

New Year Experiments

Input file: **standard input**
Output file: **standard output**
Time limit: 2 seconds
Memory limit: 256 megabytes

Ruslan gifted his friend, the hamster Artem, an array a for the New Year. Artem was very happy with the gift and decided to conduct experiments on it.

There are a total of 5 types of operations:

- **AND** — operation: a bitwise AND with the number c is applied to all elements of the array a . That is, each element a_i is replaced with a_i AND c .
- **OR** — operation: a bitwise OR with the number c is applied to all elements of the array a . That is, each element a_i is replaced with a_i OR c .
- **XOR** — operation: a bitwise XOR with the number c is applied to all elements of the array a . That is, each element a_i is replaced with a_i XOR c .
- Query for the value of the i -th element of the array.
- Query for the value of the k -th largest element of the array.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. The following describes the test cases.

The first line contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of elements in the array a .

The second line contains n integers a_i ($0 \leq a_i < 2^{30}$) — the elements of the array a .

The third line contains an integer q ($1 \leq q \leq 2 \cdot 10^5$) — the number of queries.

The next q lines contain queries in the format $x \ c$, where x — the type of operation (from 1 to 5), and c — an integer, for operations 1–3 the constraint is $0 \leq c < 2^{30}$, and for operations 4–5 the constraint is $1 \leq i, k \leq n$.

Description of operations:

- 1 c : apply the AND operation to each element of the array a .
- 2 c : apply the OR operation to each element of the array a .
- 3 c : apply the XOR operation to each element of the array a .
- 4 i : output the value of the i -th element of the array (1-based indexing).
- 5 k : output the value of the k -th largest element of the array (the 1st largest is the smallest element).

It is guaranteed that the sum of all n and q does not exceed $4 \cdot 10^5$.

Output

For each query of type 4 i or 5 k , output the corresponding value on a separate line:

- For a query of type 4 i , output the value of the i -th element of the array (1-based indexing).
- For a query of type 5 k , output the value of the k -th largest element of the array (the 1st largest is the smallest element).

Scoring

Points for each subtask are awarded only if all tests for that subtask and the necessary subtasks are successfully passed.

Subtask	Points	Constraints	Necessary Subtasks	Comments
0	0	–		Tests from the statement.
1	10	$n, q \leq 10^3$, $t \leq 5$, all types of operations		
2	15	$n \cdot q \leq 10^8$, $t \leq 2$, all types of operations	0, 1	
3	15	$q \leq 4 \cdot 10^4$, $t \leq 5$, $\max a_i < 2^{10}$	–	
4	10	no operation of type 5	–	
5	20	$n, q \leq 2 \cdot 10^5$, no operation of type 3	–	
6	10	$n, q \leq 2 \cdot 10^5$, no operations of type 1 and 2	–	
7	20	–	0 – 6	

Example

standard input	standard output
1 3 1 2 3 3 3 1 4 2 5 1	3 0

Note

Consider the input data set from the example. Before executing the queries: $a = [1, 2, 3]$.

- 3 1: apply the XOR operation to each element of the array a with the number 1: $a = [0, 3, 2]$.
- 4 2: output the value of the 2-nd element of the array, the 2-nd element is equal to 3.
- 5 1: output the value of the 1-st largest element of the array, in this case, it is 0.

Bitwise AND (**AND**) is a binary operation on a pair of non-negative integers. To compute the bitwise AND of two numbers, one must consider the representation of both numbers in binary. The result is a

number whose binary representation has a one in each position if there is a one in the binary representation of both arguments.

Bitwise OR (**OR**) is a binary operation on a pair of non-negative integers. To compute the bitwise OR of two numbers, one must consider the representation of both numbers in binary. The result is a number whose binary representation has a one in each position if there is a one in the binary representation of at least one of the arguments.

Bitwise exclusive OR (**XOR**) is a binary operation on a pair of non-negative integers. To compute the bitwise exclusive OR of two numbers, one must consider the representation of both numbers in binary. The result is a number whose binary representation has a one in each position if there is a one in the binary representation of exactly one of the arguments.