XORanges

Janez loves oranges! So he made a scanner for oranges. With a cameras and a Raspberry Pi 3b+ computer, he started creating 3D images of oranges. His image processor is not a very good one, so the only output he gets is a 32-bit integer, which holds information about the holes on the peel. A 32-bit integer D is represented as a sequence of 32 digits (bits) each of which is one or zero. If we start from 0 we can obtain D by adding 2^i for every i-th bit that is equal to one. More formally the number D is represented by the sequence $d_{31}, d_{30}, \dots d_0$ when $D = d_{31} \cdot 2^{31} + d_{30} \cdot 2^{30} + \dots + d_1 \cdot 2^1 + d_0 \cdot 2^0$. For example, 13 is represented as $0, \dots, 0, 1, 1, 0, 1$.

Janez scanned n oranges; however, sometimes he decides to rescan one of the oranges (i-th orange) during the execution of your program. This means that from this scan on, he uses the updated value for the i-th orange.

Janez wants to analyse those oranges. He finds exclusive or (XOR) operation very interesting, so he decides to make some calculations. He selects a range of oranges from l to u (where $l \leq u$) and wants to find out the value of XOR of all elements in that range, all pairs of consecutive elements in that range, all sequences of 3 consecutive elements and so on up to the sequence of $u - l + 1$ consecutive elements (all elements in the range).

I.e. If $l = 2$ and $u = 4$ and there is an array of scanned values A, program should return the value of $a_2 \oplus a_3 \oplus a_4 \oplus (a_2 \oplus a_3) \oplus (a_3 \oplus a_4) \oplus (a_2 \oplus a_3 \oplus a_4)$, where $\oplus$ represents the XOR and a_i represents the i-th element in array A.

Let XOR operation be defined as:

If the i-th bit of the first value is the same as the i-th bit of the second value, the i-th bit of the result is `0`; If the i-th bit of the first value is different as the i-th bit of the second value, the i-th bit of the result is `1`.

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

For example, $13 \oplus 23 = 26$.

$13 =$	$0 \ldots 001101$
$23 =$	$0 \ldots 010111$
$13 \oplus 23 = 26 =$	$0 \ldots 011010$

Input

In the first line of the input there are 2 positive integers n and q (total number of rescans and queries - actions).

In the next line, there are n space-separated non-negative integers, which represent values of the array A (scan results for oranges). Element a_i contains the value for i-th orange. Index i starts with 1.

Actions are described in the next q lines with three space-separated positive integers.

If the action type is 1 (rescan), the first integer equals 1 and is followed by i (index of an orange that Janez wants to rescan) and j (the result of the rescan of the i-th orange).

If the action type is 2 (query), the first integer equals 2 and is followed by l and u.

Output

You should print exactly one integer for each query with the matching result for the query. You should print every value in a new line. Note that the i-th line of the output should match the result of the i-th query.

Constraints

- $a_i \leq 10^9$
- $0 < n, q \leq 2 \cdot 10^5$

Subtasks

1. **[12 points]**: $0 < n, q \leq 100$
2. **[18 points]**: $0 < n, q \leq 500$ and there are no rescans (actions of type 1)
3. **[25 points]**: $0 < n, q \leq 5000$
4. **[20 points]**: $0 < n, q \leq 2 \cdot 10^5$ and there are no rescans (actions of type 1)
5. **[25 points]**: No additional constraints.

Examples

Example 1

Input

```
3 3
1 2 3
2 1 3
1 1 3
2 1 3
```

Output

```
2
0
```

Comment

At the begining, $A = [1, 2, 3]$. The first query is on the full range. The result of the analysis is $1 \oplus 2 \oplus 3 \oplus (1 \oplus 2) \oplus (2 \oplus 3) \oplus (1 \oplus 2 \oplus 3) = 2$.

Then the value of the first orange is updated to 3. This leads to a change on the same query (on a range $[1, 3]$) $3 \oplus 2 \oplus 3 \oplus (3 \oplus 2) \oplus (2 \oplus 3) \oplus (3 \oplus 2 \oplus 3) = 0$.

Example 2

Input

```
5 6
1 2 3 4 5
2 1 3
1 1 3
2 1 5
2 4 4
1 1 1
2 4 4
```

Output

```
2
5
4
4
```