CMPT 470/816 HOME EXERCISE 1
IMPROVING CODE AESTHETICS & FUNCTIONAL ABSTRACTION

This example introduces a simple program whose variants will follow us through the semester, as implemented in different programming paradigms (e.g., procedural, functional, object-oriented, aspect-based).

Suppose that you take on a new job, part of whose responsibilities involves maintaining and evolving existing code bases. One of these projects involves a program to compute a cellular automaton known as 'Conway's Game of Life' after the mathematician John Conway who invented in. Browsing online, you soon find extensive information on this cellular automaton and its features. While you are welcome to consult such sources, a few of the essentials are given below.

Like many other “cellular automata”, this program consists of a discrete space occupied by 'patches' (also called 'cells'), and a time proceeding in discrete time steps (or 'ticks'). For each tick, there is a simple functional rule applied uniformly to each patch in turn. This rule is typically local in character (using only information about the surrounding patches, often the immediate neighbors). Very importantly, this update rule is atomic in that it uses only the current state of the system to determine the *next state* – i.e., the computed update to each patch during a given tick depends only the the patches in the current state in that tick, and its update is independent of the updates being computed to other patches. Because of this atomicity, it thus does not matter in what order the patches are updated.

In this (“Conway's Game of Life”) automaton, each patch is either occupied or empty. The update rule examines the current environment of that patch and determines whether that patch is to be occupied or empty in the next time step. In updating a given patch, the update role only depends on the neighbouring patches in the 8 compass directions -- “North”, “South”, “East”, “West”, “Northeast” , “Northwest”, “Southeast”, “Southwest” (where the patch to the “North” of a patch P that is located at row Y and column X means the patch row Y-1 and column X, the patch to the “South” of P patch is refers to the patch at row Y-1 and column X, the patch to the “East” of P means refrs to the patch at row Y and column X-1, etc.).

Within Conway's Game of Life, a patch that is occupied (“alive”) will only remain alive if there are exactly 2 or 3 neighbouring patches occupied; otherwise, it “dies” and becomes empty. By contrast, a patch that is empty at timestep t will remain empty in timestep t+1 except if it has exactly 3 occupied neighbours (in the same compass directions as referred to above). If it does have exactly 3 such neighbours, it will be “colonized” and become live in the next timestep.

Someone who was working for your company (but who is no longer with it) has implemented Conway's Game of Life in C, including reading in the initial state of the cellular automata, running it for a fixed number of timesteps, and saving out the result. However, you hear that the existing codebase is not very well maintained.

There is talk about future evolutions to change this codebase to offer greater functionality and such that it can be more easily evolved, but your initial job with this project is to refactor the codebase so that it is cleaner, but especially better structured – particularly by breaking it up into well-defined pieces captured as functions. Because the original creator is not available for consultation, this will involve a certain amount of sleuthing and reasoning on your part, but your employers are confident that you are up to the job. Once you are up to speed, they also promise to start involving you in disucssions as to priorities for future development of this program.

Due in class January 12, 2017

CMPT 470/816 Home Exercise 1
Improving Code Aesthetics & Functional Abstraction

Please open up the file "ConwaysGameOfLifeUglyV1.c" to see the original code. It is requested that you copy this file and then refactor it to clean up the code. It is particularly requested that you seek to clean it up by inserting functions. Please note that when using GCC, this file must be compiled with with -std=c99 option, such as in the following:

```
gcc  -std=c99 ConwaysGameOfLifeUglyV1.c
```

Along with the program, on the development machine, you find a number of apparent input files, which are all ".txt" files. These are also packaged into this take-home exercise.

Due in class January 12, 2017