Codeforces Round #522 (Div. 1, based on Technocup 2019 Elimination Round 3)

A. Barcelonian Distance

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem we consider a very simplified model of Barcelona city.

Barcelona can be represented as a plane with streets of kind $x = c$ and $y = c$ for every integer c (that is, the rectangular grid). However, there is a detail which makes Barcelona different from Manhattan. There is an avenue called Avinguda Diagonal which can be represented as a the set of points (x, y) for which $ax + by + c = 0$.

One can walk along streets, including the avenue. You are given two integer points A and B somewhere in Barcelona. Find the minimal possible distance one needs to travel to get to B from A.

Input

The first line contains three integers a, b and c ($-10^9 \le a, b, c \le 10^9$, at least one of a and b is not zero) representing the Diagonal Avenue.

The next line contains four integers x_1, y_1, x_2 and y_2 ($-10^9 \le x_1, y_1, x_2, y_2 \le 10^9$) denoting the points $A = (x_1, y_1)$ and $B = (x_2, y_2)$.

Output

Find the minimum possible travel distance between A and B. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

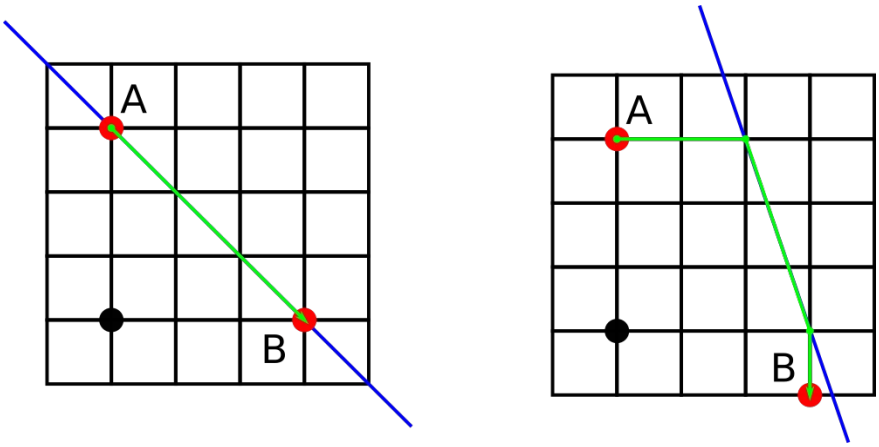
Examples

input
1 1 -3 0 3 3 0
output
4.2426406871

input
3 1 -9 0 3 3 -1
output
6.1622776602

Note

The first example is shown on the left picture while the second example us shown on the right picture below. The avenue is shown with blue, the origin is shown with the black dot.



B. The Unbearable Lightness of Weights

time limit per test: 1 second

memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a set of n weights. You know that their masses are a_1, a_2, ..., a_n grams, but you don't know which of them has which mass. You can't distinguish the weights.

However, your friend does know the mass of each weight. You can ask your friend to give you exactly k weights with the total mass m (both parameters k and m are chosen by you), and your friend will point to any valid subset of weights, if it is possible.

You are allowed to make this query only once. Find the maximum possible number of weights you can reveal after this query.

Input
The first line contains a single integer n ($1 \le n \le 100$) — the number of weights.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$) — the masses of the weights.

Output
Print the maximum number of weights you can learn the masses for after making a single query.

Examples

input
4 1 4 2 2
output
2

input
6 1 2 4 4 4 9
output
2

Note
In the first example we can ask for a subset of two weights with total mass being equal to 4, and the only option is to get $\{2, 2\}$.

Another way to obtain the same result is to ask for a subset of two weights with the total mass of 5 and get $\{1, 4\}$. It is easy to see that the two remaining weights have mass of 2 grams each.

In the second example we can ask for a subset of two weights with total mass being 8, and the only answer is $\{4, 4\}$. We can prove it is not possible to learn masses for three weights in one query, but we won't put the proof here.

C. Vasya and Maximum Matching

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya has got a tree consisting of n vertices. He wants to delete some (possibly zero) edges in this tree such that the maximum matching in the resulting graph is unique. He asks you to calculate the number of ways to choose a set of edges to remove.

A matching in the graph is a subset of its edges such that there is no vertex incident to two (or more) edges from the subset. A maximum matching is a matching such that the number of edges in the subset is maximum possible among all matchings in this graph.

Since the answer may be large, output it modulo 998244353.

Input
The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$) — the number of vertices in the tree.

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n, u \ne v$) denoting an edge between vertex u and vertex v. It is guaranteed that these edges form a tree.

Output
Print one integer — the number of ways to delete some (possibly empty) subset of edges so that the maximum matching in the resulting graph is unique. Print the answer modulo 998244353.

Examples

input
4 1 2 1 3 1 4

output
4

input
4 1 2 2 3 3 4
output
6

input
1
output
1

Note

Possible ways to delete edges in the first example:

- delete $(1, 2)$ and $(1, 3)$.
- delete $(1, 2)$ and $(1, 4)$.
- delete $(1, 3)$ and $(1, 4)$.
- delete all edges.

Possible ways to delete edges in the second example:

- delete no edges.
- delete $(1, 2)$ and $(2, 3)$.
- delete $(1, 2)$ and $(3, 4)$.
- delete $(2, 3)$ and $(3, 4)$.
- delete $(2, 3)$.
- delete all edges.

D. Chattering

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n parrots standing in a circle. Each parrot has a certain level of respect among other parrots, namely r_i. When a parrot with respect level x starts chattering, x neighbours to the right and to the left of it start repeating the same words in 1 second. Their neighbours then start repeating as well, and so on, until all the birds begin to chatter.

You are given the respect levels of all parrots. For each parrot answer a question: if this certain parrot starts chattering, how many seconds will pass until all other birds will start repeating it?

Input

In the first line of input there is a single integer n, the number of parrots ($1 \leq n \leq 10^5$).

In the next line of input there are n integers r_1, ..., r_n, the respect levels of parrots in order they stand in the circle ($1 \leq r_i \leq n$).

Output

Print n integers. i-th of them should equal the number of seconds that is needed for all parrots to start chattering if the i-th parrot is the first to start.

Examples

input
4 1 1 4 1
output
2 2 1 2

input
8 1 2 2 1 5 1 3 1
output
3 3 2 2 1 2 2 3

E. Negative Time Summation

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everyone knows that computers become faster and faster. Recently Berland scientists have built a machine that can move itself back in time!

More specifically, it works as follows. It has an infinite grid and a robot which stands on one of the cells. Each cell of the grid can either be empty or contain `0` or `1`. The machine also has a program which consists of instructions, which are being handled one by one. Each instruction is represented by exactly one symbol (letter or digit) and takes exactly one unit of time (say, second) to be performed, except the last type of operation (it's described below). Here they are:

- `0` or `1`: the robot places this number into the cell he is currently at. If this cell wasn't empty before the operation, its previous number is replaced anyway.
- `e`: the robot **e**rases the number into the cell he is at.
- `l`, `r`, `u` or `d`: the robot goes one cell to the **l**eft/**r**ight/**u**p/**d**own.
- `s`: the robot **s**tays where he is for a unit of time.
- `t`: let x be 0, if the cell with the robot is empty, otherwise let x be one more than the digit in this cell (that is, $x = 1$ if the digit in this cell is 0, and $x = 2$ if the digit is 1). Then the machine **t**ravels x seconds back in time. Note that this doesn't change the instructions order, but it changes the position of the robot and the numbers in the grid as they were x units of time ago. You can consider this instruction to be equivalent to a `Ctrl-Z` pressed x times.

For example, let the board be completely empty, and the program be `sr1t0`. Let the robot initially be at $(0, 0)$.

- [now is the moment 0, the command is `s`]: we do nothing.
- [now is the moment 1, the command is `r`]: we are now at $(1, 0)$.
- [now is the moment 2, the command is `1`]: we are at $(1, 0)$, and this cell contains 1.
- [now is the moment 3, the command is `t`]: we travel $1 + 1 = 2$ moments back, that is, to the moment 1.
- [now is the moment 1, the command is `0`]: we are again at $(0, 0)$, and the board is clear again, but after we follow this instruction, this cell has 0 in it. We've just rewritten the history. The consequences of the third instruction have never happened.

Now Berland scientists want to use their machine in practice. For example, they want to be able to add two integers.

Assume that the initial state of the machine is as follows:

- One positive integer is written in binary on the grid in such a way that its right bit is at the cell $(0, 1)$, from left to right from the highest bit to the lowest bit.
- The other positive integer is written in binary on the grid in such a way that its right bit is at the cell $(0, 0)$, from left to right from the highest bit to the lowest bit.
- All the other cells are empty.
- The robot is at $(0, 0)$.
- We consider this state to be always in the past; that is, if you manage to travel to any negative moment, the board was always as described above, and the robot was at $(0, 0)$ for eternity.

You are asked to write a program after which

- The robot stands on a non-empty cell,
- If we read the number starting from the cell with the robot and moving to the right until the first empty cell, this will be $a + b$ in binary, from the highest bit to the lowest bit.

Note that there are no restrictions on other cells. In particular, there may be a digit just to the left to the robot after all instructions.

In each test you are given up to 1000 pairs (a, b), and your program must work for all these pairs. Also since the machine's memory is not very big, your program must consist of no more than 10^5 instructions.

Input

The first line contains the only integer t ($1 \le t \le 1000$) standing for the number of testcases. Each of the next t lines consists of two positive integers a and b ($1 \le a, b < 2^{30}$) in decimal.

Output

Output the only line consisting of no more than 10^5 symbols from `01eslrudt` standing for your program.

Note that formally you may output different programs for different tests.

Example

input
2 123456789 987654321 555555555 555555555
output
01111101010111111101110111110101010111011111110101011101010101110lr