110806 Garden of Eden

Cellular automata are mathematical idealizations of physical systems in which both space and time are discrete, and the physical quantities take on a finite set of discrete values. A cellular automaton consists of a lattice (or array) of discrete-valued variables. The state of such automaton is completely specified by the values of the variables at each position in the lattice. Cellular automata evolve in discrete time steps, with the value at each position (cell) being affected by the values of variables at sites in its neighborhood on the previous time step. For each automaton there is a set of rules that define its evolution.

For most cellular automata there are configurations (states) that are unreachable: no state will produce them by the application of the evolution rules. These states are called Gardens of Eden, because they can only appear as initial states. As an example, consider a trivial set of rules that evolve every cell into 0. For this automaton, any state with non-zero cells is a Garden of Eden.

In general, finding the ancestor of a given state (or the non-existence of such an ancestor) is a very hard, computing-intensive, problem. For the sake of simplicity we will restrict the problem to one-dimensional binary finite cellular automata. In other words, the number of cells is a finite number, the cells are arranged in a linear fashion, and their state will be either "`0`" or "`1`." To simplify the problem further, each cell state will depend only on its previous state and that of its immediate left and right neighbors.

The actual arrangement of the cells will be along a circle, so that the last cell is a neighbor of the first cell.

Problem definition

Given a circular binary cellular automaton, you must determine whether a given state is a Garden of Eden or a reachable state. The cellular automaton will be described in terms of its evolution rules. For example, the table below shows the evolution rules for the automaton: $Cell = XOR(Left, Right)$.

Left $[i-1]$	Cell $[i]$	Right $[i+1]$	New State	
0	0	0	0	$0 * 2^0$
0	0	1	1	$1 * 2^1$
0	1	0	0	$0 * 2^2$
0	1	1	1	$1 * 2^3$
1	0	0	1	$1 * 2^4$
1	0	1	0	$0 * 2^5$
1	1	0	1	$1 * 2^6$
1	1	1	0	$0 * 2^7$
				90 = Automaton Identifier

With the restrictions imposed on this problem, there are only 256 different automata. An identifier for each automaton can be generated by taking the *new state* vector and interpreting it as a binary number, as shown in the table. The example automaton has identifier 90, while the *identity* automaton (where every state evolves to itself) has identifier 204.

Input

The input will consist of several test cases. Each input case describes a cellular automaton and a state on a single line. The first item on the line will be the identifier of the cellular automaton you must work with. The second item in the line will be a positive integer N ($4 \leq N \leq 32$) indicating the number of cells for this test case. Finally, the third item in the line will be a state represented by a string of exactly N zeros and ones. Your program must keep reading lines until the end of file.

Output

If an input case describes a Garden of Eden, output the string `GARDEN OF EDEN`. If the input does not describe a Garden of Eden (it is a reachable state) you must output the string `REACHABLE`.

The output for each test case must be on a different line.

Sample Input

```
0 4 1111
204 5 10101
255 6 000000
154 16 1000000000000000
```

Sample Output

```
GARDEN OF EDEN
REACHABLE
GARDEN OF EDEN
GARDEN OF EDEN
```


