# Travel 2

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 3 seconds |
| Memory limit: | 1024 megabytes |

*This is an interactive problem.*

Little Cyan Fish, a zealous traveler, is on a mission to circumnavigate the world of fish consisting of $n$ cities, numbered from 1 to $n$. Starting at city 1, he wishes to map out all $m$ bidirectional roads connecting these cities. It is guaranteed that it is possible to travel to any city by using the roads. Furthermore, each road connects two distinct cities, and no two cities have more than one road between them. In other words, the cities and roads can be treated as a simple and connected undirected graph. However, without a map, Little Cyan Fish is unaware of the roads' existence, so he has absolutely no idea which roads exist in this world. Therefore, his goal is to find the information on all the roads.

To accomplish this, Little Cyan Fish can walk in this world. When he stands in a city, Little Cyan Fish can see the index $x$ of the current city and the number of roads connected to the city $x$, $d_x$. He can then choose to walk through the $i$-th road ($1 \le i \le d_x$) from city $x$. The order of roads from each city is unknown but fixed. After the walking, Little Cyan Fish will be standing at the other endpoint of the road he choose, and report to you the information of the new city, i.e., its index and the number of roads connected to it.

Your goal is to help Little Cyan Fish find a strategy to find all the roads by repeating the process above. Since Little Cyan Fish only has a limited time, and the world travel would consume a large amount of time, he can only walk no more than $(2m + 2n)$ times. Please help him!

## Interaction Protocol

There are multiple test cases in a single test file. The first line of the input contains a single integer $T$ ($1 \le T \le 1\,000$), indicating the number of the test cases.

For each test case, you will not know the exact value of $n$ and $m$ — the interaction process begins immediately. It is guaranteed that $1 \le n \le 2\,500$ and $1 \le m \le 10^4$.

At each round of the interaction, the interactor will provide the index $x$, indicating the city that Little Cyan Fish stands at, and $d_x$, indicating the number of roads connecting the city $x$. To make a walk, you need to print "`>` $i$" on a separate line ($1 \le i \le d_x$), indicating that you would like to walk through the $i$-th road connecting the city $x$. After that, the new round of the interaction begins, and the interactor will provide you with the new index and the number of the roads connecting to that city. The number of the queries you made should be no more than $2m + 2n$.

To give your answer, you need to print "`!` $x_1\ y_1\ x_2\ y_2\ \cdots\ x_m\ y_m$", indicating all the roads you found. The order of the edges can be arbitrary, you only need to provide one of the possible plans. Printing the answer is not considered a query and does not count toward the $2m + 2n$ limit.

After you propose your answer, if you find all the $m$ roads correctly, the interactor will print a single line `Correct` — then, the next test case will be followed immediately. Otherwise, if the $m$ roads you found are incorrect, the interactor will print a single line `Wrong`, and the interaction process will be terminated. If you find less than $m$ roads, the interactor will wait for you to print more edges. This may result in "Idleness Limit Exceeded".

After printing a query, do **NOT** forget to output end of line and flush the output. To do this, use `fflush(stdout)` or `cout.flush()` in C++, `System.out.flush()` in Java, `flush(output)` in Pascal, or `stdout.flush()` in Python.

It is guaranteed that the hidden graph is simple and connected, the sum of $n$ over all test cases will not exceed $10^4$, and the sum of $m$ over all test cases will not exceed $4 \times 10^4$.

The interactor of the problem is **non-adaptive**, which means that the graph will be fixed in advance, and it will not be changed based on the interaction.

## Example

| standard input | standard output |
|---|---|
| 2 | |
| 1 1 | |
| | > 1 |
| 2 1 | |
| | > 1 |
| 1 1 | |
| | ! 1 2 |
| Correct | |
| 1 3 | |
| | > 3 |
| 2 2 | |
| | > 1 |
| 1 3 | |
| | > 3 |
| 2 2 | |
| | > 2 |
| 4 2 | |
| | > 2 |
| 1 3 | |
| | > 2 |
| 3 1 | |
| | ! 1 2 1 3 1 4 2 4 |
| Correct | |