Security

There are N objects numbered $0, 1, \ldots, N-1$ and M security gates numbered $0, 1, \ldots, M-1$. Each security gate i has a secret subset of objects $A[i]$.

You may send a package P consisting of a non-empty subset of the N objects. The package passes through the security gates in increasing order. Security gate i confiscates the package if $|A[i] \cap P|$ is odd. A package stops at the first security gate that confiscates it. After sending a package, you are told the index of this security gate.

It is guaranteed that every non-empty package is confiscated by some security gate.

A security gate is called **useful** if there exists a package for which it is the first security gate that confiscates it. It can be proven that there are exactly N useful security gates. Let their indices, in increasing order, be

$$c[0] < c[1] < \ldots < c[N-1].$$

Your goal is to send packages so that, for every $0 \leq i < N$, security gate $c[i]$ confiscates exactly $N - i$ of the packages you send.

You may choose each package based on the answers obtained for previously sent packages.

Implementation details

You should implement the following procedure.

```
void solve(int N, int M)
```

- N: the number of objects.
- M: the number of security gates.
- This procedure is called exactly once for each test case.

You may call the following procedure:

```
int send_package(std::vector<int> P)
```

- P must contain the indices of a non-empty subset of the objects.
- The procedure returns the index of the first security gate that confiscates P.

- Each object may appear at most once in P.

The behavior of the grader is **not adaptive**. This means that the sets $A[0], A[1], \ldots, A[M-1]$ are fixed before `solve` is called.

Example

Suppose $N = 3$ and $M = 4$, and the secret subsets are:

- $A[0] = \{0, 1\}$
- $A[1] = \{0, 1\}$
- $A[2] = \{1, 2\}$
- $A[3] = \{2\}$

In this case, the useful security gates are 0, 2, and 3. Gate 1 is not useful because any package it would confiscate has already been confiscated by gate 0.

One possible sequence of calls to `send_package` is:

Package P	Returned gate
$\{0\}$	0
$\{1\}$	0
$\{0, 2\}$	0
$\{0, 1\}$	2
$\{2\}$	2
$\{0, 1, 2\}$	3

For example, the package $\{0, 1, 2\}$ passes through gates 0, 1, and 2, since its intersection with each of their subsets has even size. It is then confiscated by gate 3.

At the end, gate 0 has confiscated 3 packages, gate 2 has confiscated 2 packages, and gate 3 has confiscated 1 package. Therefore, the goal has been achieved.

Note: The secret subsets are shown only for illustration. They are not provided to your program.

Constraints and Scoring

- $1 \leq N \leq 300$
- $N \leq M \leq 2000$
- Every non-empty package is confiscated by some security gate.
- There are exactly N useful security gates.

Subtask	Score	Additional Constraints
1	7	$A[i] = \{i\}$ for every $0 \le i < M$, and $M = N$.
2	9	$\lvert A[i] \rvert = 1$ for every $0 \le i < M$.
3	6	$\lvert A[i] \rvert \le 2$ for every $0 \le i < M$, and $\bigcap_{i=0}^{M-1} A[i] \neq \emptyset$.
4	14	$\lvert A[i] \rvert \le 2$ for every $0 \le i < M$.
5	31	$A[i] = \{L[i], L[i]+1, \ldots, R[i]\}$ for every $0 \le i < M$.
6	33	No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

- line 1: N M
- line $2+i$ ($0 \le i < M$): K_i $A[i][0]$ $A[i][1]$ $\ldots$ $A[i][K_i-1]$

Here, K_i is the number of objects in the set $A[i]$.

The sample grader calls `solve(N, M)` and simulates the function `send_package`.

After `solve` terminates, the sample grader prints one of the following messages:

- `Correct` if all useful gates have been found and each useful gate $c[i]$ has confiscated exactly $N-i$ packages.
- `You did not find all useful gates` if fewer than N gates have confiscated at least one package.
- `Wrong` if all useful gates have been found, but the required number of packages was not confiscated by each of them.
- `Invalid package` if `send_package` is called with an invalid package. In this case, the sample grader terminates immediately.