Radio Direction Finding

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

This is an interactive problem.

Radio direction finding, also known as radio orienteering or radio fox hunting, is a sport that combines radio technology with outdoor navigation. Participants use specialized receivers to locate hidden radio transmitters, testing their sense of direction and radio operation skills.

A university offers a PE course in radio direction finding. During the final exam, the teacher takes the students to a local tourist attraction. The map of the attraction can be considered as a cycle of n points (n is an odd number), numbered clockwise as $1, 2, \ldots, n$. The teacher has previously buried two transmitters at two different points and given you a radio receiver.

This receiver is special; each time it receives a signal, you need to manually press a button on the receiver, and then the receiver will tell you the sum of the shortest distances from these two transmitters to your current location (distance is defined as the number of edges traversed). The passing criterion for the final exam is to find the positions of these two transmitters using the receiver no more than 40 times.

Now, please write an interactive program to successfully pass the final exam.

Interaction Protocol

First, read a positive integer T ($1 \leq T \leq 10^3$) from the standard input, indicating the number of test cases.

For each test case, first read a positive integer n ($3 \leq n \leq 10^9$, and n is odd) from the standard input, indicating the number of points in the cycle.

Then, start the interaction process. Each time you interact, you can output the following two types of data to the standard output:

- `? x`: Indicates that you are using the receiver at point x, and then your program needs to read an integer *dist* from the standard input, indicating the sum of the shortest distances from the two transmitters to point x. You need to ensure that $1 \leq x \leq n$.
  - For each set of data, the `? x` operation should not exceed 40 times. If you inquire more than the specified number of times, the interactor will immediately end and you will receive a "Wrong Answer" verdict.
- `! x y`: Indicates your answer, i.e., the points where the two transmitters are located (order does not matter) are at x and y. In this case:
  - If your output format is correct and the answer is correct, the interactor will immediately move on to the next test case;
  - Otherwise, if your output format is incorrect or the answer is incorrect, the interactor will immediately end and you will receive a "Wrong Answer" verdict.

Note: Each output needs to include a line break and flush the buffer, otherwise you may get unexpected results other than the correct answer. To flush the buffer, you can:

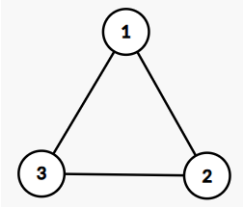
- For C or C++, use `fflush(stdout)` or `cout.flush()`.
- For Java, use `System.out.flush()`.
- For Python, use `stdout.flush()`.

Example

standard input	standard output
2 3 1 1 2 5 4 3	 ? 1 ? 2 ? 3 ! 1 2 ? 5 ? 1 ! 2 3

Note

The first sample is as shown in the following figure, with hidden points at 1 and 2:



- Inquire `? 1`, the corresponding sum of shortest distances is $0 + 1 = 1$;
- Inquire `? 2`, the corresponding sum of shortest distances is $1 + 0 = 1$;
- Inquire `? 3`, the corresponding sum of shortest distances is $1 + 1 = 2$;
- Finally, output `! 1 2`, the answer is correct.