A. T-primes

time limit per test: 2 seconds
memory limit per test: 256 megabytes

We know that prime numbers are positive integers that have exactly two distinct positive divisors. Similarly, we'll call a positive integer t T-prime, if t has exactly three distinct positive divisors.

You are given an array of n positive integers. For each of them determine whether it is T-prime or not.

Input

The first line contains a single positive integer, n ($1 \le n \le 10^5$), showing how many numbers are in the array. The next line contains n space-separated integers x_i ($1 \le x_i \le 10^{12}$).

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is advised to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print n lines: the i-th line should contain "YES" (without the quotes), if number x_i is T-prime, and "NO" (without the quotes), if it isn't.

Examples

input

```
3
4 5 6
```

output

```
YES
NO
NO
```

Note

The given test has three numbers. The first number 4 has exactly three divisors — 1, 2 and 4, thus the answer for this number is "YES". The second number 5 has two divisors (1 and 5), and the third number 6 has four divisors (1, 2, 3, 6), hence the answer for them is "NO".

B. Soldier and Number Game

time limit per test: 3 seconds
memory limit per test: 256 megabytes

Two soldiers are playing a game. At the beginning first of them chooses a positive integer n and gives it to the second soldier. Then the second one tries to make maximum possible number of rounds. Each round consists of choosing a positive integer $x > 1$, such that n is divisible by x and replacing n with n / x. When n becomes equal to 1 and there is no more possible valid moves the game is over and the score of the second soldier is equal to the number of rounds he performed.

To make the game more interesting, first soldier chooses n of form $a! / b!$ for some positive integer a and b ($a \geq b$). Here by $k!$ we denote the *factorial* of k that is defined as a product of all positive integers not large than k.

What is the maximum possible score of the second soldier?

Input

First line of input consists of single integer t ($1 \leq t \leq 1\,000\,000$) denoting number of games soldiers play.

Then follow t lines, each contains pair of integers a and b ($1 \leq b \leq a \leq 5\,000\,000$) defining the value of n for a game.

Output

For each game output a maximum score that the second soldier can get.

Examples

input

```
2
3 1
6 3
```

output

```
2
5
```

C. Prime Swaps

time limit per test: 2 seconds

memory limit per test: 256 megabytes

You have an array $a[1], a[2], ..., a[n]$, containing distinct integers from 1 to n. Your task is to sort this array in increasing order with the following operation (you may need to apply it multiple times):

- choose two indexes, i and j $(1 \le i < j \le n$; $(j - i + 1)$ is a prime number);
- swap the elements on positions i and j; in other words, you are allowed to apply the following sequence of assignments: $tmp = a[i], a[i] = a[j], a[j] = tmp$ (tmp is a temporary variable).

You do not need to minimize the number of used operations. However, you need to make sure that there are at most $5n$ operations.

Input

The first line contains integer n $(1 \le n \le 10^5)$. The next line contains n distinct integers $a[1], a[2], ..., a[n]$ $(1 \le a[i] \le n)$.

Output

In the first line, print integer k $(0 \le k \le 5n)$ — the number of used operations. Next, print the operations. Each operation must be printed as "i j" $(1 \le i < j \le n$; $(j - i + 1)$ is a prime).

If there are multiple answers, you can print any of them.

Examples

input
```
3
3 2 1
```

output
```
1
1 3
```

input
```
2
1 2
```

output
```
0
```

input
```
4
4 2 3 1
```

output
```
3
2 4
1 2
2 4
```

D. Reducing Fractions

time limit per test: 2 seconds

memory limit per test: 256 megabytes

To confuse the opponents, the Galactic Empire represents fractions in an unusual format. The fractions are represented as two sets of integers. The product of numbers from the first set gives the fraction numerator, the product of numbers from the second set gives the fraction denominator. However, it turned out that the programs that work with fractions in this representations aren't complete, they lack supporting the operation of reducing fractions. Implement this operation and the Empire won't forget you.

Input

The first input line contains two space-separated integers n, m $(1 \le n, m \le 10^5)$ that show how many numbers the first set (the numerator) and the second set (the denominator) contain, correspondingly.

The second line contains n space-separated integers: $a_1, a_2, ..., a_n$ $(1 \le a_i \le 10^7)$ — the numbers that are multiplied to produce the numerator.

The third line contains m space-separated integers: $b_1, b_2, ..., b_m$ $(1 \le b_i \le 10^7)$ — the numbers that are multiplied to produce the denominator.

Output

Print the answer to the problem in the form, similar to the form of the input data. The number of values in the sets you print n_{out}, m_{out} must satisfy the inequality $1 \le n_{out}, m_{out} \le 10^5$, and the actual values in the sets $a_{out, i}$ and $b_{out, i}$ must satisfy the inequality $1 \le a_{out, i}, b_{out, i} \le 10^7$.

Separate the values in the lines by spaces. The printed fraction must be reduced, that is, there mustn't be such integer x $(x > 1)$, that the numerator and the denominator of the printed fraction are divisible by x. If there are several matching answers, print any of them.

Examples

input

```
3 2
100 5 2
50 10
```

output

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

input

```
4 3
2 5 10 20
100 1 3
```

output

```
1 1
20
3
```

Note

In the first test sample the numerator equals 1000, the denominator equals 500. If we reduce fraction 1000/500 by the greatest common divisor of the numerator and the denominator (by 500), we obtain fraction 2/1.

In the second test sample the numerator equals 2000, the denominator equals 300. If we reduce fraction 2000/300 by the greatest common divisor of the numerator and the denominator (by 100), we obtain fraction 20/3.

E. On Number of Decompositions into Multipliers

time limit per test: 1 second

memory limit per test: 256 megabytes

You are given an integer m as a product of integers $a_1, a_2, \dots a_n$ $(m = \prod_{i=1}^{n} a_i)$. Your task is to find the number of distinct decompositions of number m into the product of n ordered positive integers.

Decomposition into n products, given in the input, must also be considered in the answer. As the answer can be very large, print it modulo 1000000007 $(10^9 + 7)$.

Input

The first line contains positive integer n $(1 \le n \le 500)$. The second line contains space-separated integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^9)$.

Output

In a single line print a single number k — the number of distinct decompositions of number m into n ordered multipliers modulo 1000000007 $(10^9 + 7)$.

Examples

input
```
1
15
```
output
```
1
```

input
```
3
1 1 2
```
output
```
3
```

input
```
2
5 7
```
output
```
4
```

Note

In the second sample, the get a decomposition of number 2, you need any one number out of three to equal 2, and the rest to equal 1.

In the third sample, the possible ways of decomposing into ordered multipliers are [7,5], [5,7], [1,35], [35,1].

A decomposition of positive integer m into n ordered multipliers is a cortege of positive integers $b = \{b_1, b_2, \dots b_n\}$ such that $m = \prod_{i=1}^{n} b_i$. Two decompositions b and c are considered different, if there exists index i such that $b_i \ne c_i$.

F – Counting heaps

We are given a rooted tree of n vertices. The vertices are to be labeled with numbers $1, 2, \ldots, n$ so that each label is unique and the heap condition holds, i.e. the label of any vertex is less than the label of its parent. How many such labellings exist? Since this number may be quite large, calculate only its remainder modulo m.

Input

The input contains several tree descriptions. The first line contains the number of input trees t ($t \leq 250$). Each tree description begins with a line containing the size of the tree n ($1 \leq n \leq 500000$) and an integer m ($2 \leq m \leq 10^9$). $n-1$ lines follow, i-th of which contains $p(i+1)$, the number of the parent of the $i+1$-th vertex ($1 \leq p(i+1) \leq i$). Vertex number 1 will be the root in each tree, so its parent will not be given. Total size of the input will not exceed 50MB.

Output

For each tree output the number of its valid labellings modulo given m.

Example

Input	Output

4
3 1000000
1
1
4 1000000
1
1
1
5 1000000
1
2
3
4
5 1000000
1
1
3
3
	2
	6
	1
	8

The 8 possible labellings from the last example test case are as follows: