Tournament

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

This is an interactive problem.

A tournament graph on n vertices numbered from 1 to n is hidden. A tournament graph is a directed graph where every edge connects two distinct vertices, and for any pair of distinct vertices u and v, there is exactly one edge oriented either from u to v or from v to u.

You only know the number of vertices in the graph, and in one query, you can find out the direction of the edge between any pair of vertices.

A vertex is called *good* if there is at most one outgoing edge from it. Your task is to find any *good* vertex or determine that there are no such vertices, using no more than 2000 queries.

Input

The first line contains two integers g and t ($0 \le g \le 10$, $1 \le t \le 100$) — the test group and the number of test cases. This is followed by a description of the test cases.

The first line of each test case contains a single integer n ($1 \le n \le 500$) — the number of vertices of the hidden graph.

Interaction Protocol

Your program will interact with the jury program using standard input and output streams. Each interaction will consist of solving the problem for multiple sets of input data.

First, your program should read two integers g and t ($1 \le t \le 100$) — the test group and the number of test cases within one interaction with the jury program. Then, t times, you need to perform the interaction to solve the problem for the input data set.

Let's consider the interaction protocol for one input data set.

First, your program should read one integer n ($1 \le n \le 500$) — the number of vertices in the hidden graph.

After that, you can make queries. For one query, output «`? u v`» ($1 \le u, v \le n$, $u \ne v$). In response to this, the jury program will print «`forward`» in a single line if the edge is directed from u to v, and «`backward`» if the edge is directed from v to u. For each test case, you can make at most 2000 queries.

To output the answer, print «`! u`» ($1 \le u \le n$ or $u = -1$), where u is a *good* vertex, or -1 if there is no such vertex. If the output answer is correct, the jury program will print «`OK`». After that, your program should immediately move on to process the next set of input data or terminate if it was the last graph. Otherwise, the jury program will print «`WRONG`». When reading this line, your program should immediately terminate its execution.

Please note that the constraints on n are given for each test case, there are no additional constraints on the sum of n for all sets of input data.

If you use «`cout << ... << endl`» in C++, «`System.out.println`» in Java, «`print`» in Python, «`writeln`» in Pascal, the output stream will be flushed automatically, so no additional action is required.

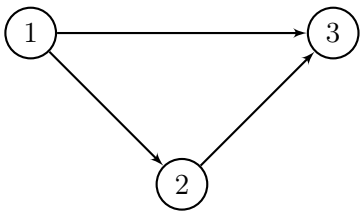
If you use a different method of output, it is recommended to flush the output stream. Please note that a new line should be output in any case. To flush the output stream, you can use «`fflush(stdout)`» in C++, «`flush(output)`» in Pascal, «`System.out.flush()`» in Java, «`sys.stdout.flush()`» in Python.

Example

standard input	standard output
0 2 3 forward backward OK 5 forward forward backward backward forward forward backward forward forward forward OK	 ? 1 2 ? 3 2 ! 3 ? 1 2 ? 1 3 ? 1 4 ? 1 5 ? 2 3 ? 2 4 ? 2 5 ? 3 4 ? 3 5 ? 4 5 ! -1

Note

In the first test case, since the jury program is not adaptive, the graph is predetermined and has the following form:



Therefore, in response to the query «? 1 2», the jury program outputted «forward», as the edge is directed from 1 to 2, and in response to the query «? 3 2», the jury program outputted «backward», as the edge is directed from 2 to 3. In this case, vertices 2 and 3 are *good*, while vertex 1 is not. Therefore, in response to the query «! 3», the jury program printed «OK».

In the second test case, there are multiple outgoing edges from each vertex, so in response to the query «! −1», the jury program printed «OK».

Scoring

The tests for this problem consist of 10 groups. Points for each group are awarded only if all the tests in that group and some tests from the previous groups pass. **Offline-testing** means that the results of testing your solution on this group will only be available after the competition ends.

In this problem, the jury program can act in two ways:

- Non-adaptive. That is, the structure of the graph is fixed in advance and does not adjust to the requests.
- Adaptive. That is, the structure of the graph can change during the interaction. However, it is guaranteed that there always exists at least one graph that fits the answers to all the already asked queries. In this case, the answer of your solution will be considered correct only if it is correct for all suitable graphs.

Group	Score	Additional constaints		Required groups	Comment
		n	Jury program		
0	0	–	Non-adaptive	–	Samples.
1	14	$n \leq 63$	Adaptive	0	
2	8	–	Adaptive	–	Graph is *special*[1]
3	11	–	Adaptive	–	Graph is *special*[2]
4	12	–	Adaptive	–	Graph is *special*[3]
5	9	$n \leq 100$	Adaptive	0, 1	
6	10	$n \leq 400$	Non-adaptive	0	
7	9	$n \leq 400$	Adaptive	0, 1, 5, 6	
8	8	$n \leq 450$	Non-adaptive	0, 6	
9	10	–	Non-adaptive	0, 6, 8	
10	9	–	Adaptive	0 – 9	**Offline-testing.**

[1] A graph is called *special*[1] if there exists a permutation of vertices v_1, v_2, …, v_n such that for any $i < j$, the edge is oriented from vertex v_i to vertex v_j.

[2] A graph is called *special*[2] if for any vertex with a number $v > 2$, the edge is oriented from it to vertex 1 or to vertex 2, or to both vertices 1 and 2.

[3] A graph is called *special*[3] if $n \geq 4$ and there exists a permutation of vertices v_1, v_2, …, v_n such that the edge (v_i, v_{i+1}) is oriented from v_i to v_{i+1} and any other edge (v_i, v_j) $(i + 1 < j)$ is oriented from v_j to v_i.