Elementary Cellular Automaton

Elementary Cellular Automaton

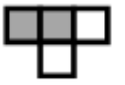
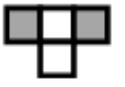
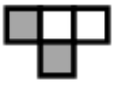
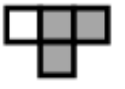
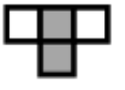
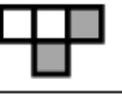
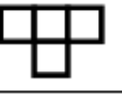
Elementary Cellular Automaton (ECA) is a discrete modeling technique used in science and engineering to study the behavioral patterns that emerge in nature. For more information see http://mathworld.wolfram.com/ElementaryCellularAutomaton.html.

The state of the automaton consists of an array of *N* cells that can take either a TRUE or FALSE state. For each iteration, a new state is computed based on the current state and a set of rules for the automaton.

The new state for cell *i* is calculated based on the previous state of cell to the left (*i-1*), same position (*i*), and to the right (*i+1*). If the state happens to coincide with an edge (e.g., when *i=0*), then the "imaginary" cell is assumed to be FALSE.

There are 2×2×2=8 possible binary states for the three cells neighboring a given cell as shown in the table below. Furthermore, there are a total of 2^8=256 elementary cellular automata, each of which can be indexed with an 8-bit binary number. This is usually known as a rule. For example, rule 30 (30 base 10 = 00011110 base 2) is shown in the table below. In the diagram, the previous state's left, middle, and right cells are shown in the top row and the second row shows the current cell in the new state being modified.

The rules for this assignment are as follows:

#	Left Cell (i-1)	Middle Cell (i)	Right Cell (i+1)	New Cell (State j+1)	Pictorial Representation
1	TRUE	TRUE	TRUE	FALSE	
2	TRUE	TRUE	FALSE	FALSE	
3	TRUE	FALSE	TRUE	FALSE	
4	TRUE	FALSE	FALSE	TRUE	
5	FALSE	TRUE	TRUE	TRUE	
6	FALSE	TRUE	FALSE	TRUE	
7	FALSE	FALSE	TRUE	TRUE	
8	FALSE	FALSE	FALSE	FALSE	

For example, consider the case where *N=10* and it is set as follows for state *j* :

Then, to calculate state *j+1* , we need to iterate through every cell as follows. For each case, the neighbors are highlighted to make it easier to see.

When *i=0* , the previous state is {FALSE, FALSE, TRUE}, which corresponds to rule #7 that says the result should be TRUE. Note that in the previous state, the cell at *i-1* does not exist, so we assume it is FALSE.

Then, we can calculate the next cell at position *i=1* which corresponds to rule #6.

Continuing on, we see the following changes for *i=2* through *i=9* .

Thus, the new state has a different pattern of TRUE/FALSE cells. The process then repeats starting with the new state.

The cellular automaton ends after the cells reach a steady state (no more changes from one iteration to the next) or one reaches a maximum number of iterations.

Task

Write a program that can draw any rule in the elementary cellular automaton given a rule number, integer N for the number of cells, and initial conditions for the cells. The program should output the automaton until either of the two terminating conditions is met.

Input

The input will contain the following single-space separated parameters:

- 1 <= Rule # <= 256

- 0 <= Maximum number of iterations to print < 1000
- 8 <= Number of cells, N <= 64
- 0 <= Integer representing initial condition for N cells <= 2^64 -1
  (where the binary representation shows the TRUE/FALSE state of the N cells)

Output

The program should output the iteration number formatted as a left-aligned 3-digit number padded with spaces. Then, each iteration of the automaton should be displayed with cells marked as TRUE shown as an asterisk character (*) and cells marked as FALSE should be a space character ( ). The cells in the state should be surrounded by dashes to make it easier to differentiate when the empty states are.
Note: There is a newline character at the end of the last line of the output.

Sample Input 1

---

30 20 8 16

Sample Output 1

```
1   -   *     -
2   -  ***    -
3   - **  *   -
4   -** **** -
5   -*  *    *-
6   -***** **-
7   -*      * -
8   -**    ***-
9   -* * **|  -
10  -* * * * -
11  -* * * **-
12  -* * * * -
13  -* * * **-
14  -* * * * -
15  -* * * **-
16  -* * * * -
17  -* * * **-
18  -* * * * -
19  -* * * **-
20  -* * * * -
```