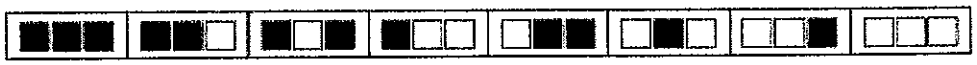
Elementary Cellular Automata

Filename: `automata`

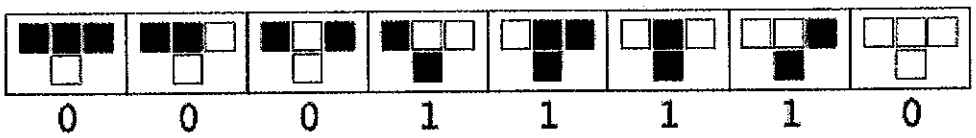
Cellular automata are an interesting application of computer science, simulating out processes by creating local rules on a grid with individual "cells." Each unit of time, or "generation," the cells change based on a set of rules, and the process is repeated on the new cells.

The simplest kind of cellular automata are one-dimensional or elementary cellular automata. The idea is that you start with a certain number of cells, each of which can be "dead" or "alive" arranged on a Möbius strip (i.e., it wraps from the end back to the beginning), and during each generation, consider cell and the states of its left neighbor, itself, and its right neighbor to decide the cell's state for the next step. Being the savvy mathematician, you know that means that for any given cell, there can be 2^3, or 8, combinations as shown in graphical form below.

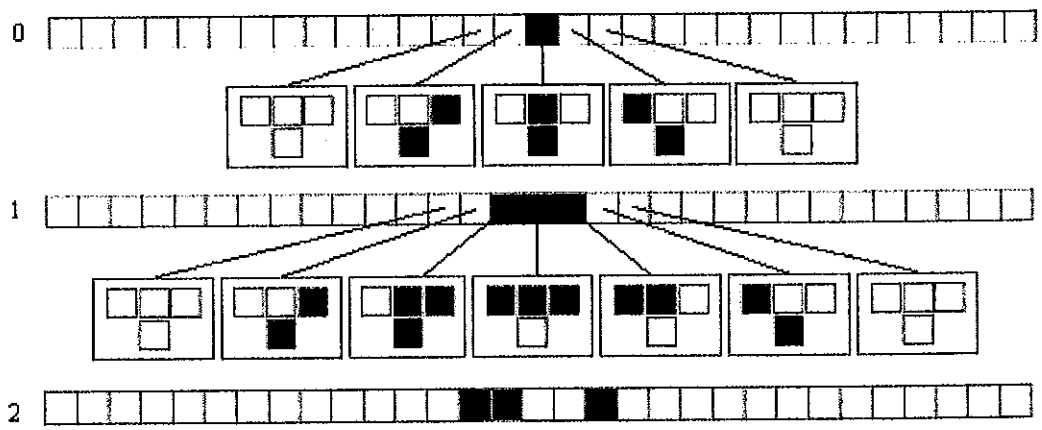


For each of those 8 combinations, the result can be either on or off as a result, depending on the rules of the automata. This means that there are 2^8, or 256, different elementary cellular automata, one for each possible result of each possible state.

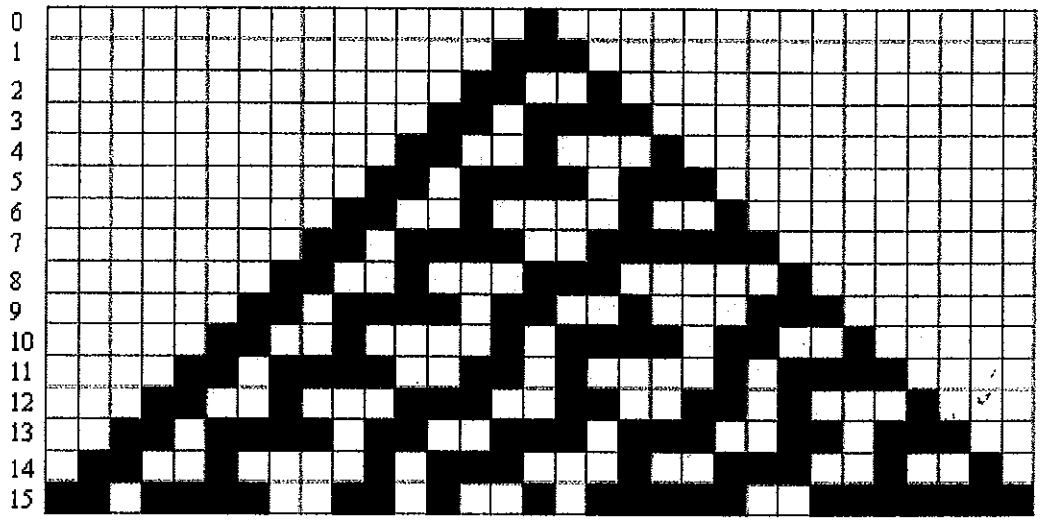
For example, here is one possible rule set, with each triplet of squares representing a state (the middle cell is the one being considered), and the square below it representing a result, with a white square being "dead" or "0", and a black square being "alive" or "1".



For example, here is two generations of an automata using the above rules, starting with a single cell.



Showing the original generation and fifteen generations at the same time yields an interesting triangular pattern:



The Problem:

Given the initial state of a cellular automata and a rules set, print a two dimensional text image representing the progression of the states over a number of generations.

The Input:

The first line of the input will contain a positive integer, t ($t \leq 1000$), of automata to follow. Each automata will consist of a pair of integers, g ($0 \leq g \leq 100$) and w ($3 \leq w \leq 100$), followed by two strings on separate lines consisting of only zeroes and ones. g is the number of generations that should be simulated and w is the number of cells in the simulated space.

The first string will be exactly w characters long and will represent the initial state of the space. A "0" denotes a dead cell, while a "1" denotes a living cell. The second string will be exactly 8 characters long and represents the rule set for an automata, given in the order shown in the picture on the previous page. A "1" will result in a live cell, and "0" means that the state will result in a dead cell.

For example, "00011110" will result in a live cell if the current cell and the cell to the right is alive, or only a single of the three cells (left, current, and right) are alive. All other configurations will result in a dead cell.

The Output:

For each automata, output a header "`Automata #a:`" where a is the number of the automata, starting from 1, on its own line. Starting from the next line, output the initial state of the grid, followed by g states, each on their own line, of the board as a series of characters. A living cell should be denoted by an octothorpe ("#") and a dead cell by a period (".") . Each automata should be followed by a blank line.

Sample Input:

```
3
0 5
01010
11111111
3 9
000010000
00011110
3 6
010010
00010000
```

Sample Output:

```
Automata #1:
.#.#.

Automata #2:
....#....
...###...
..##..#..
.##.####.

Automata #3:
.#..#.
..#..#
#..#..
.#..#.
```