Those Who Continue to Wait, and Those Who Do Not Come Back

‘

Time Limit: 1.5 Second
Memory Limit: 2048 MB

— Who am I? I don't know... But I think there was something important. That's right. I have to go back.
— No matter what?
— No matter what.
— Even though it'll only bring you lots of pain?
— Well, yeah. But I have a promise to keep.
— I see...

Elq promises to send Chtholly back to her world, but she has a small request for Chtholly. Elq loves data structures so she asks Chtholly to design a new data structure for her before she leaves so that she can play with it when Chtholly is not here. However, Chtholly cannot wait to see Willem again, so she asks you for help because she heard that you just learned a lot of data structures! She promises to share with you the butter cakes Willem made for her if you can complete Elq's request.

Given an array $a_1, \ldots, a_n$ with n elements, the data structure needs to support the following basic operations:

- `Assign` l r x ($1 \le l \le r \le n$, $1 \le x \le 10^9$): change all elements in range $[l, r]$ to x.
- `Multiply` l r x ($1 \le l \le r \le n$, $1 \le x \le 10^9$): for each element in range $[l, r]$, multiply it by x.
- `Divide` l r x ($1 \le l \le r \le n$, $1 \le x \le 1000$): for each element in range $[l, r]$ that is divisible by x, divide it by x.

In addition, Elq wants your data structure to support the following types of query (which are supported by some data structures that you might or might not learned in the course):

- `Sum` l r ($1 \le l \le r \le n$): find the sum of all elements in range $[l, r]$. (Fenwick Tree)
- `Max` l r ($1 \le l \le r \le n$): find the maximum element in range $[l, r]$. (Binary Lifting)
- `Xor` l r ($1 \le l \le r \le n$): find the exclusive or of all elements in range $[l, r]$. (Sqrt Tree)
- `Kth` l r k ($1 \le l \le r \le n$, $1 \le k \le r - l + 1$): find the k-th largest element in range $[l, r]$. (Persistent Segment Tree)
- `Greater` l r x ($1 \le l \le r \le n$, $1 \le x \le 10^9$): count the number of elements in range $[l, r]$ that is strictly greater than x. (Sqrt Decomposition)
- `Unique` l r ($1 \le l \le r \le n$): find the number of unique values in range $[l, r]$. (Fenwick Tree)

- `Pair` l r ($1 \leq l \leq r \leq n$): find the number of pairs (i, j) such that $l \leq i \leq j \leq r$ and $a_i = a_j$. (Mo's Algorithm)
- `Subarray` l r x ($1 \leq l \leq r \leq n$, $1 \leq x \leq 10^9$): find $\max_{l \leq l' \leq r' \leq r} \sum_{i=l'}^{r'} a_i - x$ (i.e. maximum subarray sum of elements in range $[l, r]$ if all elements decrease by x). (Segment Tree)
- `Prime` l r ($1 \leq l \leq r \leq n$): find the number of prime numbers in range $[l, r]$. (Fenwick Tree)
- `ReLU` l r x ($1 \leq l \leq r \leq n$, $1 \leq x \leq 10^9$): find $\sum_{i=l}^{r} \max(a_i, x)$. (Segment Tree Beats)

Input

The first line of input contains two integers n and q ($1 \leq n, q \leq 10^5$) - the number of elements in the array and the number of operations.

The next line contains n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) - the elements of the array.

The next q lines describe the operations. Each line contains a single operation described above. **It is guaranteed that all test cases are generated randomly and the values of $a_1, \ldots, a_n$ will not exceed 10^9 after each operation.**

Output

For each query, output a single integer denoting the answer.

Sample Inputs

```
5 13
1 2 3 4 5
Xor 3 5
Subarray 1 4 2
Kth 2 5 3
Assign 2 4 10
Unique 1 5
Pair 1 4
Multiply 1 2 5
Greater 1 3 10
Max 1 5
Divide 1 5 10
Sum 2 4
ReLU 2 4 3
Prime 1 5
```

Sample Outputs

```
2
3
3
3
7
1
50
7
11
3
```