A. Nuts

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

You have a nuts and lots of boxes. The boxes have a wonderful feature: if you put x $(x \geq 0)$ divisors (the spacial bars that can divide a box) to it, you get a box, divided into $x + 1$ sections.

You are minimalist. Therefore, on the one hand, you are against dividing some box into more than k sections. On the other hand, you are against putting more than v nuts into some section of the box. What is the minimum number of boxes you have to use if you want to put all the nuts in boxes, and you have b divisors?

Please note that you need to minimize the number of used boxes, not sections. You do not have to minimize the number of used divisors.

Input

The first line contains four space-separated integers k, a, b, v ($2 \leq k \leq 1000$; $1 \leq a, b, v \leq 1000$) — the maximum number of sections in the box, the number of nuts, the number of divisors and the capacity of each section of the box.

Output

Print a single integer — the answer to the problem.

Examples

input
```
3 10 3 3
```
output
```
2
```

input
```
3 10 1 3
```
output
```
3
```

input
```
100 100 1 1000
```
output
```
1
```

Note

In the first sample you can act like this:

- Put two divisors to the first box. Now the first box has three sections and we can put three nuts into each section. Overall, the first box will have nine nuts.
- Do not put any divisors into the second box. Thus, the second box has one section for the last nut.

In the end we've put all the ten nuts into boxes.

The second sample is different as we have exactly one divisor and we put it to the first box. The next two boxes will have one section each.

B. Trees in a Row

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

The Queen of England has n trees growing in a row in her garden. At that, the i-th $(1 \le i \le n)$ tree from the left has height a_i meters. Today the Queen decided to update the scenery of her garden. She wants the trees' heights to meet the condition: for all i $(1 \le i < n)$, $a_{i+1} - a_i = k$, where k is the number the Queen chose.

Unfortunately, the royal gardener is not a machine and he cannot fulfill the desire of the Queen instantly! In one minute, the gardener can either decrease the height of a tree to any positive integer height or increase the height of a tree to any positive integer height. How should the royal gardener act to fulfill a whim of Her Majesty in the minimum number of minutes?

Input

The first line contains two space-separated integers: n, k $(1 \le n, k \le 1000)$. The second line contains n space-separated integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 1000)$ — the heights of the trees in the row.

Output

In the first line print a single integer p — the minimum number of minutes the gardener needs. In the next p lines print the description of his actions.

If the gardener needs to increase the height of the j-th $(1 \le j \le n)$ tree from the left by x $(x \ge 1)$ meters, then print in the corresponding line "`+ j x`". If the gardener needs to decrease the height of the j-th $(1 \le j \le n)$ tree from the left by x $(x \ge 1)$ meters, print on the corresponding line "`- j x`".

If there are multiple ways to make a row of trees beautiful in the minimum number of actions, you are allowed to print any of them.

Examples

input

```
4 1
1 2 1 5
```

output

```
2
+ 3 2
- 4 1
```

input

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

output

```
0
```

C. Searching for Graph

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

Let's call an undirected graph of n vertices p-*interesting*, if the following conditions fulfill:

- the graph contains exactly $2n + p$ edges;
- the graph doesn't contain self-loops and multiple edges;
- for any integer k ($1 \le k \le n$), any subgraph consisting of k vertices contains at most $2k + p$ edges.

A *subgraph* of a graph is some set of the graph vertices and some set of the graph edges. At that, the set of edges must meet the condition: both ends of each edge from the set must belong to the chosen set of vertices.

Your task is to find a p-*interesting* graph consisting of n vertices.

Input

The first line contains a single integer t ($1 \le t \le 5$) — the number of tests in the input. Next t lines each contains two space-separated integers: n, p ($5 \le n \le 24$; $p \ge 0$; $2n + p \le \frac{n(n-1)}{2}$) — the number of vertices in the graph and the interest value for the appropriate test.

It is guaranteed that the required graph exists.

Output

For each of the t tests print $2n + p$ lines containing the description of the edges of a p-*interesting* graph: the i-th line must contain two space-separated integers a_i, b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$) — two vertices, connected by an edge in the resulting graph. Consider the graph vertices numbered with integers from 1 to n.

Print the answers to the tests in the order the tests occur in the input. If there are multiple solutions, you can print any of them.

Examples

input

```
1
6 0
```

output

```
1 2
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
```

D. Upgrading Array

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

You have an array of positive integers $a[1], a[2], \ldots, a[n]$ and a set of *bad* prime numbers $b_1, b_2, \ldots, b_m$. The prime numbers that do not occur in the set b are considered *good*. The *beauty* of array a is the sum $\sum_{i=1}^{n} f(a[i])$, where function $f(s)$ is determined as follows:

- $f(1) = 0$;
- Let's assume that p is the minimum prime divisor of s. If p is a good prime, then $f(s) = f(\frac{s}{p}) + 1$, otherwise $f(s) = f(\frac{s}{p}) - 1$.

You are allowed to perform an arbitrary (probably zero) number of operations to improve array a. The *operation of improvement* is the following sequence of actions:

- Choose some number r ($1 \le r \le n$) and calculate the value g = GCD($a[1], a[2], \ldots, a[r]$).
- Apply the assignments: $a[1] = \frac{a[1]}{g}$, $a[2] = \frac{a[2]}{g}$, $\ldots$, $a[r] = \frac{a[r]}{g}$.

What is the maximum beauty of the array you can get?

Input

The first line contains two integers n and m ($1 \le n, m \le 5000$) showing how many numbers are in the array and how many bad prime numbers there are.

The second line contains n space-separated integers $a[1], a[2], \ldots, a[n]$ ($1 \le a[i] \le 10^9$) — array a. The third line contains m space-separated integers $b_1, b_2, \ldots, b_m$ ($2 \le b_1 < b_2 < \ldots < b_m \le 10^9$) — the set of bad prime numbers.

Output

Print a single integer — the answer to the problem.

Examples

input

```
5 2
4 20 34 10 10
2 5
```

output

```
-2
```

input

```
4 5
2 4 8 16
3 5 7 11 17
```

output

```
10
```

Note

Note that the answer to the problem can be negative.

The GCD(x_1, x_2, ..., x_k) is the maximum positive integer that divides each x_i.

E. Strictly Positive Matrix

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

You have matrix a of size $n \times n$. Let's number the rows of the matrix from 1 to n from top to bottom, let's number the columns from 1 to n from left to right. Let's use a_{ij} to represent the element on the intersection of the i-th row and the j-th column.

Matrix a meets the following two conditions:

- for any numbers i, j ($1 \le i, j \le n$) the following inequality holds: $a_{ij} \ge 0$;
- $\sum_{i=1}^{n} a_{ii} > 0$.

Matrix b is *strictly positive*, if for any numbers i, j ($1 \le i, j \le n$) the inequality $b_{ij} > 0$ holds. You task is to determine if there is such integer $k \ge 1$, that matrix a^k is strictly positive.

Input

The first line contains integer n ($2 \le n \le 2000$) — the number of rows and columns in matrix a.

The next n lines contain the description of the rows of matrix a. The i-th line contains n non-negative integers $a_{i1}, a_{i2}, \dots, a_{in}$ ($0 \le a_{ij} \le 50$). It is guaranteed that $\sum_{i=1}^{n} a_{ii} > 0$.

Output

If there is a positive integer $k \ge 1$, such that matrix a^k is strictly positive, print "YES" (without the quotes). Otherwise, print "NO" (without the quotes).

Examples

input

```
2
1 0
0 1
```

output

```
NO
```

input

```
5
4 5 6 1 2
1 2 3 4 5
6 4 1 2 4
1 1 1 1 1
4 4 4 4 4
```

output

```
YES
```