Codeforces Round #641 (Div. 2)

A. Orac and Factors

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

Orac is studying number theory, and he is interested in the properties of divisors.

For two positive integers a and b, a is a divisor of b if and only if there exists an integer c, such that $a \cdot c = b$.

For $n \geq 2$, we will denote as $f(n)$ the smallest positive divisor of n, except 1.

For example, $f(7) = 7, f(10) = 2, f(35) = 5$.

For the fixed integer n, Orac decided to add $f(n)$ to n.

For example, if he had an integer $n = 5$, the new value of n will be equal to 10. And if he had an integer $n = 6$, n will be changed to 8.

Orac loved it so much, so he decided to repeat this operation several times.

Now, for two positive integers n and k, Orac asked you to add $f(n)$ to n exactly k times (note that n will change after each operation, so $f(n)$ may change too) and tell him the final value of n.

For example, if Orac gives you $n = 5$ and $k = 2$, at first you should add $f(5) = 5$ to $n = 5$, so your new value of n will be equal to $n = 10$, after that, you should add $f(10) = 2$ to 10, so your new (and the final!) value of n will be equal to 12.

Orac may ask you these queries many times.

Input

The first line of the input is a single integer t $(1 \leq t \leq 100)$: the number of times that Orac will ask you.

Each of the next t lines contains two positive integers n, k $(2 \leq n \leq 10^6, 1 \leq k \leq 10^9)$, corresponding to a query by Orac.

It is guaranteed that the total sum of n is at most 10^6.

Output

Print t lines, the i-th of them should contain the final value of n in the i-th query by Orac.

Example

input

```
3
5 1
8 2
3 4
```

output

```
10
12
12
```

Note

In the first query, $n = 5$ and $k = 1$. The divisors of 5 are 1 and 5, the smallest one except 1 is 5. Therefore, the only operation adds $f(5) = 5$ to 5, and the result is 10.

In the second query, $n = 8$ and $k = 2$. The divisors of 8 are $1, 2, 4, 8$, where the smallest one except 1 is 2, then after one operation 8 turns into $8 + (f(8) = 2) = 10$. The divisors of 10 are $1, 2, 5, 10$, where the smallest one except 1 is 2, therefore the answer is $10 + (f(10) = 2) = 12$.

In the third query, n is changed as follows: $3 \to 6 \to 8 \to 10 \to 12$.

B. Orac and Models

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

There are n models in the shop numbered from 1 to n, with sizes $s_1, s_2, \ldots, s_n$.

Orac will buy some of the models and will arrange them in the order of increasing numbers (i.e. indices, but not sizes).

Orac thinks that the obtained arrangement is **beatiful**, if for any two adjacent models with indices i_j and i_{j+1} (note that $i_j < i_{j+1}$, because Orac arranged them properly), i_{j+1} is divisible by i_j and $s_{i_j} < s_{i_{j+1}}$.

For example, for 6 models with sizes $\{3, 6, 7, 7, 7, 7\}$, he can buy models with indices 1, 2, and 6, and the obtained arrangement will be beautiful. Also, note that the arrangement with exactly one model is also considered beautiful.

Orac wants to know the maximum number of models that he can buy, and he may ask you these queries many times.

Input

The first line contains one integer t $(1 \le t \le 100)$: the number of queries.

Each query contains two lines. The first line contains one integer n $(1 \le n \le 100\,000)$: the number of models in the shop, and the second line contains n integers $s_1, \dots, s_n$ $(1 \le s_i \le 10^9)$: the sizes of models.

It is guaranteed that the total sum of n is at most $100\,000$.

Output

Print t lines, the i-th of them should contain the maximum number of models that Orac can buy for the i-th query.

Example

input

```
4
4
5 3 4 6
7
1 4 2 3 6 4 9
5
5 4 3 2 1
1
9
```

output

```
2
3
1
1
```

Note

In the first query, for example, Orac can buy models with indices 2 and 4, the arrangement will be beautiful because 4 is divisible by 2 and 6 is more than 3. By enumerating, we can easily find that there are no beautiful arrangements with more than two models.

In the second query, Orac can buy models with indices 1, 3, and 6. By enumerating, we can easily find that there are no beautiful arrangements with more than three models.

In the third query, there are no beautiful arrangements with more than one model.

C. 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 $s = \{s_1, s_2, \dots, s_k\}$, 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.
- $\text{lcm}(s)$ 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 $\text{lcm}(\{4, 6\}) = 12$. Note that for any positive integer x, $\gcd(\{x\}) = \text{lcm}(\{x\}) = x$.

Orac has a sequence a with length n. He come up with the multiset $t = \{\text{lcm}(\{a_i, a_j\}) \mid i < j\}$, 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, $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 200\,000)$.

Output

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

Examples

input

```
2
```

```
1 1
```

output

```
1
```

input

```
4
10 24 40 80
```

output

```
40
```

input

```
10
540 648 810 648 720 540 594 864 972 648
```

output

```
54
```

Note

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

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

D. 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
5 3
1 5 2 6 1
1 6
6
3 2
1 2 3
4 3
3 1 2 3
10 3
1 2 3 4 5 6 7 8 9 10
```

output

```
no
yes
yes
no
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 $[2, 10]$.

E. 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
000
111
000
1 1 1
2 2 2
3 3 3
```

output

```
1
1
1
```

input

```
5 2 2
01
10
01
10
01
1 1 4
5 1 4
```

output

```
0
0
```

input

```
5 5 3
01011
```

```
10110
01101
11010
10101
1 1 4
1 2 3
5 5 3
```

output

```
1
0
1
```

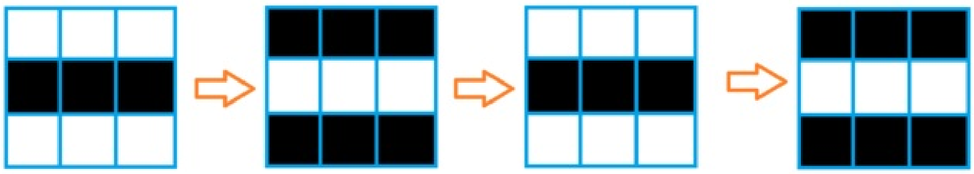
input

```
1 1 3
0
1 1 1
1 1 2
1 1 3
```

output

```
0
0
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.

F. 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 $1 \leq i < j \leq n$, $p_i = k - 1$ 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 $(1 \leq n \leq 5000)$.

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, $s = \{[1]\}$.