Find the Permutation (findperm)

✍ This is an interactive problem. Your program communicates with the evaluator: it should alternately write messages to the evaluator on the standard output and read the next input from the standard input.

There is a hidden permutation $P = [P_1, P_2, \dots, P_N]$ of the numbers $1, 2, \dots, N$.[1]

Figure 1: The permutation, hidden between two walls.

You can ask questions in the form (i, j) with $1 \le i, j \le N$. Note that i and j does not have to be distinct.

The evaluator responds with the **most significant bit** of the bitwise AND of P_i and P_j.[2] Here, the most significant bit of a number x is the exponent of the highest power of 2 that appears when writing x in base 2. For example, the most significant bit of $5 = 101_{(2)}$ is 2, since 2^2 appears in the binary representation of 5. If the bitwise AND of P_i and P_j is 0, the evaluator responds with -1.

Your task is to determine the hidden permutation of at most 1000 numbers by asking no more than 30 000 questions.

☞ Among the attachments of this task you may find a template file `findperm.*` with a sample incomplete implementation.

Input

The first line of the input contains the number N, the length of the permutation. After reading N, you can ask questions by printing them to the standard output in the following format:

```
? i j
```

where i and j describe the question (i, j).

[1] A permutation of the numbers $1, 2, \dots, N$ is a sequence of length N consisting of the numbers $1, 2, \dots, N$, containing each number exactly once.

[2] The bitwise AND of two numbers x and y is the number that contains the common bits of x and y. For example, if $x = 11 = 1011_{(2)}$ and $y = 13 = 1101_{(2)}$, then their bitwise AND will be $9 = 1001_{(2)}$.

For each question, if i and j are between 1 and N (inclusive) and the solution has not asked more than 30 000 questions, you receive the answer. Otherwise, you receive -100 and the evaluator terminates the interaction immediately with an 'Output isn't correct' verdict.

Once found, you have to print the hidden permutation in the following format:

`!` P_1 P_2 $\ldots$ P_N

Printing the hidden permutation is not counted as a question.

After guessing the hidden permutation, or receiving the -100 answer from the evaluator, your program shall terminate immediately, otherwise you might receive a 'Time limit exceeded' verdict.

Constraints

- $1 \leq N \leq 1000$.

Scoring

Your program will be tested against several test cases. The test cases are being scored **independently**, and your final score is the sum of the scores of the test cases. The evaluator is **not adaptive**, meaning that in each test case, the hidden permutation is fixed before your program asks any questions.

Let Q denote the number of questions asked by your program in a test case. Then, your score for this test case is calculated in the following way:

- If $Q > 30\,000$ you will get 0% of the points for the test;
- If $25\,000 < Q \leq 30\,000$ you will get 20% of the points for the test;
- If $20\,000 < Q \leq 25\,000$ you will get 40% of the points for the test;
- If $15\,000 < Q \leq 20\,000$ you will get 60% of the points for the test;
- If $Q \leq 15\,000$ you will get 100% of the points for the test.

Examples

input	output
5	
2	? 1 5
1	? 2 3
-1	? 4 3
0	? 4 4
	! 4 3 2 1 5

Explanation

Suppose that the hidden permutation is $P = [4, 3, 2, 1, 5]$. The first question asks the most significant bit of the numbers $P_1 = 4$ and $P_5 = 5$, which is 2.

The second one asks about the hidden numbers 3 and 2, for which the answer is 1.

The bitwise `AND` of numbers 2 and 1 is 0 and thus, the answer for the third question will be -1.

Lastly, the answer to the fourth question is 0, because the bitwise AND of 1 and 1 is 1.

Note that this is just a sample interaction to showcase the process of asking questions, receiving answers, and guessing the hidden permutation. It **does not** necessarily represent an approach to finding the hidden permutation.