# Codeforces Round #641 (Div. 1)

## A. Orac and LCM

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

For the multiset of positive integers $s = \{s_1, s_2, \ldots, s_k\}$, define the Greatest Common Divisor (GCD) and Least Common Multiple (LCM) of $s$ as follow:

- $\gcd(s)$ is the maximum positive integer $x$, such that all integers in $s$ are divisible on $x$.
- $\operatorname{lcm}(s)$ is the minimum positive integer $x$, that divisible on all integers from $s$.

For example, $\gcd(\{8, 12\}) = 4, \gcd(\{12, 18, 6\}) = 6$ and $\operatorname{lcm}(\{4, 6\}) = 12$. Note that for any positive integer $x$, $\gcd(\{x\}) = \operatorname{lcm}(\{x\}) = x$.

Orac has a sequence $a$ with length $n$. He come up with the multiset $t = \{\operatorname{lcm}(\{a_i, a_j\}) \mid i < j\}$, and asked you to find the value of $\gcd(t)$ for him. In other words, you need to calculate the GCD of LCMs of all pairs of elements in the given sequence.

### Input

The first line contains one integer $n$ $(2 \le n \le 100\,000)$.

The second line contains $n$ integers, $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 200\,000)$.

### Output

Print one integer: $\gcd(\{\operatorname{lcm}(\{a_i, a_j\}) \mid i < j\})$.

**Examples**

| input |
| --- |
| 2<br>1 1 |
| **output** |
| 1 |

| input |
| --- |
| 4<br>10 24 40 80 |
| **output** |
| 40 |

| input |
| --- |
| 10<br>540 648 810 648 720 540 594 864 972 648 |
| **output** |
| 54 |

### Note

For the first example, $t = \{\operatorname{lcm}(\{1, 1\})\} = \{1\}$, so $\gcd(t) = 1$.

For the second example, $t = \{120, 40, 80, 120, 240, 80\}$, and it's not hard to see that $\gcd(t) = 40$.

## B. Orac and Medians

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

Slime has a sequence of positive integers $a_1, a_2, \ldots, a_n$.

In one operation Orac can choose an arbitrary subsegment $[l \ldots r]$ of this sequence and replace all values $a_l, a_{l+1}, \ldots, a_r$ to the value of median of $\{a_l, a_{l+1}, \ldots, a_r\}$.

In this problem, for the integer multiset $s$, the median of $s$ is equal to the $\lfloor \frac{|s|+1}{2} \rfloor$-th smallest number in it. For example, the median

of $\{1, 4, 4, 6, 5\}$ is $4$, and the median of $\{1, 7, 5, 8\}$ is $5$.

Slime wants Orac to make $a_1 = a_2 = \ldots = a_n = k$ using these operations.

Orac thinks that it is impossible, and he does not want to waste his time, so he decided to ask you if it is possible to satisfy the Slime's requirement, he may ask you these questions several times.

### Input
The first line of the input is a single integer $t$: the number of queries.

The first line of each query contains two integers $n$ $(1 \le n \le 100\,000)$ and $k$ $(1 \le k \le 10^9)$, the second line contains $n$ positive integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$

The total sum of $n$ is at most $100\,000$.

### Output
The output should contain $t$ lines. The $i$-th line should be equal to 'yes' if it is possible to make all integers $k$ in some number of operations or 'no', otherwise. You can print each letter in lowercase or uppercase.

**Example**

| input |
|---|
| 5<br>5 3<br>1 5 2 6 1<br>1 6<br>6<br>3 2<br>1 2 3<br>4 3<br>3 1 2 3<br>10 3<br>1 2 3 4 5 6 7 8 9 10 |
| **output** |
| no<br>yes<br>yes<br>no<br>yes |

### Note
In the first query, Orac can't turn all elements into $3$.

In the second query, $a_1 = 6$ is already satisfied.

In the third query, Orac can select the complete array and turn all elements into $2$.

In the fourth query, Orac can't turn all elements into $3$.

In the fifth query, Orac can select $[1, 6]$ at first and then select $[2, 10]$.

## C. Orac and Game of Life

time limit per test: 2 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

**Please notice the unusual memory limit of this problem.**

Orac likes games. Recently he came up with the new game, "`Game of Life`".

You should play this game on a black and white grid with $n$ rows and $m$ columns. Each cell is either black or white.

For each iteration of the game (the initial iteration is $0$), the color of each cell will change under the following rules:

- If there are no adjacent cells with the same color as this cell on the current iteration, the color of it on the next iteration will be the same.
- Otherwise, the color of the cell on the next iteration will be different.

Two cells are adjacent if they have a mutual edge.

Now Orac has set an initial situation, and he wants to know for the cell $(i, j)$ (in $i$-th row and $j$-th column), what will be its color at the iteration $p$. He may ask you these questions several times.

### Input
The first line contains three integers $n, m, t$ $(1 \le n, m \le 1000, 1 \le t \le 100\,000)$, representing the number of rows, columns, and the number of Orac queries.

Each of the following $n$ lines contains a binary string of length $m$, the $j$-th character in $i$-th line represents the initial color of cell

$(i, j)$. '0' stands for white, '1' stands for black.

Each of the following $t$ lines contains three integers $i, j, p$ $(1 \le i \le n, 1 \le j \le m, 1 \le p \le 10^{18})$, representing a query from Orac.

## Output

Print $t$ lines, in $i$-th line you should print the answer to the $i$-th query by Orac. If the color of this cell is black, you should print '1'; otherwise, you should write '0'.

### Examples

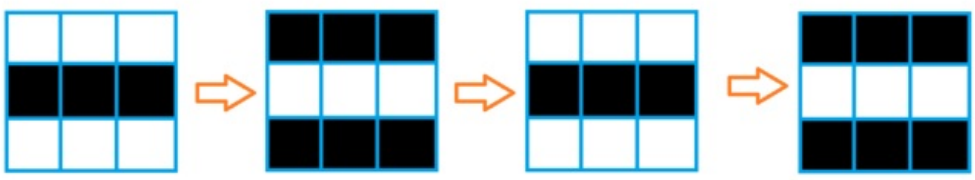
| input |
| --- |
| 3 3 3<br>000<br>111<br>000<br>1 1 1<br>2 2 2<br>3 3 3 |
| **output** |
| 1<br>1<br>1 |

| input |
| --- |
| 5 2 2<br>01<br>10<br>01<br>10<br>01<br>1 1 4<br>5 1 4 |
| **output** |
| 0<br>0 |

| input |
| --- |
| 5 5 3<br>01011<br>10110<br>01101<br>11010<br>10101<br>1 1 4<br>1 2 3<br>5 5 3 |
| **output** |
| 1<br>0<br>1 |

| input |
| --- |
| 1 1 3<br>0<br>1 1 1<br>1 1 2<br>1 1 3 |
| **output** |
| 0<br>0<br>0 |

## Note



For the first example, the picture above shows the initial situation and the color of cells at the iteration $1$, $2$, and $3$. We can see that the color of $(1, 1)$ at the iteration $1$ is black, the color of $(2, 2)$ at the iteration $2$ is black, and the color of $(3, 3)$ at the iteration $3$ is also black.

For the second example, you can prove that the cells will never change their colors.

# D. Slime and Biscuits

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

Slime and his $n$ friends are at a party. Slime has designed a game for his friends to play.

At the beginning of the game, the $i$-th player has $a_i$ biscuits. At each second, Slime will choose a biscuit randomly uniformly among all $a_1 + a_2 + \ldots + a_n$ biscuits, and the owner of this biscuit will give it to a random uniform player among $n - 1$ players except himself. The game stops when one person will have all the biscuits.

As the host of the party, Slime wants to know the expected value of the time that the game will last, to hold the next activity on time.

For convenience, as the answer can be represented as a rational number $\frac{p}{q}$ for coprime $p$ and $q$, you need to find the value of $(p \cdot q^{-1}) \mod 998\,244\,353$. You can prove that $q \mod 998\,244\,353 \neq 0$.

## Input

The first line contains one integer $n$ $(2 \leq n \leq 100\,000)$: the number of people playing the game.

The second line contains $n$ non-negative integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_1 + a_2 + \cdots + a_n \leq 300\,000)$, where $a_i$ represents the number of biscuits the $i$-th person own at the beginning.

## Output

Print one integer: the expected value of the time that the game will last, modulo $998\,244\,353$.

### Examples

| input |
| --- |
| 2<br>1 1 |
| **output** |
| 1 |

| input |
| --- |
| 2<br>1 2 |
| **output** |
| 3 |

| input |
| --- |
| 5<br>0 0 0 0 35 |
| **output** |
| 0 |

| input |
| --- |
| 5<br>8 4 2 0 1 |
| **output** |
| 801604029 |

## Note

For the first example, in the first second, the probability that player $1$ will give the player $2$ a biscuit is $\frac{1}{2}$, and the probability that player $2$ will give the player $1$ a biscuit is $\frac{1}{2}$. But anyway, the game will stop after exactly $1$ second because only one player will occupy all biscuits after $1$ second, so the answer is $1$.

# E. Slime and Hats

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

Slime and Orac are holding a turn-based game. In a big room, there are $n$ players sitting on the chairs, looking forward to a column and each of them is given a number: player $1$ sits in the front of the column, player $2$ sits directly behind him; player $3$ sits directly behind player $2$, and so on; player $n$ sits directly behind player $n - 1$. Each player wears a hat that is either black or white. As each player faces forward, player $i$ knows the color of player $j$'s hat if and only if $i$ is larger than $j$.

At the start of each turn, Orac will tell **whether there exists a player wearing a black hat in the room**.

After Orac speaks, if the player can uniquely identify the color of his hat, he will put his hat on the chair, stand up and leave the room. All players are smart, so if it is possible to understand the color of their hat using the obtained information during this and previous rounds, they will understand it.

In each turn, all players who know the color of their hats will leave at the same time in this turn, which means a player can only leave in the next turn if he gets to know the color of his hat only after someone left the room at this turn.

Note that when the player needs to leave, he will put the hat on the chair before leaving, so the players ahead of him still cannot see his hat.

The $i$-th player will know who exactly left the room among players $1, 2, \ldots, i-1$, and how many players among $i+1, i+2, \ldots, n$ have left the room.

Slime stands outdoor. He watches the players walking out and records the numbers of the players and the time they get out. Unfortunately, Slime is so careless that he has only recorded some of the data, and this given data is in the format "`player` $x$ `leaves in the` $y$`-th round`".

Slime asked you to tell him the color of each player's hat. If there are multiple solutions, you can find any of them.

### Input
The first line contains a integer $n$ $(1 \leq n \leq 200\,000)$.

The second line contains $n$ integers $t_1, t_2, \ldots, t_n$ $(0 \leq t_i \leq 10^{15})$. If $t_i = 0$, then there are no data about player $i$; otherwise it means player $i$ leaves in the $t_i$-th round.

At least one solution exists for the given input.

### Output
Print one binary string of $n$ characters. The $i$-th character of the string should be '1' if player $i$ wears a black hat and should be '0', otherwise. If there are multiple solutions, you can print any of them.

**Examples**

| input |
| --- |
| 5<br>0 1 1 0 0 |
| **output** |
| 00000 |

| input |
| --- |
| 5<br>0 2 2 0 0 |
| **output** |
| 00001 |

| input |
| --- |
| 5<br>0 0 0 0 0 |
| **output** |
| 00000 |

| input |
| --- |
| 5<br>4 4 0 4 4 |
| **output** |
| 00100 |

### Note
In the first example, for the given solution, all the players wear white hats. In the first turn, Orac tells all the players that there are no players wearing a black hat, so each player knows that he is wearing a white hat, and he will leave in the first turn.

In the second example, for the given solution, the player $5$ wears a black hat, other players wear white hats. Orac tells all the players that there exists a player wearing a black hat, and player $5$ know that the other players are all wearing white hats, so he can infer that he is wearing a black hat; therefore he leaves in the first turn, other players leave in the second turn. Note that other players can infer that they are wearing white hats immediately after player $5$ leaves, but they have to wait for the next turn to leave according to the rule.

In the third example, there is no information about the game, so any output is correct.

## F1. Slime and Sequences (Easy Version)

time limit per test: 2 seconds

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

**Note that the only differences between easy and hard versions are the constraints on $n$ and the time limit. You can make hacks only if all versions are solved.**

Slime is interested in sequences. He defined **good** positive integer sequences $p$ of length $n$ as follows:

- For each $k > 1$ that presents in $p$, there should be at least one pair of indices $i, j$, such that $1 \le i < j \le n$, $p_i = k - 1$ and $p_j = k$.

For the given integer $n$, the set of all good sequences of length $n$ is $s_n$. For the fixed integer $k$ and the sequence $p$, let $f_p(k)$ be the number of times that $k$ appears in $p$. For each $k$ from $1$ to $n$, Slime wants to know the following value:

$$\left(\sum_{p \in s_n} f_p(k)\right) \bmod 998\,244\,353$$

### Input
The first line contains one integer $n$ $(1 \le n \le 5000)$.

### Output
Print $n$ integers, the $i$-th of them should be equal to $\left(\sum_{p \in s_n} f_p(i)\right) \bmod 998\,244\,353$.

**Examples**

| input |
| --- |
| 2 |
| **output** |
| 3 1 |

| input |
| --- |
| 3 |
| **output** |
| 10 7 1 |

| input |
| --- |
| 1 |
| **output** |
| 1 |

### Note
In the first example, $s = \{[1, 1], [1, 2]\}$.

In the second example, $s = \{[1, 1, 1], [1, 1, 2], [1, 2, 1], [1, 2, 2], [2, 1, 2], [1, 2, 3]\}$.

In the third example, $s = \{[1]\}$.

## F2. Slime and Sequences (Hard Version)

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

**Note that the only differences between easy and hard versions are the constraints on $n$ and the time limit. You can make hacks only if all versions are solved.**

Slime is interested in sequences. He defined **good** positive integer sequences $p$ of length $n$ as follows:

- For each $k > 1$ that presents in $p$, there should be at least one pair of indices $i, j$, such that $1 \le i < j \le n$, $p_i = k - 1$ and $p_j = k$.

For the given integer $n$, the set of all good sequences of length $n$ is $s_n$. For the fixed integer $k$ and the sequence $p$, let $f_p(k)$ be the number of times that $k$ appears in $p$. For each $k$ from $1$ to $n$, Slime wants to know the following value:

$$\left(\sum_{p \in s_n} f_p(k)\right) \bmod 998\,244\,353$$

### Input

The first line contains one integer $n$ $(1 \le n \le 100\,000)$.

## Output

Print $n$ integers, the $i$-th of them should be equal to $\left(\sum_{p \in s_n} f_p(i)\right) \bmod 998\,244\,353$.

### Examples

| input |
| --- |
| 2 |
| **output** |
| 3 1 |

| input |
| --- |
| 3 |
| **output** |
| 10 7 1 |

| input |
| --- |
| 1 |
| **output** |
| 1 |

## Note

In the first example, $s = \{[1, 1], [1, 2]\}$.

In the second example, $s = \{[1, 1, 1], [1, 1, 2], [1, 2, 1], [1, 2, 2], [2, 1, 2], [1, 2, 3]\}$.

In the third example, $s = \{[1]\}$.