Lucid Traveler

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

Having awakened, *Ivy* is exploring the road network of Node 08.

Node 08 can be modeled as an n-vertex **edge cactus**, and *Ivy* needs to find every edge in the graph. Fortunately, she may ask the Eye of *Horus* some queries. In each query, *Ivy* chooses two vertices x and y and a set of vertices S containing neither x nor y. The Eye of *Horus* tells her whether x and y are connected after all vertices in S, together with their incident edges, are deleted from the graph. However, the Eye of *Horus* has limited computational power, so such a query incurs a cost of $\max(|S| - 1, 0)$.

Time is limited. *Ivy* may make at most $3n^2$ queries, and the sum of the costs of all queries must not exceed $n\lceil \log_2 n \rceil + 2n$. You need to help *Ivy* devise a strategy that is guaranteed to determine every edge in the graph.

Recall that an edge cactus is an undirected simple connected graph in which every edge belongs to at most one simple cycle. A simple graph is a graph with neither multiple edges nor self-loops.

Note that the interactor is **adaptive**, meaning that the graph for each test case is not pre-determined. The interactor always guarantees that there is at least one graph consistent with all previous replies, but it may choose its reply based on your query. The interactor determines the final graph only when you submit your final answer.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 128$) indicating the number of test cases. For each test case:

The first line contains an integer n ($2 \le n \le 128$) indicating the number of vertices in the graph.

It is guaranteed that the sum of n over all test cases does not exceed 256.

The query-count and query-cost limits are evaluated **separately** for each test case.

Interaction Protocol

To ask a query, output one line. First output `?` followed by a space, then print three integers x, y, and k, followed by k integers $s_1, s_2, \ldots, s_k$, all separated by spaces. Here, $k = |S|$, $1 \le x, y \le n$, $x \ne y$, $0 \le k \le n - 2$, and $s_1, s_2, \ldots, s_k$ must be pairwise distinct and different from both x and y.

After flushing your output, your program should read a single integer t indicating the answer to your query. If x and y are still connected after deleting S, then $t = 1$; otherwise, $t = 0$. Recall that you can ask at most $3n^2$ queries for a test case, and the sum of the costs of all queries must not exceed $n\lceil \log_2 n \rceil + 2n$.

If you want to report all edges in the graph, first output one line. First output `!` followed by a space, then print an integer m indicating the number of edges. Then output m lines, where the i-th line contains two integers u_i and v_i indicating an edge. The edges may be listed in any order, and the endpoints of each edge may be listed in either order, but the same edge must not be printed more than once.

After flushing your output, your program should continue processing the next test case, or exit immediately if there are no more test cases. Note that your final answer does not count as a query and incurs no query cost.

To flush your output, you can use:

- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java and Kotlin.

- `sys.stdout.flush()` in Python.

Example

standard input	standard output
2 3 1 6 0 1 0	 ? 1 2 1 3 ! 3 1 2 1 3 2 3 ? 2 6 2 4 5 ? 1 4 0 ? 1 4 2 2 5 ! 7 1 2 4 5 1 5 2 4 4 3 4 6 3 6