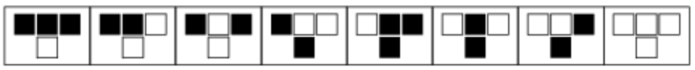
Exercises for Math Circles 2018

November 28

Some of the exercises below will require use of **Wolfram Alpha**. But for a more powerful option, try **Wolfram Programming Lab** — you will have to sign up (it's free), but then you'll see how handy it is.

1. Here is Rule 30.



   Starting with a single live cell, iterate these rules ten times by hand. Then open up Wolfram Lab and enter the code

```
ArrayPlot[CellularAutomaton[30,{{1},0},10]]
```

   to get the computer to do it.

2. Use binary expansions to write the transition rules for Rule 90.

3. Use binary expansions to figure out the rule number for the following transition rules.

4. How many (1-dimensional elementary) cellular automata are there in total?

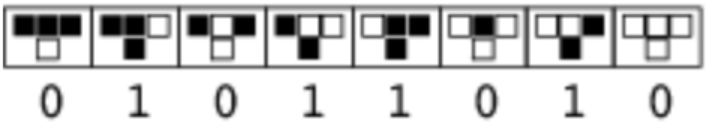
5. Enter code into Wolfram Lab to draw Rule 173 with initial state `10101101` (padded with zeros) and 100 iterations. Here `1` means a black cell, and `0` means a white cell.

6. The code `RandomInteger[1,250]` generates 250 random 0's and 1's. Use this to iterate Rule 173 on some random initial state 100 times.

7. Enter the code `RulePlot[CellularAutomaton[126]]` to print out the transition rules for Rule 126.

8. Prove that the state `{{1},0}` is a Garden Of Eden for Rule 126. (This one is pretty hard: you have to show that Rule 26 will never reach this state no matter what initial state you start with.)

9. Here is Rule 90.



   Find predecessors for each of the following states in Rule 90.

   (a) `00011000`

   (b) `000111000`

   (c) `000101100111000`

   Convince yourself that every state has a predecessor.

10. **Multistate automata:** The following code will generate the 3-color Rule 679,458

```
CellularAutomaton[{679458,3}]
```

Enter this into Wolfram Lab and see what happens! (Don't forget to encase this line in `ArrayPlot[...]`.)

11. To change the colors, use `ColorRule->{Pink->1,Blue->2}`. This will change `1`'s into pink cells, and `2`'s into blue cells.

```
ArrayPlot[CellularAutomaton[{679458,3},{{1},0},50],ColorRules->{Pink->1, Blue->2}]
```

Enter this into Wolfram Lab and see what happens! Also, try to generate the rule plot for 3-color Rule 679,458.

12. **Totalistic rules:** The following code will generate the 3-color totalistic Rule 2049

```
CellularAutomaton[{2049,{3,1}}]
```

(Don't worry about the `1` for now.) Try iterating this on a random initial state, 250 times, and output the plot. Then change the colors: change the `1`'s to red, and the `2`'s to orange.