Machine

The ancient Egyptian computer scientists built several machines that shuffled arrays of integers. Thousands of years later, archaeologists discovered one of their machines and examined it.

The machine takes as input an array of N integers and operates in a very predictable manner. It has a built-in permutation P of numbers from 0 to $N-1$ that it uses to shuffle the input array. Specifically, P is an array of length N containing each element between 0 and $N-1$ (inclusive) exactly once.

Because of corrosion, the machine not only shuffles the numbers, but also takes their bitwise XOR with some unknown number X. More formally, the machine takes as input an array A of length N, consisting of non-negative integers. Then, it returns another array B of length N such that $B[i] = A[P[i]] \oplus X$ ($0 \leq i < N$), where $\oplus$ denotes the bitwise XOR operator. Note that X is a fixed number, which does not change when you use the machine.

The bitwise XOR of two non-negative integers c and d is computed as follows. Assume that c and d have at most t bits in their binary representation, that is $\max(c, d) < 2^t$. Then $c \oplus d$ is a number z whose j-th bit ($0 \leq j < t$) is 1 if and only if the j-th bit of c and d is different.

The archaeologists are interested in the built-in permutation P. Your task is to find P by using the machine. The subtasks in this task impose limits on the number of times you can use the machine and the maximum number you can provide in array A.

Implementation Details

You should implement the following procedure:

```
std::vector<int> find_permutation(int N)
```

- N: the length of P and the machine input and output.
- This procedure can be called multiple times in each test case. Each call is called a *scenario*.
- This procedure should return the built-in permutation P.

The above procedure can make calls to the following procedure:

```
std::vector<int> use_machine(std::vector<int> A)
```

- A: an array of length N specifying the input to the machine.
- The elements of A must be integers from 0 to $2^{15} - 1 = 32767$, inclusive.

- This procedure returns an array B such that $B[i] = A[P[i]] \oplus X$ (for all $0 \leq i < N$).
- This procedure can be called at most 129 times in each scenario.

Let T denote the number of scenarios in a test case. The grader calls the `find_permutation` procedure once for each scenario.

Constraints

- $1 \leq T \leq 100$
- $3 \leq N \leq 128$
- $0 \leq X \leq 255$
- $0 \leq P[i] < N$ for each i such that $0 \leq i < N$
- $P[i] \neq P[j]$ (for all i and j such that $0 \leq i < j < N$)

Subtasks

Let Q be the maximum allowed number of calls to the procedure `use_machine` and M be the maximum number that can be provided to the machine as input.

Subtask	Score	Condition	Additional Constraints
1	7	$Q \leq 129; M \leq 2^{15} - 1$	
2	12	$Q \leq 2; M \leq 2^{15} - 1$	
3	19	$Q \leq 1; M \leq 2^{15} - 1$	
4	21	$Q \leq 1; M \leq 2 \cdot N$	
5	10	$Q \leq 1; M \leq N + 2$	N is odd
6	31	$Q \leq 1; M \leq N + 2$	

Examples

Example 1

Consider a scenario in which $P = [0, 1, 2, 3, 4]$ and $X = 3$. The procedure `find_permutation` is called in the following way:

```
find_permutation(5)
```

This procedure may call `use_machine([6, 6, 2, 9, 5])`, which returns $[5, 5, 1, 10, 6]$. At this point, it can be deduced that $X = 3$. The procedure may then call `use_machine([1, 8, 4, 0, 4])`, which returns $[2, 11, 7, 3, 7]$. At this point, P can be deduced and so the procedure should

return $[0, 1, 2, 3, 4]$. In this example, 2 calls are made to the `use_machine` procedure and the maximum number provided as input to this procedure is 9.

Example 2

Consider a scenario in which $P = [0, 4, 3, 1, 2]$ and $X = 8$. The procedure `find_permutation` is called in the following way:

```
find_permutation(5)
```

This procedure may call `use_machine([0, 5, 1, 1, 2])`, which returns $[8, 10, 9, 13, 9]$. Based on this output, P can be deduced and so the procedure should return $[0, 4, 3, 1, 2]$. In this example, only 1 call is made to the `use_machine` procedure, and the maximum number given as input to the machine is 5.

Sample grader

The sample grader input starts with a line containing a single integer T followed by T scenarios in the following format.

```
N  X
P[0]  P[1]  ...  P[N-1]
```

The sample grader writes the output of the T scenarios, each in the following format.

```
S
R[0]  R[1]  ...  R[S-1]
Q'  M'
```

Here, R is the array returned by `find_permutation` and S is its length. Q' is the number of calls to `use_machine` and M' is the maximum number provided as input to any of the calls.