttt{b}, \texttt{c}\}$ for each ℓ from 1 to n.

Input

The only line of the input contains an integer n ($1 \le n \le 120$).

Output

For each ℓ from 1 to n, print the number of square-free words of length ℓ over the alphabet $\{\texttt{a}, \texttt{b}, \texttt{c}\}$ on a separate line.

Example

standard input	standard output
10	3 6 12 18 30 42 60 78 108 144

Note

You can see `https://oeis.org/A006156` for more details, but this probably won't help you much.

Problem I. Inv

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A permutation p on n elements is an *involution* if $p(p(i)) = i$ for each i from 1 to n inclusive. Your task is to compute the number of involutions on n elements with k inversions. To make your life easier, we ask you to print only the parity of this number.

Input

In the only line of the input, two space-separated integers are given: n ($1 \le n \le 500$), the length of the involution, and k ($0 \le k \le \frac{n(n-1)}{2}$), the number of inversions.

Output

Print a single number (0 or 1): the number of involutions on n elements with exactly k inversions, when taken modulo 2.

Examples

standard input	standard output
`4 1`	`1`
`10 21`	`0`

Note

In the first sample, there are 3 such involutions.

Problem J. Justice For Everyone

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

Suppose you have n pairwise different numbers on a desk, denoted by $a_1, a_2, \ldots, a_n$ (order matters). In one turn, you can choose two different indices $i_1 < i_2$ and simultaneously increase a_{i_1} and a_{i_2} by one. The only condition is that the numbers on the desk should be different in every moment. Your task is to find the number of ways to obtain pairwise different numbers $b_1, b_2, \ldots, b_n$ (in exactly this order). As this number can be very large, print it modulo 998 244 353.

Input

The first line of the input contains a single integer n ($1 \le n \le 30$). The second and the third lines of the input contain n space-separated integers each: the arrays a_i and b_i respectively ($1 \le a_i, b_i \le 200$). All a_i are guaranteed to be pairwise different, same for b_i.

Output

Print the answer modulo prime number 998 244 353.

Examples

standard input	standard output

3
1 2 3
3 4 5
	1

3
1 2 3
7 8 9
	42

3
1 4 7
3 6 9
	6

Note

In the first sample, the only way is to make operations in the order $\{2,3\}, \{1,3\}, \{1,2\}$.

In the third sample, we can make the three operations $\{1,2\}, \{2,3\}, \{1,3\}$ in any order.

Problem K. Keep It Cool

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Consider the set of all pairs (a, b) with a and b satisfying the constraint $1 \le a < b \le n$. A permutations of all these pairs is called *balanced* if and only if, for any a, b and c satisfying $1 \le a < b < c \le n$, the pair (a, c) is between the pairs (a, b) and (b, c) in the permutation. That is, those three pairs appear in the permutation either in the order $(a, b), (a, c), (b, c)$, or in the order $(b, c), (a, c), (a, b)$.

Moreover, there are m additional restrictions on the permutation, i-th of them states that the pair (a_i, b_i) should appear before the pair (c_i, d_i) in the permutation. Compute the number of balanced permutations that satisfy these m restrictions.

Input

In the first line of the input, there are space-separated integers n and m ($2 \le n \le 10$; $0 \le m \le 10$). The i-th of the following m lines contains space-separated integers a_i, b_i, c_i, and d_i ($1 \le a_i < b_i \le n$; $1 \le c_i < d_i \le n$; pairs (a_i, b_i) and (c_i, d_i) are different).

Output

Print the number of balanced permutations where (a_i, b_i) appears before (c_i, d_i) for every i. As this number can be very large, print it modulo prime number 998 244 353.

Examples

standard input	standard output
2 0	1

4 3
1 2 2 3
1 2 3 4
2 3 3 4
	2

Note

In the second sample, the permutations are:
$(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)$;
$(1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4)$.

Problem L. Lower Algorithmics

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are given a set A of integers from 1 to 1000 inclusive. Your task is to find the number of positive integers that can be represented as a sum of several elements of A, with number of summands being from ℓ to r inclusive. **Equal summands are allowed**. Note that each number is counted only once, even if it has several such representations.

Input

The first line of the input contains three space-separated integers: n, the number of elements in A ($1 \le n \le 1000$), followed by ℓ and r, the bounds on number of summands ($1 \le \ell \le r \le 2000$). The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_1 < a_2 < \ldots < a_n \le 1000$): the elements of the set A in increasing order.

Output

Output the number of integers that are representable as a sum of elements of A, with number of summands being between ℓ and r.

Examples

standard input	standard output
2 1 2 1 2	4
3 1 3 1 3 10	18
6 3 5 1 2 3 4 5 6	28