There are 8 prison cells in a row, and each cell is either occupied or vacant.

Each day, whether the cell is occupied or vacant changes according to the following rules:

* If a cell has two adjacent neighbors that are both occupied or both vacant, then the cell becomes occupied.
* Otherwise, it becomes vacant.

(Note that because the prison is a row, the first and the last cells in the row can't have two adjacent neighbors.)

We describe the current state of the prison in the following way: cells[i] == 1 if the i-th cell is occupied, else cells[i] == 0.

Given the initial state of the prison, return the state of the prison after N days (and N such changes described above.)

**Example 1:**

**Input:** cells = [0,1,0,1,1,0,0,1], N = 7

**Output:** [0,0,1,1,0,0,0,0]

**Explanation:**

The following table summarizes the state of the prison on each day:

Day 0: [0, 1, 0, 1, 1, 0, 0, 1]

Day 1: [0, 1, 1, 0, 0, 0, 0, 0]

Day 2: [0, 0, 0, 0, 1, 1, 1, 0]

Day 3: [0, 1, 1, 0, 0, 1, 0, 0]

Day 4: [0, 0, 0, 0, 0, 1, 0, 0]

Day 5: [0, 1, 1, 1, 0, 1, 0, 0]

Day 6: [0, 0, 1, 0, 1, 1, 0, 0]

Day 7: [0, 0, 1, 1, 0, 0, 0, 0]

**Example 2:**

**Input:** cells = [1,0,0,1,0,0,1,0], N = 1000000000

**Output:** [0,0,1,1,1,1,1,0]

**Note:**

1. cells.length == 8
2. cells[i] is in {0, 1}
3. 1 <= N <= 10^9