Problem A. Algorithm Was Applied

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

Consider a connected undirected graph on n vertices. Denote this graph as the **starting graph**.

A tuple of integers (a, b, c) $(1 \le a < b < c \le n)$ is **good** if and only if

1. Vertices a and b are connected by an edge.
2. Vertices a and c are connected by an edge.
3. Vertices b and c are not connected by an edge.

The following algorithm was applied to the starting graph. While there is one, choose an arbitrary good tuple (a, b, c) and add an edge between vertices b and c. Denote the resulting graph as the **completed graph**. It can be proven that the completed graph does not depend on the choices of good tuple at each iteration.

You are given a starting graph. How many colorings in n colors does the completed graph have? Graph coloring is an assignment of colors to vertices such that no two adjacent vertices share a color. Refer to problem D if you need an even more formal definition.

Output the correct answer modulo 998244353. Formally, if the real answer is y and your answer is x, it will be considered correct if $-2^{63} \le x < 2^{63}$ and $x - y$ is divisible by 998244353.

Input

The first line of input contains two integers n and m $(2 \le n \le 3 \cdot 10^5, 1 \le m \le 3 \cdot 10^5)$, the number of vertices and edges in the starting graph, respectively.

Next m lines contain descriptions of starting graph edges. The i-th of them contains integers u and v $(1 \le u < v \le n)$, the endpoints of the i-th edge.

It is guaranteed that the given graph is connected and doesn't contain multiple edges.

Output

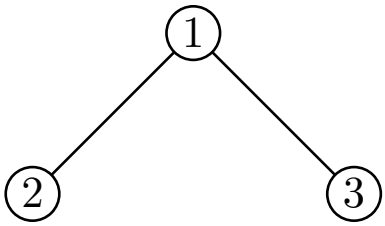
Print a single integer, the number of colorings in n colors of the completed graph modulo 998244353.

Examples

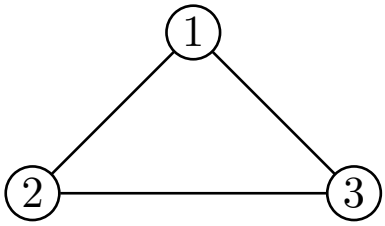
standard input	standard output
3 2 1 2 1 3	6
8 9 1 2 3 8 1 3 2 6 4 7 5 6 2 5 2 4 7 8	645120

Note

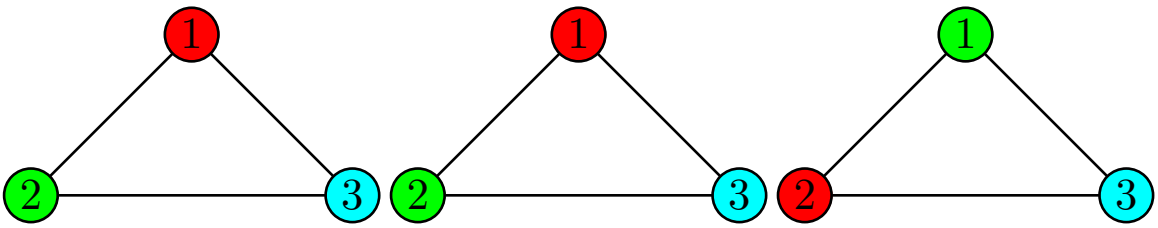
In the first example the starting graph looks like this:

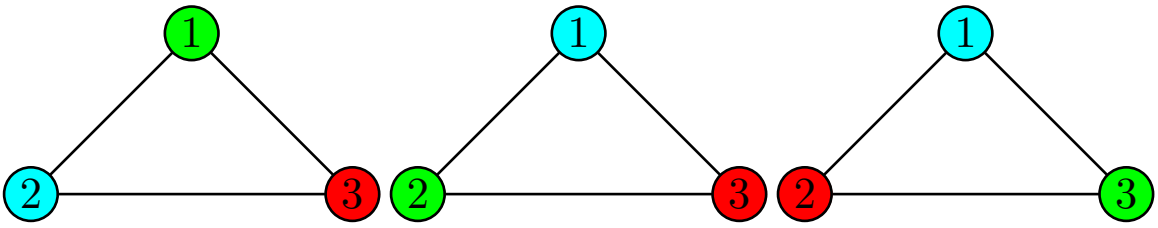


The only possible good tuple is $(1, 2, 3)$, so completed graph looks like this:



Here are all 6 colorings of such a graph in 3 colors:





Problem B. Balanced Rainbow Sequence

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

Lewin and Gan have a ~~parentheses~~ sequence. Nah, that word is clumsy as hell. Okay, one more time:

You are given a bracket sequence. Each bracket has one of 3 colors: lime, gray and blue and is either opening or closing.

You are allowed to change some brackets from opening to closing and vice versa. You are not allowed to change the order or colors of brackets. You want to do zero or more changes such that the resulting sequence satisfies the following:

1. If all the lime brackets are removed, the remaining brackets form a balanced bracket sequence.
2. If all the gray brackets are removed, the remaining brackets form a balanced bracket sequence.

A bracket sequence S is balanced if it satisfies one of the following conditions:

1. S is empty.
2. $S = XY$, where both X and Y are non-empty balanced bracket sequences.
3. $S = (X)$, where X is a balanced bracket sequence. Note that the starting opening brace and the trailing closing brace may have different colors.

Is it possible to do so and if it is, what is the minimum number of changes you have to make?

Input

The first line contains a single integer n ($1 \le n \le 6000$), the length of the bracket sequence.

The second line contains a string s of length n consisting of characters "(" and ")", the bracket sequence.

The third line contains n integers c_i ($0 \le c_i \le 2$). c_i denotes the color of the i-th bracket. 0 corresponds to lime, 1 to gray and 2 to blue.

Output

Print a single integer — the minimum number of changes you have to make if it is possible to satisfy the conditions and `-1` otherwise.

Examples

standard input	colored input	standard output
5))))(0 1 2 2 2	5))))(0 1 2 2 2	4
3 ()(0 2 1	3 ()(0 2 1	-1
6 (()()) 0 0 0 0 0 0	6 (()()) 0 0 0 0 0 0	0

Note

In the first example a possible resulting sequence is `(()()`.

Problem C. Called Convergient

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

You are a bettor. You have a given amount of money. You make bets of non-decreasing sizes, until you lose all your money or reach a specified threshold. You win in the former case and lose in the latter.

Formally, initially you have x units of money and you have a counter y, which is equal to your last bet. It is initially equal to 0. You repeatedly do the following:

1. If $x \geq 1$ you win.
2. If $y > x$ *you get nothing.* You lose. *Good day, sir.*
3. You set y to a new positive real value not greater than x, which must not be less than the previous one.
4. With a given probability p x increases by y and with probability $(1 - p)$ x decreases by y.

What is the probability of winning if you bet optimally?

For those who like calculus, the probability of winning from a given position is the supremum of the set of probabilities of winning of all possible strategies.

Output the answer modulo 998244353. Formally, the actual answer is guaranteed to be an irreducible fraction $\frac{P}{Q}$ where Q is co-prime to 998244353. Output an integer X, such that $-2^{63} \leq X < 2^{63}$ and $XQ - P$ is divisible by 998244353.

Tests are generated randomly more or less, therefore if you do all calculations modulo 998244353 you won't encounter division of 0 by 0, unless you multiply both sides by a number divisible by 998244353 just for the sake of it.

Input

Both x and p are given as irreducible fractions.

The only line of input contains four integers a, b, d and e ($1 \leq a, b, d, e \leq 10^6$).

x is equal to $\frac{a}{b}$. $0 < x < 1$, i.e. $a < b$. a and b are co-prime.

p is equal to $\frac{d}{e}$. $0 < p < \frac{1}{2}$, i.e. $d < \lceil \frac{e}{2} \rceil$. d and e are co-prime.

Output

Output a single integer — the answer to the problem modulo 998244353.

Examples

standard input	standard output
1 2 1 3	332748118
17 28 3 29	60518375

Note

In the first example the best strategy is to simply bet all your $\frac{1}{2}$ money and win or lose instantly. The probability of winning is p which is equal to $\frac{1}{3}$.

Appendix

In this problem you can easily end up in a situation where you know a number A, such that for each real number c except A, you have an argument that c makes no sense as the answer to the problem. In that case A is indeed the answer as the answer does always exist even if you don't have the strategy with probability of winning exactly A (actually there may be no such strategy, example of a game where it is easier to understand is present later in the statement). However, for your convenience some definitions which may help you develop a better understanding of the problem are provided.

Limit of a sequence f_n is an integer F, such that for all real $\varepsilon > 0$ there exists an integer N, such that for every $n > N, |f_n - F| < \varepsilon$. To put it more understandable way, F is limit if elements of the sequence with a big enough indices become arbitrarily close to F. It can be proven that if the limit exists it is unique. If it does sequence f_n is said to **converge** to F and is called **convergent**. It can be shown that a sequence is convergent if and only if for all real $\varepsilon > 0$ there exists an integer N, such that for every $n_1, n_2 > N, |f_{n_1} - f_{n_2}| < \varepsilon$. In other words, a sequence converges if elements with big enough indices become arbitrarily close to each other. Examples:

1. $f_n = \frac{n}{n+1}$. 1 is the limit of this sequence. Therefore this sequence converges to 1.
2. $f_n = n$. This sequence has no limit.
3. $f_n = 5$. 5 is the limit.

Supremum, sometimes called the least upper bound, of a possibly infinite set of real numbers is the least real number greater or equal than all elements of the set. It can be proven that any non-empty set bounded from above has a supremum. Obviously the supremum is unique because there can not be multiple least numbers satisfying some property. One of them is always greater than another and is therefore not least. Examples:

1. Supremum of the set $\{2, 5, 3, 100\}$ is 100.
2. Supremum of the set of real numbers x: $0 < x < 2$ is 2. Note that 2 itself is not present in the set.
3. Supremum of the set of all negative real numbers is 0.
4. The set of all positive real numbers has no supremum because it is unbounded from above.

Strategy is a definition of how you will play a game and which actions will you make under all possible circumstances. Examples:

1. Consider the following game: You roll a six sided die. After than you may choose to reroll it. You win if the value on top of the die is 6. The strategy for the game can be represented as 6 boolean variables $x_1, x_2, \ldots, x_6$. x_i describes whether you will reroll if you get i on your first roll. The vast majority of them are blatantly unoptimal but they are valid strategies nonetheless.
2. Consider the following game: You choose a positive real number x, such that $0 < x < 1$. After that another real number y is chosen from the range $[0, 1)$. You win if $x > y$. A strategy in this game can be represented by a single real number — the value of x is choose.
3. A strategy in the game this problem is about can be represented a possibly infinite binary decision tree. Each vertex contains a real number — the bet you make if you are standing at it. After than you move to the left child of the decision tree if your bet won and to the right if it lost. Some branches of the decision tree may be cut off if the game has ended by then.

Probability of winning for a particular strategy in a single player game is, *surprisingly,* the probability of that strategy leading to a win for the player. Examples:

1. In the die game (see above) the strategy "always reroll" yields the probability of winning of $\frac{1}{6}$.
2. In the "Choose x to win game" (see above) a strategy has the probability of winning of exactly x (the number it dictates you to choose).
3. In the game this problem is about the probability of winning for a particular strategy can be hard to determine however it is always defined. Consider some strategy and let w_n, d_n, l_n denote the probabilities that after n turns the game is won, is ongoing and is lost respectively. It is obvious that $w_n + d_n + l_n = 1$, w_n and l_n

are non-decreasing and d_n is non-increasing. Furthermore, d_n converges to 0 (A sketch of the proof: let the first bet of the strategy be a. This means that all the following bets are at least a. Let k denote $\lceil \frac{1}{a} \rceil$. If you win k bets in a row the game ends as a win for you if it has not ended before. Therefore every k turns there is an at least p^k chance that the game will end and at most $1 - p^k$ chance that the game will continue. A geometric progression with ratio less than 1 is known to converge to 0). As $w_n + d_n$ is non-increasing and w_n is non-decreasing for all positive N and $n_1, n_2 \geq N$, $w_{n_1}, w_{n_2} \in [w_N, w_N + d_N] \to |w_{n_1} - w_{n_2}| \leq d_N$. As d_N converges to 0 this means that w_n is convergent. The limit of w is the probability of winning.

Probability of winning a game is the supremum of the set of probabilities of winning of all possible strategies. Examples:

1. The probability of winning the die game (see above) is $\frac{11}{36}$ if you reroll everything but 6.
2. The probability of winning the "Choose x to win game" (see above) is 1. Note that there isn't a strategy with the probability of winning of 1. However there are strategies with the probability of winning arbitrarily close to 1.
3. *The probability of winning the game this problem is about is described in this video https://youtu.be/BcxeZ4Wwdn0 (an unofficial cover of Barbara Streisand - Duck Sauce)*

Problem D. Doesn't Contain Loops or Multiple Edges

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

Coloring of a labeled undirected graph with n vertices in k colors is an assignment of colors to its vertices, such that each vertex receives an integer color x ($1 \le x \le k$) and no two adjacent vertices have the same color. A coloring can be treated as an array of integers from 1 to k of length n, i-th element of which corresponds to the color of the i-th vertex of the graph.

Coloring b is **monotonic** to coloring a of the same graph if exactly one of the following statements holds:

1. $\forall_{1 \le i \le n}\ a_i \le b_i$
2. $\forall_{1 \le i \le n}\ a_i \ge b_i$

Note that a coloring is not monotonic to itself because in that case both statements above hold.

You are given a labeled undirected graph and its coloring a in k colors. Is there a coloring b of the given graph in k colors which is monotonic to a?

Input

The first line contains three integers n, m and k ($1 \le n, m, k \le 3 \cdot 10^5$), the number of vertices in the graph, the number of the edges in the graph and the number of colors, respectively.

The second line contains n integers a_i ($1 \le c_i \le k$), the colors of vertices.

m lines follow. i-th of them contains two integers u_i and v_i ($1 \le u_i < v_i \le n$), describing an edge between vertices u_i and v_i.

The graph doesn't contain loops or multiple edges. It is guaranteed that the array a describes a valid coloring of the given graph.

Output

Print `1` if there exists a coloring b monotonic to a and print `0` otherwise.

Examples

standard input	standard output
3 2 3 1 2 3 1 2 2 3	1
3 3 3 1 2 3 1 2 2 3 1 3	0
6 6 3 1 2 3 1 2 3 4 5 1 2 2 3 3 4 1 6 5 6	0

Problem E. Equal Adjacent Elements

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

Hello sir, I am very interested in coding.. But I am not able to get the solution to a problem. Can you help me??

An array of integers is **bad** if it contains a pair of equal adjacent elements. An array is **good** if it is not bad.

You are given a good array. How many ways are there to remove its elements one by one, concatenating the left and right part after each removal, such that at no point the array is bad? Two ways are different if there is a step on which indices of removed elements differ.

For example, after the removal of 1 from a good array $[2, 1, 2]$ it becomes a bad array $[2, 2]$ and after the removal of 5 from a good array $[1, 2, 3, 4, 5, 6]$ it becomes $[1, 2, 3, 4, 6]$ and stays good.

Output the answer congruent to the real one modulo 998244353. Formally, if the real answer is y and your answer is x, it will be considered correct if $-2^{63} \le x < 2^{63}$ and $x - y$ is divisible by 998244353.

Input

The first line contains a single integer n ($1 \le n \le 500$), the number of elements in the array.

The second line contains n integers a_i ($1 \le a_i \le n$), elements of the array.

Output

Output a single integer — the answer to the problem modulo 998244353.

Examples

standard input	standard output
3 1 2 3	6
4 1 2 1 2	8
12 1 2 3 1 2 3 1 2 3 1 2 3	25660800
1 1	-998244352
2 1 2	998244355

Note

In the first example all elements are distinct, so it's impossible to get a bad array at some point. That's why there are $3! = 6$ ways to remove all the elements.

In the fourth example the real answer is 1 because there is only one way to remove the only element. The given answer -998244352 is congruent to 1 modulo 998244353, so it is correct too.

In the fifth example the real answer is 2.

Sorry for my bad English.

Problem F. Formally, You Choose Three Integers

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

You are given two arrays a and b of the same length n.

You are allowed to perform zero or more operations on a of the following kind:

Choose a contiguous subarray of even length and cyclically shift it by an even number of positions. Formally, you choose three integers i, j and k ($0 \le i < j < k \le n$, $j - i$ is even, $k - i$ is even) and a becomes equal to $a_{0:i} + a_{j:k} + a_{i:j} + a_{k:n}$, where $a_{l:r}$ denotes a slice with Python indexing. Precisely, it contains elements in the range of indices $[l, r)$ in 0-indexing and $(l, r]$ in 1-indexing.

Is it possible to transform a into b?

Input

The first line contains a single integer n ($1 \le n \le 3 \cdot 10^5$), the length of a and b.

The second line contains n integers a_i ($1 \le a_i \le n$), elements of a.

The third line contains n integers b_i ($1 \le b_i \le n$), elements of b.

Output

Print `1` if it is possible to transform s into t and `0` otherwise.

Examples

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

Problem G. Game

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

There are two arrays of integers a and b of length n and m, respectively.

There is a chosen position in each array. Denote the chosen position in a as c and in b as d. The **state** is a tuple (c, d). The chosen positions are never out of bounds, i.e. $1 \le c \le n$ and $1 \le d \le m$. At the start of the game both c and d are equal to 1.

Two players are playing a game taking turns alternately. On his turn each player must do one of the following:

1. Move. Change the chosen position in one of the arrays. You can't change a chosen position to itself. It is not allowed to make a move which will result in a state appearing for the 10^{228}-th time. The starting state is considered as having appeared once at the beginning of the game.
2. *Screw you guys, I'm going home.* Finish the game.

Note that there are $n \cdot m$ states and each state appears no more than $10^{228} - 1$ times. Therefore the game is finite.

The score of the game is equal to the sum of elements on the chosen positions, i.e. $a_c + b_d$. The player who moves first minimizes it by playing accordingly and the one who moves second maximizes it.

What is the score of the game assuming perfect play from both sides?

Input

The first line contains two n and m ($1 \le n, m \le 10^5$), lengths of arrays a and b, respectively.

The second line contains n integers a_i ($1 \le a_i \le 10^8$), the elements of a.

The third line contains m integers b_i ($1 \le b_i \le 10^8$), the elements of b.

Output

Output a single integer — the score of the game.

Examples

standard input	standard output
2 2 1 3 1 3	2
2 1 3 2 2	4
3 1 3 3 2 2	5
4 5 10 1 2 3 10 1 2 3 4	11
6 7 4 5 6 1 2 3 5 2 1 3 4 6 7	7
4 3 64 16 4 1 32 8 2	33

Problem H. Hat With An Integer

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

The are n enumerated wise ones. Each of them has a hat with an integer written on it. Each wise one sees numbers on other wise ones' hats, but does not see the number on his own. Let a_i denote the number written on the hat of the i-th wise one.

There are also two arrays of integers of length $n-1$ each: b and c. We call the assignment of integers to hats **valid** if there exists at least one value of i, such that $1 \le i \le n-1$ and at least one the following holds:

1. $a_{i+1} < a_i + b_i$
2. $a_{i+1} > a_i . + c_i$

The wise ones know that the actual assignment (i.e. the array a) is valid.

The following process happens: Each day, starting from the first, if at the beginning of the day there exists a wise one with index i who knows an integer x, such that $a_i \neq x$ (i.e. he knows a number which is definitely not written on his hat) he announces this at the end of the day and the process ends. If no such wise one is present the process continues on the next day and so on

Calculate whether the process will end and if it will, calculate the day on which it will happen.

Input

The first line contains a single integer n ($2 \le n \le 10^5$), the number of wise ones.

The second line contains n integers a_i ($-10^{10} \le a_i \le 10^{10}$).

The third line contains $n-1$ integers b_i ($-10^{10} \le b_i \le 10^{10}$).

The fourth line contains $n-1$ integers c_i ($-10^{10} \le c_i \le 10^{10}$).

It is guaranteed, that the array a is valid.

Output

Output `-1` if the process will never end. Output the day on which the process will end otherwise.

Examples

standard input	standard output
3 1 2 3 0 0 0 0	2
2 -1 -1 1 -1	-1
7 1 4 2 -5 10000000000 6 -7 -1 2 -3 4 6 5 2 3 322 5 100 312	5
8 3 2 8 25 16 24 24 24 -1 0 3 4 -2 7 6 -1 2 7 5 -1 12 11	4

Note

Consider the first example. The constraints given by arrays b and c can be rephrased as "not all integers are equal".

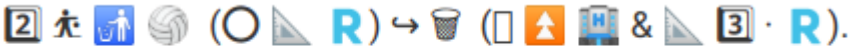
On the first day no wise one can deduce a number which is not written on his hat.

On the second day the third wise one knows, that if the number on his hat was 2, on the first day the first wise one would have seen that both other wise ones have 2 on their hats and would have deduced that he does not wear a hat with 2 written on it. Thus on the second day the third wise one knows that he does not wear a hat with 2 written on it.

Is this readable? Probably not. Could this have been phrased much better to be readable? Probably not.

Problem I. Intersect With Other Balls

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



And now for something completely different - an English statement.

Two players are taking turns throwing balls (circles of radius r) into a rubbish bin (a rectangle of height h and of width $3 \cdot r$).

When the player throws a ball, he initially places the ball strictly inside the bin in such way, that it touches the upper boundary of the bin and does not intersect with other balls. It may touch other balls though (not like it matters with integer inputs). Then the ball moves strictly down until it touches the bottom of the bin or another ball. After that it stops and doesn't move for the rest of game.

The player who can not make a turn (i.e. there is not enough space to initially place the ball) loses.

Who will win, assuming perfect play?

Input

The only line contains two integers r and h ($1 \le r, h \le 10^8$, $2 \cdot r < h$), the radius of the balls (also one third of the width of the bin) and the height of the bin respectively.

Output

Print 1 if the player who goes first wins and 2 otherwise.

Examples

standard input	standard output
1 4	2
2 28	1
3 22	1
14 88	1
23 5100	2

Problem J. J The Attacker Has

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

This problem is about the Braindead card game which is invented on 21 February 2019 and is loosely based on some other unspecified card game. It is fairly easy to guess though.

There are cards of n suits and m ranks. Some suits are trumps. Note that there may be multiple trumps or none at all. Unlike most actual card games multiple cards may share both rank and suit.

There are two players. Each of them has some non-empty set of cards. This set of cards is called the hand. Both players know each other's hands. The cards in the hand of some player are called his cards.

A card of suit s_1 with rank r_1 beats a card of suit s_2 and rank r_2 if one of the following conditions hold:

1. $s_1 = s_2$ and $r_1 > r_2$
2. s_1 is a trump and s_2 is not a trump.

One of the player is the attacker and the other is the defender. The following steps happen

1. The attacker starts the round by playing one of his cards. This is called the **starting attack**.
2. The defender plays one of his cards, which beats the last card the attacker played. If the defender has no such card he loses the game.
3. The attacker plays one of his cards, such that its rank is equal to the rank of some card played before by either player. If the attacker has no such card he loses the game.
4. Go to Step 2.

A player can not play the same card twice.

How many starting attacks are there, such that the attacker wins the game from there assuming players play optimally afterwards (after the starting attack)? Starting attacks which play different cards with the same rank and suit are considered different starting attacks.

Input

The first line contains two integers n and m ($1 \le n, m \le 18$), the number of suits and ranks, respectively.

The second line contains n integers t_i ($0 \le t_i \le 1$). t_i is equal to 1 if the i-th suit is a trump and to 0 if it is a regular suit (not a trump).

Next n lines describe the hand of the attacker. i-th of them contains m integers $a_{i,j}$ ($0 \le a_{i,j} \le 10^{12}$). $a_{i,j}$ is equal to the number of cards of suit i with rank j the attacker has.

Next n lines describe the hand of the defender in the same format.

It is guaranteed that both players hands are non-empty.

Output

Output a single integer — the number of winning first attacks.

Examples

standard input	standard output
3 2 0 0 1 1 1 1 0 1 0 0 1 0 1 1 1	0
3 1 0 1 0 1 0 1 0 1 0	2
2 4 1 1 5 5 5 0 0 0 0 0 5 5 5 0 0 0 0 10	15
4 13 1 0 1 0 3 2 0 2 0 3 0 0 3 5 2 1 5 3 3 2 2 2 1 0 4 5 1 5 3 3 1 4 4 2 0 3 2 5 2 5 0 5 3 3 5 4 2 3 3 4 2 2 4 3 2 3 4 3 4 9 0 4 0 1 4 0 1 2 5 1 8 8 6 5 1 1 7 2 4 1 3 9 3 7 3 1 8 9 2 0 0 1 9 6 6 4 6 6 7 9 5 3 2 9 5 0 7 8	97
1 2 0 4294967296 0 0 2147483647	4294967296

Problem K. K Integers

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

You are given a string s of length n.

A sequence of integers t is an **index sequence** if $1 \le t_1 < t_2 < \ldots < t_k \le n$, where k is the length of t.

A string corresponding to an index sequence t is the the following string: $s_{t_1} s_{t_2} \ldots s_{t_k}$. Note that it is always a subsequence of s.

You are given an index sequence. Find the lexicographically smallest string which corresponds to some index sequence which contains the given one as a subsequence.

Input

The first line contains the string s consisting of n ($1 \le n \le 5 \cdot 10^5$) lowercase English letters.

The second line contains a single integer k ($1 \le k \le n$), length of t.

The third line contains k integers t_i ($1 \le t_i \le n$). t is an index sequence.

Output

Print a single string — the answer to the problem.

Examples

standard input	standard output
links 2 3 4	ink
abacaba 2 4 6	aacab
pepega 2 2 6	ea
gaypride 2 6 7	aid
pogchamp 3 1 2 3	pog
frankerz 1 8	aerz
blessrng 8 1 2 3 4 5 6 7 8	blessrng
residentsleeper 4 1 3 7 15	resdeneeer

Problem L. Labeled Connected Graphs

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

You are given an integer n and a prime modulo m.

Calculate the sum of distances between the first and the second vertices over all distinct labeled connected graphs with n vertices.

Output any integer congruent to the actual sum modulo m. Formally, if the actual sum is S output any integer x such that $-2^{63} \le x < 2^{63}$ and $x - S$ is divisible by m.

Input

The only line contains two integers n and m ($2 \le n \le 400$, $10^6 + 3 \le m \le 10^9 + 9$, m is prime), the number of vertices in the graphs and the modulo.

Output

Print a single integer — the answer to the problem.

Examples

standard input	standard output
2 998244353	1
3 1001177	5
4 1000003	54
5 1000159	1108
6 1000253	41880
7 100000007	2946440
10 100001303	82834735
25 100002013	77432340
31 100001887	7237626
42 100002593	44678783
68 31235417	22825855
117 12345847	4325099
200 1000000007	194453485
228 12348143	6438982
300 34569361	24221941
312 34570903	12146306
322 41236777	26590080
350 41237641	40908795
366 66666667	57608403
378 99000007	61227322
399 99990001	46973248
400 1000000009	478599227

Note

If you manage to get WA in this problem and we reasonably believe that you did not intentionally try to do so, we might give you a cookie somehow.

Problem M. Moves You Need to Make

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

You are given a permutation.

A move is one of the following:

1. Swap two adjacent elements.
2. Swap the first and the last elements. Can be used at most once.

What is the minimum number of moves you need to make to sort the given permutation?

Input

The first line contains a single integer n ($1 \le n \le 3 \cdot 10^5$), the length of the permutation.

The second line contains n integers a_i ($1 \le a_i \le n$), the permutation itself.

Output

Output a single integer — the minimum number of moves you need to make to sort the given permutation.

Examples

standard input	standard output
1 1	0
2 1 2	0
3 3 2 1	1
4 4 2 1 3	2
5 4 1 5 3 2	4
6 1 5 3 4 2 6	5
7 3 2 1 7 6 5 4	9
8 4 2 6 1 5 3 7 8	8
9 9 8 7 6 5 4 3 2 1	22
10 8 2 9 5 1 7 10 4 6 3	17
11 7 2 3 9 11 1 8 6 4 10 5	23
12 3 10 6 2 4 12 7 8 5 1 11 9	18

Problem N. Number Of Vertices

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

New meta.

A **zigzag cycle** in an undirected graph is a sequence of vertices $a_0, a_1, \ldots, a_{k-1}$, not necessarily distinct, such that for all $i : 0 \le i < k$ a_i and $a_{(i+1) \mod k}$ are adjacent in the graph and one of the following holds:

1. $a_{(i+k-1) \mod k} < a_i, a_i > a_{(i+1) \mod k}$
2. $a_{(i+k-1) \mod k} > a_i, a_i < a_{(i+1) \mod k}$

A cycle contains the edge (u, v) p times if there exist exactly p distinct $i : 0 \le i < k$, such that $a_i = u$, $a_{(i+1) \mod k} = v$ or $a_i = v$, $a_{(i+1) \mod k} = u$.

A graph is **splittable** if there exists a set of zigzag cycles, such that for each edge exactly one cycle contains it 1 time and all the remaining cycles contain it 0 times, i.e. you can split edges of the graph into zigzag cycles.

There is a graph which is initially empty. Process the following types of queries:

1. Add an edge between vertices u and v.
2. Remove an edge between vertices u and v.

After each query print whether the graph is splittable.

Input

The first line contains two integers n and q ($2 \le n \le 3 \cdot 10^5, 1 \le q \le 3 \cdot 10^5$) — the number of vertices in the graph and the number of queries, respectively.

q lines follow. i-th of them contains three integers t, u, v ($t \in \{1, 2\}, 1 \le u < v \le n$) — the type of query and the endpoints of the edge you have to add if $t = 1$ or remove if $t = 2$. No query will ask to you to add an already present edge or to delete an absent one.

Output

Print q lines. i-th of them should contain `1` if the graph is splittable after the first i queries and `0` otherwise.

Example

standard input	standard output
6 10 1 1 4 1 1 5 1 4 5 1 3 5 1 3 4 2 4 5 1 4 5 1 2 5 1 2 6 1 4 6	0 0 0 0 0 1 0 0 0 1

Note

After processing all the queries one possible set of zigzag cycles is $\{[1, 4, 3, 5], [2, 6, 4, 5]\}$.