Interactive

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Aidos has come up with a puzzle and challenged Temirulan to solve it. He picked a sequence a of n non-negative integers numbered from 1 to n: $a_1, a_2, ..., a_n$.

Temirulan can ask two types of questions:

- *ask* — Reveal the number at position i of the given sequence.
- *get_pairwise_xor* — For the given sequence of distinct integer numbers: $i_1, i_2, ..., i_k$ get a set of pairwise values of *xor* for the elements of the sequence a at indexes $i_1, i_2, ..., i_k$, $\{a_{i_x} \oplus a_{i_y} \mid 1 \leq x, y \leq k\}$.

For example, let's assume that Aidos has picked the sequence $[1, 5, 6, 3]$. Then for the question $ask(2)$, Aidos will answer with the number 5 and for the question $get\_pairwise\_xor(\{3, 4\})$, Aidos will answer with the sequence $[0, 0, 5, 5]$, because

- $a_3 \oplus a_4 = 6 \oplus 3 = 5$
- $a_4 \oplus a_3 = 3 \oplus 6 = 5$
- $a_3 \oplus a_3 = 6 \oplus 6 = 0$
- $a_4 \oplus a_4 = 3 \oplus 3 = 0$.

Temirulan failed to cope with the puzzle and your task is to help him. Find the hidden sequence using the questions described above.

Input

Your task is to implement the following function: `int[] guess(int n)`

- n: the length of the hidden sequence.
- The function is called exactly one time for each test.
- The function has to return the hidden sequence in the same order.

Your function can call the following functions:

1. `int ask(int i)`
   - i: index of the number in sequence, $1 \leq i \leq n$.
   - The function returns the i-th number of the hidden sequence.
2. `int[] get_pairwise_xor(int[] pos)`
   - *pos*: **non empty** list of indexes of the sequence.
   - All of the elements in *pos* must be distinct numbers.
   - Let k be the number of elements in *pos*. Then $1 \leq pos_i \leq n$ for each $1 \leq i \leq k$.
   - The function returns sorted list of k^2 elements: a set of pairwise values *xor*, $\{a_{pos_x} \oplus a_{pos_y} \mid 1 \leq x, y \leq k\}$.

You can call both functions no more than 200 times in total for each test. If any of the above conditions are violated, your program will get **Wrong Answer** verdict. Otherwise, your program will get **Accepted** verdict and your score is calculated based on the number of calls of the functions *ask* and *get_pairwise_xor* (Refer to the section “Scoring”).

Scoring

- $2 \leq n \leq 100$
- $0 \leq a_i \leq 10^9$ for each $1 \leq i \leq n$.

In this task, the grader is NOT adaptive. It means that the sequence a is fixed at the beginning of the running of the grader and does not depend on calls from your program.

1. (6 points) $n \leq 4$
2. (94 points) No additionals constraints. For this subtask, your score is calculated in the following manner. Let q be the total number of calls of the functions *ask* and *get_pairwise_xor*.
   - If $q \leq 15$, your score is 94.
   - If $15 < q \leq 40$, your score is $84 - 2(q - 16)$.
   - If $40 < q \leq 50$, your score is 35.
   - Otherwise, your score is 0.

Note

The *xor* operation is the bitwise exclusive OR.

Let the hidden sequence a be $[1, 5, 6, 3]$. Grader calls the function. Example of the interaction is below.

Call	Result
$ask(2)$	5
$get\_pairwise\_xor(\{1, 2, 3\})$	$\{0, 0, 0, 3, 3, 4, 4, 7, 7\}$
$ask(3)$	6
$get\_pairwise\_xor(\{4, 2\})$	$\{0, 0, 6, 6\}$
$get\_pairwise\_xor(\{2\})$	$\{0\}$

The sample grader reads the input in the following format:

- Line 1: n
- Line 2: $a_1, a_2, ..., a_n$