A. 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.

Sample test(s)

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
```

B. 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 $a[1], a[2], ..., a[n]$ and a set of *bad* prime numbers $b_1, b_2, ..., b_m$. 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($a[1], a[2], ..., a[r]$).
- Apply the assignments: $a[1] = \frac{a[1]}{g}$, $a[2] = \frac{a[2]}{g}$, ..., $a[r] = \frac{a[r]}{g}$.

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

Output

Print a single integer — the answer to the problem.

Sample test(s)

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.

C. 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 $a_{i1}, a_{i2}, ..., a_{in}$ $(0 \le a_{ij} \le 50)$. 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).

Sample test(s)

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
```

D. Beautiful Pairs of Numbers

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

The sequence of integer pairs $(a_1, b_1), (a_2, b_2), ..., (a_k, b_k)$ is *beautiful*, if the following statements are fulfilled:

- $1 \le a_1 \le b_1 < a_2 \le b_2 < ... < a_k \le b_k \le n$, where n is a given positive integer;
- all numbers b_1 - a_1, b_2 - a_2, ..., b_k - a_k are distinct.

For the given number n find the number of beautiful sequences of length k. As the answer can be rather large, print the remainder after dividing it by 1000000007 $(10^9 + 7)$.

Input

The first line contains integer t $(1 \le t \le 2 \cdot 10^5)$ — the number of the test data.

Each of the next t lines contains two integers n and k $(1 \le k \le n \le 1000)$.

Output

For each test from the input print the answer to the problem modulo 1000000007 $(10^9 + 7)$. Print the answers to the tests in the order in which the tests are given in the input.

Sample test(s)

input

```
6
1 1
2 1
2 2
3 1
3 2
3 3
```

output

```
1
3
0
6
2
0
```

Note

In the first test sample there is exactly one beautiful sequence: $(1, 1)$.

In the second test sample, the following sequences are beautiful:

- $(1, 1)$;
- $(1, 2)$;
- $(2, 2)$.

In the fourth test sample, the following sequences are beautiful:

- $(1, 1)$;
- $(1, 2)$;
- $(1, 3)$;
- $(2, 2)$;
- $(2, 3)$;
- $(3, 3)$.

In the fifth test sample, the following sequences are beautiful:

- $(1, 1), (2, 3)$;
- $(1, 2), (3, 3)$.

In the third and sixth samples, there are no beautiful sequences.

E. Two Rooted Trees

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You have two rooted undirected trees, each contains n vertices. Let's number the vertices of each tree with integers from 1 to n. The root of each tree is at vertex 1. The edges of the first tree are painted blue, the edges of the second one are painted red. For simplicity, let's say that the first tree is blue and the second tree is red.

Edge $\{x, y\}$ is called bad for edge $\{p, q\}$ if two conditions are fulfilled:

1. The color of edge $\{x, y\}$ is different from the color of edge $\{p, q\}$.
2. Let's consider the tree of the same color that edge $\{p, q\}$ is. Exactly one of vertices x, y lies both in the subtree of vertex p and in the subtree of vertex q.

In this problem, your task is to simulate the process described below. The process consists of several stages:

1. On each stage edges of exactly one color are deleted.
2. On the first stage, exactly one blue edge is deleted.
3. Let's assume that at the stage i we've deleted edges $\{u_1, v_1\}$, $\{u_2, v_2\}$, ..., $\{u_k, v_k\}$. At the stage $i+1$ we will delete all undeleted bad edges for edge $\{u_1, v_1\}$, then we will delete all undeleted bad edges for edge $\{u_2, v_2\}$ and so on until we reach edge $\{u_k, v_k\}$.

For each stage of deleting edges determine what edges will be removed on the stage. Note that the definition of a bad edge always considers the initial tree before it had any edges removed.

Input

The first line contains integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in each tree.

The next line contains n - 1 positive integers $a_2, a_3, ..., a_n$ ($1 \le a_i \le n$; $a_i \ne i$) — the description of edges of the first tree. Number a_i means that the first tree has an edge connecting vertex a_i and vertex i.

The next line contains n - 1 positive integers $b_2, b_3, ..., b_n$ ($1 \le b_i \le n$; $b_i \ne i$) — the description of the edges of the second tree. Number b_i means that the second tree has an edge connecting vertex b_i and vertex i.

The next line contains integer idx ($1 \le idx < n$) — the index of the blue edge that was removed on the first stage. Assume that the edges of each tree are numbered with numbers from 1 to n - 1 in the order in which they are given in the input.

Output

For each stage of removing edges print its description. Each description must consist of exactly two lines. If this is the stage when blue edges are deleted, then the first line of the description must contain word $Blue$, otherwise — word Red. In the second line print the indexes of the edges that will be deleted on this stage in the increasing order.

Sample test(s)

input

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

output

```
Blue
3
Red
1 3
Blue
1 2
Red
2
```

Note

For simplicity let's assume that all edges of the root tree received some direction, so that all vertices are reachable from vertex 1. Then a *subtree* of vertex v is a set of vertices reachable from vertex v in the resulting directed graph (vertex v is also included in the set).