Problem A. Alice and Bob

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

Alice and Bob will play a game by alternating turns with Alice going first.

They have a directed acyclic graph, such that each edge $u \to v$ satisfies $u < v$.

All vertices in this DAG are colored one of two colors, and vertices have tokens on them (a vertex may contain more than one token).

During her move, Alice will choose a white vertex u which contains at least one token. Then, she will choose some outgoing edge, $u \to v$, and move one token from vertex u to the vertex v.

During his move, Bob will choose a black vertex u which contains at least one token. Then, he will choose some outgoing edge $u \to v$ and move one token from vertex u to the vertex v.

The person who can't move loses.

Alice and Bob haven't decided on the configuration of tokens yet, but they have decided that each vertex at the beginning of the game will contain at most one token. Among all 2^n placement of tokens, calculate how many of them Alice will win under the optimal play of both players? As this value may be large, find it modulo 998 244 353.

Input

The first line contains two integers n, m ($1 \le n \le 300, 0 \le m \le \frac{n(n-1)}{2}$): the number of vertices and edges in the graph.

The second line contains a string of length n. If the i-th character is 'W', then the vertex is white. Otherwise, it will be equal to 'B' and be black.

Each of the next m lines contains two vertices u, v ($1 \le u < v \le n$), denoting an edge $u \to v$.

It is guaranteed that the DAG will have no multiple edges.

Output

Print one integer: the number of ways to place at most one token on each vertex such that Alice wins, modulo 998 244 353.

Examples

standard input	standard output
5 4 WWWWW 1 2 2 3 3 4 4 5	30
5 4 BWBWB 1 2 2 3 3 4 4 5	24
9 14 BWWBBBWWW 1 2 1 9 2 3 2 4 2 6 2 8 3 4 3 7 4 7 4 8 5 7 5 9 6 9 7 8	300
10 15 BWBWBBWWBW 1 2 1 5 1 10 2 6 2 8 3 6 3 7 4 10 5 6 5 7 5 8 6 8 6 9 7 10 8 9	228

Note

In the first example, Alice will win in all the cases, where she can make at least one move (because Bob will never be able to move), so the answer is $2^5 - 2$.

Problem B. Brackets

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

There are $2n$ elements divided into n pairs.

For each pair, you should either assign an opening bracket to both elements, or closing bracket to both elements. You need to make the resulting sequence of brackets a correct bracket sequence or determine that it is impossible. If there are several possible solutions, find the solution with the smallest lexicographically string (of $2n$ brackets, '(' is smaller than ')').

Input

The first line contains one integer n ($1 \le n \le 200\,000$).

The next line contains $2n$ integers, $p_1, p_2, \ldots, p_{2n}$ ($1 \le p_i \le n$). All integers from 1 to n appear exactly two times in this sequence.

Output

If it is impossible to choose one type of bracket for each pair to make the derived bracket sequence correct, print "(" (Russian sad smiley). Otherwise, print the desired lexicographically minimal correct bracket sequence.

Examples

standard input	standard output
2 1 2 1 2	()()
1 1 1	(
4 4 3 1 2 3 2 1 4	(
4 3 1 2 1 4 3 2 4	(()()())
4 2 4 3 1 3 4 2 1	()()()()
4 4 4 3 3 1 2 1 2	((((()))))
4 1 3 1 2 4 4 2 3	()(())()

Problem C. Circles

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

Given an array of non-negative integers $s_1, \ldots, s_n$ with $n \geq 3$, let's call a sequence of n non-negative numbers (not necessarily integers) $x_1, x_2, \ldots, x_n$ *balanced* if for each i, the constraint $x_i + x_{i+1} \leq s_i$ is satisfied, where $x_{n+1} = x_1$.

Let's denote $f(s_1, s_2, \ldots, s_n)$ as the largest $x_1 + x_2 + \ldots + x_n$ among all balanced configurations of weights.

You are given an array of non-negative integers $a_1, a_2, \ldots, a_n$.

Find $n - 2$ numbers: $f(a_1, a_2, a_3), f(a_1, a_2, a_3, a_4), \ldots, f(a_1, a_2, a_3, \ldots, a_n)$.

Input

The first line contains one integer n ($3 \leq n \leq 100\,000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 100\,000$).

Output

Print $n - 2$ numbers: $f(a_1, a_2, a_3), f(a_1, a_2, a_3, a_4), \ldots, f(a_1, a_2, a_3, \ldots, a_n)$.

Your answer will be considered correct if the relative or absolute error of all values in it is at most 10^{-9}.

Examples

standard input	standard output
4 20 20 20 15	30.0 35
6 1 2 1 2 1 2	2 2 3 3
12 1 1 1 3 1 1 2 5 3 2 1 2	1.5 2 3 3 4 5 8 8 9 9

Note

In the first example, for the prefix with three elements we can set values $\{10, 10, 10\}$, for the next prefix we can set values $\{10.1, 9.9, 10.1, 4.9\}$.

Problem D. Deja Vu

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

You are given an array $x_1, x_2, \ldots, x_n$.

You need to perform two types of queries on this array.

- Given i and y, set $x_i = y$.
- Given l, find the smallest d among all tuples (a, b, c, d) with $l \le a < b < c < d$ and $x_a < x_b < x_c < x_d$, or reply that there are no such tuples.

Input

The first line contains two integers n, q ($1 \le n, q \le 500\,000$): the number of elements in the array and the number of queries.

The second line contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^9$).

Each of the next q lines contains the description of a query.

If the first integer in the line is equal to 1, then the next two integers are i and y ($1 \le i \le n$, $1 \le y \le 10^9$), describing a query of the first type.

Otherwise, the first integer in the line is equal to 2, and the next integer is equal to l ($1 \le l \le n$), describing a query of the second type.

Output

For each query of the second type, return the smallest d among all tuples (a, b, c, d) such that $l \le a < b < c < d$ and $x_a < x_b < x_c < x_d$, or print "-1" if there are no such tuples.

Example

standard input	standard output
11 10 1 2 3 4 5 10 9 8 7 6 8 2 1 1 3 2 2 1 1 1 2 2 1 2 5 2 6 1 9 6 1 10 7 2 5	4 5 6 -1 -1 11

Problem E. Easiest Sum

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

The function $f(a_1, a_2, \ldots, a_n)$ represents the largest sum of elements on a non-empty subsegment in the array $a_1, a_2, \ldots, a_n$.

You are given an array $a_1, a_2, \ldots, a_n$.

You can spend one coin and decrease any element of a by 1.

Another function, $g(k)$, represents the smallest value of $f(a_1, a_2, \ldots, a_n)$ you can achieve by spending at most k coins.

Find $g(1) + g(2) + \ldots + g(k)$. As this value may be very large, find it modulo $998\,244\,353$.

Input

The first line of input contains one integer, n ($1 \le n \le 100\,000$): the number of elements in a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^8 \le a_i \le 10^8$).

The third line contains one integer k ($1 \le k \le 10^{13}$).

Output

Print $g(1) + g(2) + \ldots + g(k)$, modulo $998\,244\,353$.

Examples

standard input	standard output
5 1 -1 2 -2 3 3	5
3 -3 -5 -35 1	998244349

Note

In the first example, $g(1) = 2, g(2) = 2, g(3) = 1$.

In the second example, $g(1) = -4$.

Problem F. Funny Salesman

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

You are given a tree, and each edge has a non-negative integer weight.

Let $d(u, v)$ — The maximum of the edge weights on the unique simple path between vertices u and v.

Find the largest $\sum_{i=2}^{n} 2^{d(p_{i-1}, p_i)}$ among all permutations of vertices $p_1, p_2, \ldots, p_n$.

Input

The first line contains one integer n ($2 \le n \le 100\,000$): the number of vertices in the tree.

Each of the next $n-1$ lines contains three integers u, v, w ($1 \le u, v \le n, 0 \le w \le 30$), an edge in the tree with endpoints u, v having weight w.

Output

Print one integer: the largest $\sum_{i=2}^{n} 2^{d(p_{i-1}, p_i)}$.

Examples

standard input	standard output
5 1 2 0 2 3 0 3 4 0 4 5 1	6
10 2 1 1 3 1 1 1 4 0 5 1 2 6 4 1 2 7 2 8 4 2 8 9 3 6 10 0	42

Note

In the first example, one of the optimal permutations is $\{4, 5, 3, 2, 1\}$.

Problem G. Graph Coloring

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

You are given a tournament, represented as a complete directed graph (for all pairs i, j of two different vertices, there is exactly one edge among $i \to j$ and $j \to i$), with $n \leq 3000$ vertices. You need to color its edges into 14 colors.

There should be no path $i \to j \to k$ in this graph such that the colors of edges $i \to j$ and $j \to k$ are the same.

It is guaranteed that this is always possible.

Input

The first line of input contains one integer n ($3 \leq n \leq 3000$): the number of vertices in the given tournament.

Next $n - 1$ lines contain the description of the graph: the i-th line contains a binary string with i characters.

If the j-th character in this string is equal to '`1`', then the graph has an edge from $(i+1) \to j$. Otherwise, it has an edge from $j \to (i+1)$.

Output

The output should contain $n - 1$ lines, where the i-th line contains a string with i characters.

The j-th character in this string should be a lowercase Latin letter between '`a`' and '`n`'. If the graph has an edge from $(i+1) \to j$, then this character represents the color of the edge from $(i+1) \to j$. Otherwise it represents the color of the edge from $j \to (i+1)$.

There should be no path $i \to j \to k$ in this graph such that the colors of edges $i \to j$ and $j \to k$ are the same.

Examples

standard input	standard output
3 1 11	a ab
5 1 10 100 0100	a bc def ghij

Problem H. Hidden Graph

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

There is a simple undirected graph with n vertices. This graph has one additional property:

- Any induced subgraph contains a vertex with a degree at most k.

You need to find this hidden graph. You can check for any subset whether it is an independent set. If it is not, we will show you an edge with both ends inside of this independent set.

We won't change the graph during the interaction, so you may assume that it is fixed.

However, we may choose any edge inside the induced subgraph to show.

In other words, in all tests, the graph is fixed, but the interactor is adaptive.

You need to guess the graph in at most $2nk + n$ queries.

Interaction Protocol

The interaction starts with a line containing one integer n: the number of vertices in the hidden graph ($2 \leq n \leq 2000$).

The integer k is not given, but it satisfies the constraint $1 \leq k < n, nk \leq 2000$.

After that, you can make queries.

To make a query, print one line containing "`?` k" ($1 \leq k \leq n$) and then k distinct integers $v_1, v_2 \ldots, v_k$ ($1 \leq v_i \leq n$). Separate consecutive integers by single spaces. Then `flush` the output.

After each query, read one line with two integers i, j. If there are no edges in the induced subgraph $v_1, v_2, \ldots, v_k$, $i = j = -1$, otherwise, $i \neq j, i \in v, j \in v$, and there is an edge with ends i and j in the graph.

When you find the graph, in the first line print one line containing "`!` m". And the next m lines should contain the description of edges of the graph, each of them should contain two integers u, v ($1 \leq u, v \leq n$), denoting the indices of vertices connected by an edge.

Your solution will get `Wrong Answer` or `Time Limit Exceeded` if you make more than $2nk + n$ queries.

Your solution will get `Idleness Limit Exceeded` if you don't print anything or forget to flush the output.

To `flush`, you need to do the following right after printing a query and a line break:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Example

standard input	standard output
3 1 2 2 3 1 3	? 2 1 2 ? 2 2 3 ? 2 1 3 ! 3 1 2 2 3 1 3

Problem I. Insects

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

You have n black ants in your terrarium, and the i-th black ant lives at coordinate (a_i, b_i).

Each day for the next m days, you will buy a new ant for your terrarium. You are only buying white ants, and the i-th white ant that you are buying will live at coordinate (x_i, y_i).

Each day, you feed some of your insects. If you feed an insect, the insect will not be hungry in that day. If the i-th white ant is hungry and the j-th black ant is hungry, and $x_i \geq a_j$ and $y_i \geq b_j$, they will fight. Find, for each day, the smallest number of ants to feed such that there are no fights.

Input

The first line contains one integer n ($1 \leq n \leq 100\,000$): the number of black ants in your terrarium.

Each of the next n lines contains the description of black ants. The i-th of them contain two integers, a_i, b_i ($0 \leq a_i, b_i \leq 100\,000$).

The next line contains one integer m ($1 \leq m \leq 100\,000$): the number of days in which you are going to buy new white ants.

Each of the next m lines contains the description of white ants in the order you buy them, such that the i-th of them contains two integers, x_i, y_i ($0 \leq x_i, y_i \leq 100\,000$).

Note that different ants can live at points with the same coordinates.

Output

Print m integers, such that the i-th of them equals the smallest number of ants that you should feed to avoid fights among the black ants $1, 2, \ldots, n$ and the white ants $1, 2, \ldots, i$.

Example

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

Problem J. Joining Points

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

You are given $3n$ different points on a circle. Each of these points is colored in one of n colors, such that each color appears exactly three times.

You want to draw n non-intersecting arcs with ends on given points.

For these arcs, the ends of the arc should have equal colors, and no other point on the arc should have this color.

Note that you are drawing arcs, not chords.

Find the number of suitable drawings, modulo 998 244 353.

Input

The first line of input contains one integer n ($1 \le n \le 200\,000$): the number of colors.

Next line contains $3n$ integers $c_1, c_2, \ldots, c_{3n}$ ($1 \le c_i \le n$): the color of the i-th point on the circle, in clockwise order.

It is guaranteed that each color appears exactly three times.

Output

Print one integer: the number of suitable drawings modulo 998 244 353.

Examples

standard input	standard output
3 1 1 1 2 2 2 3 3 3	8
2 1 1 2 2 1 2	3