Czech ACM Student Chapter
Charles University in Prague
Masaryk University
Pavol Jozef Šafárik University in Košice
Slovak University of Technology

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2023

Expressions

`expressions.(c|cpp|py)`, `Expressions.(java|kt)`

Part of the training in arithmetic classes in Tomorrow Programming School consists of quick evaluation of a given arithmetic expression, in the head only. Fortunately, the expressions contain only positive integers, no brackets, and only three operations: addition, subtraction, and multiplication. Thus, the result is guaranteed to be an integer. Moreover, at introductory stages, a student has to evaluate only the parity of the result – is it odd or even?

Unfortunately, the professor presenting the expressions on the blackboard is known to be quite absentminded. He often rewrites various numbers in an expression, even more times, while the students are already calculating. Typically, it changes the value of the whole expression, and the students have to start their calculations all over again.

The students decided to write a program which would help them. The input of the program will be the original expression, and the sequence of subsequent value changes in it. For each change, the program will calculate the value of the modified expression, possibly even without recalculating the whole expression from scratch.

Input Specification

The first input line contains two numbers N and M ($1 \leq N, M \leq 10^5$), the number of integers in the expression, and the number of subsequent value changes in the expression. The second input line contains N token-separated integers $A_1, \ldots, A_N$ ($1 \leq A_i \leq 10^9$). Each token is `+`, `-`, or `*` (plus, minus, star (multiplication)). Each of the next M lines describe one value change in the expression. It contains two space-separated numbers X ($1 \leq X \leq N$) and Y ($1 \leq Y \leq 10^9$), which means that the X-th number in the expression was changed to value Y.

The changes are considered to be subsequent, i.e., when a next change appears on a different position, the previous change in the expression is preserved.

Output Specification

Output $M + 1$ lines. The i-th line contains either "`even`" or "`odd`", depending on whether the input expression evaluates to an even or to an odd value after application of the first $i - 1$ changes. In particular, the first output line indicates whether the expression evaluates to an even or to an odd value before any change was done. The evaluation follows standard arithmetic rule of operations precedence. Multiplication is granted higher precedence than addition and subtraction.

Sample Input 1

```
6 4
11 + 22 * 33 - 44 * 55 * 66
1 2
2 3
4 5
3 5
```

Output for Sample Input 1

```
odd
even
odd
odd
odd
```