Homework 3
02-201: Programming for Scientists
Frequent Words and Cellular Automata
Due: Thursday, Feb. 11, 2015 at 11:59 PM

Reading

Read through Chapter 7 of *An Introduction to Programming in Go*.

Solving the Frequent Words Problem

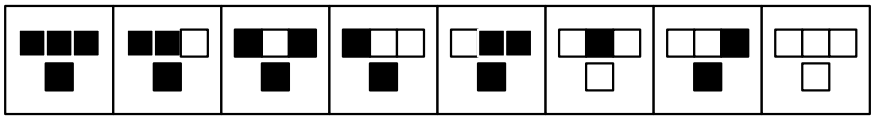
You should now be able to implement functions in Go solving the Frequent Words Problem and the Minimum Skew Problem.

You can solve these at http://rosalind.info/problems/ba1b and http://rosalind.info/problems/ba1f/.

Introduction to Cellular Automata

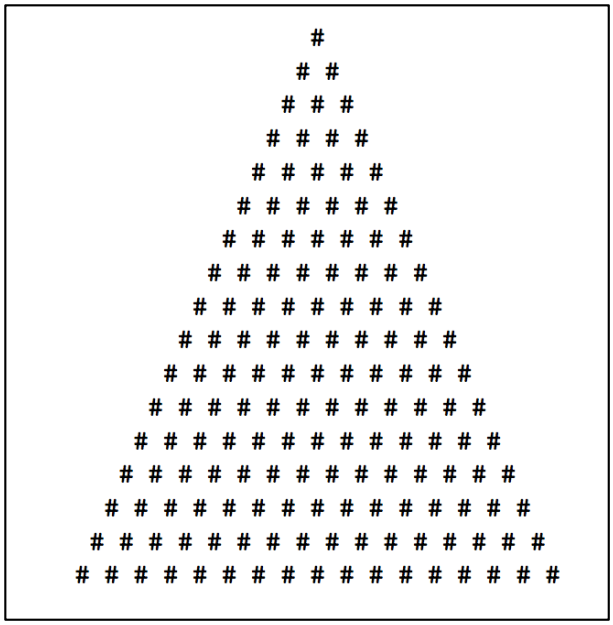
Cellular automata (CAs) are a simplified model of physics and complex systems that has been studied for many years. One-dimensional CAs are defined by the following rules:

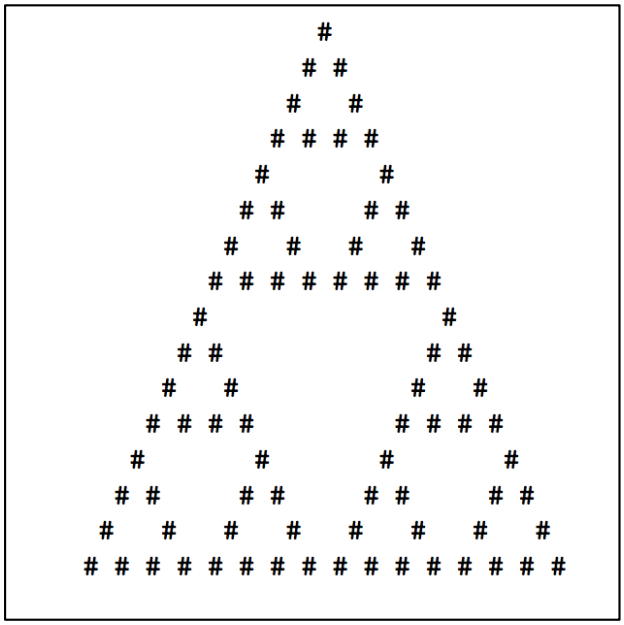
- The "universe" is an infinitely long line of "cells."
- Each "cell" can either be *filled* or *empty*.
- The state of a cell i at time t depends on the states of cells $i-1$, i, and $i+1$ at time $t-1$. The rule for how cell i depends on these cells defines the particular CA you are studying. The rules are represented by diagrams of the following type:



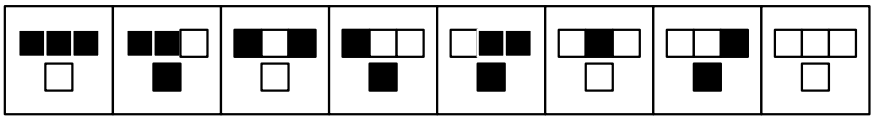
  For example, the first part of the rule above means that if cell $i-1$, i and $i+1$ are all *filled* at time $t-1$, then at time t, cell i will be *filled*. All the sub rules (boxes above) are applied simultaneously during a single step.

The interesting thing about CAs is that they can give surprisingly complex behavior. For example, the rule above produces the pattern on the left:





The rule



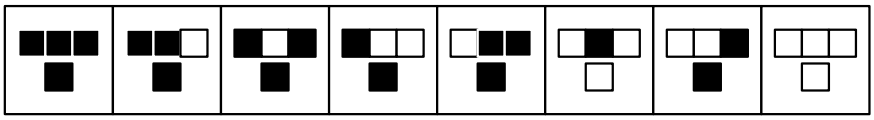
Produces the pattern on the right above, something even more complex.

The Autolab Assignment

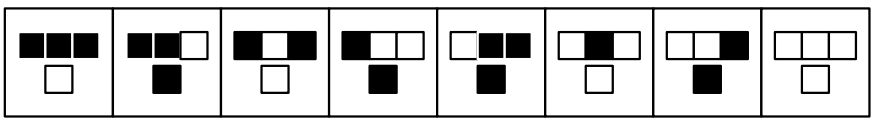
Input. You will write a program that takes in a CA rule and produces a given number of iterations of the CA. Your command should be able to be run using the following command line:

```
go run ca.go RULE WIDTH STEPS
```

Where `STEPS` is the number of steps to run the CA. Since we can't work with an infinitely long sequence of cells, we will only model `WIDTH` cells. Finally, `RULE` is a sequence of 8 `'0'`s and `'1'`s that give the outcome of each subrule in the order used in the examples above. For example, the rule:



would be given as "`11111010`" and the rule



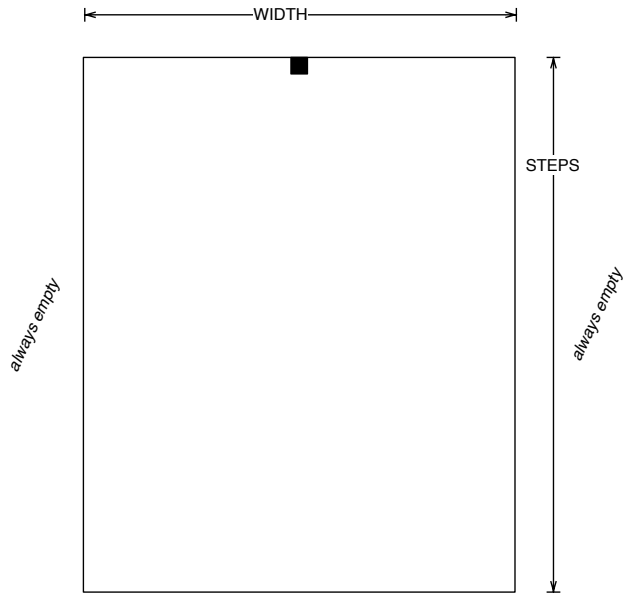
would be given as "`01011010`".

Output. At each step, you should output the state of the cells. You should print an *empty* cell as a single space and a *filled* cell as a `#` character as in the examples above. You should output the state of the cells at the start too.

Your cells should initially all be *empty*, except for the cell at $\lfloor$`WIDTH`$/2\rfloor$, which should be *filled*. (The $\lfloor x \rfloor$ notation means throw out the fractional part of x, e.g., $\lfloor 3.1415 \rfloor = 3$.)

If any of the command line parameters are not valid, you should output an error message that begins with `Error:` and stop the program.

Your program should assume that any cells outside of the `WIDTH` cells you are modeling are *empty* forever. That is you are just modifying cells $0 \ldots$ `WIDTH` $- 1$. The set up thus looks like this:



Learning outcomes

After completing this assignment, you should:

- be able to create arrays (using slices in Go).
- be able to walk through the items in an array.
- be able to access characters in a string.
- have gained additional practice writing for loops, if statements, and functions.
- have practiced writing code with good "style" (formatting, modularity, comments).