Restore the Array

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

Little L has a hidden set of size n, $\{a_1, a_2, \ldots, a_n\}$ ($0 \le a_i < 2^{30}$), with all elements distinct.

To guess the values of the elements in the set, you can make queries to Little L. In a query, you give a non-negative integer c, and Little L will tell you $\max\{a_i \oplus c\}$, where $\oplus$ denotes bitwise XOR [†].

For example, if $a = \{1, 4, 6\}$ and $c = 3$, Little L will respond 7, because $\max\{1 \oplus 3, 4 \oplus 3, 6 \oplus 3\} = \max\{2, 7, 5\} = 7$.

You need to guess all elements of the set using no more than $2n - 2$ queries.

[†] Bitwise XOR is defined as adding two binary numbers bit by bit modulo 2. For example: $(0011)_2 \oplus (0101)_2 = (0110)_2$.

Input

Each test file contains multiple test cases. The first line contains the number of test cases T ($1 \le T \le 10^4$). The description of the test cases follows.

Each test case contains a single line with a positive integer n ($2 \le n \le 10^5$), representing the size of the set.

For each test file, it is guaranteed that the sum of n over all test cases does not exceed 10^5.

Interaction Protocol

To make a query, output in the format "`?` c" ($0 \le c < 2^{30}$). After flushing the output, your program should read a single integer from input, which is the value of $\max\{a_i \oplus c\}$. For each test case, you can make at most $2n - 2$ queries. If you exceed this number of queries, the verdict will be **undefined**.

To guess the set, output in the format "`!` a_1 a_2 $\ldots$ a_n" ($0 \le a_i < 2^{30}$). The order of elements can be arbitrary. After flushing the output, the interaction for the current test case ends. You should immediately start handling the next test case, or exit if all test cases have been processed.

Note that the interactor is **adaptive**. This means that the elements in the set are not fixed at the beginning but may be dynamically adjusted based on your queries. The interactor guarantees that adjustments will not invalidate previous responses.

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

Example

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