CSCI 247 Computer Systems I
Homework 3 : Game of Life

In this homework assignment you will interface with a cellular automation simulator. The rules are very simple. The board is a grid. Each grid square can hold at most one cell. For a cell to survive from generation to the next, it needs two or three neighbors. Too many neighbors cause the cell to starve. Too few neighbors causes an incurable case of ennui. All is not lost. If a grid square has exactly three neighbors, a new cell is born, which is kind of strange when you think about it.

In this simulation, a grid square that includes a cell is marked as 1, and 0 otherwise. For example

$$\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 0 & \underline{1} & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{array}$$

would represent a 16 square grid, with 7 cells, where the underlined cell has no neighbors.

There are many variations on this game, which is Conway's Game of Life and is a B3/S23 game. That is, cells survive with two or three neighbors and are born with exactly 3 neighbors. This game is can be extended into the real domain or more dimensions. However, we will stick with the original B3/S23.

You are being given a very naive evolution implementation called `evolve`. It works, but it is slow. See how fast you can make it run (we've discussed optimization techniques).

Outcomes: Understand and apply the loop and other optimizations we discussed, including:

- Remove loop inefficiencies and reduce procedure calls
- Reduce unnecessary memory references and loop unrolling

You CAN work in groups of 2 in completing this assignment. In that case, only a single person uploads .c files to Canvas, and the names of both people should be in the header of each file.

The instructions in this document are meant to be executed on the lab machines (you need to ssh in). If you want to work on your personal machine, see section **Working on your personal computer**.

Details

The experimental set-up is implemented in a library called *liblife.a*. Since it has solutions (different types of `evolve`), it is a binary, which is saved to `/home/jagodzf/lib`. A provided *Makefile* and *life.c* are available at the following, which you should retrieve.

```
/home/jagodzf/public_html/teaching/csci247/homeworks/hw3-gameOfLife
```

There are multiple ways to run the simulation/test program.

- **Silent mode.** The program runs the test cases against the provided `evolve` methods for 500 generations. It then reports the average performance of each method. The performance value is an equation that considers memory moves, efficiency, memory accesses, etc. A low value performance measure specifies that fewer resources were needed to run the program. Hence, a lower performance value is preferred. This is the default mode.
- **Full GUI mode**. The simulator will attempt to open a new WINDOW showing the simulation running. If you are connecting via SSH to the CS computers, you must export your view. This is invoked via `–fg`.
- **Lazy GUI mode**. This is a command-line version of the GUI, that does not open a new WINDOW (hence is suitable for invoking via SSH). This will print the top-left 10 x 10 cells of the 1024 x 1024 grid for each generation. Invoke via `–lg`.

For example, compiling and invoking via the simple GUI mode:

```
$ make
$ ./life -lg
```

All invocations will print the generation number, as well as the final stats for each implementation of the `evolve` method.

Working on your personal computer

Although you can download the `lib` (and header file at `/home/jagodzf/include/`), note that the Makefile references `lib` and `include` in the `jagodzf` directory, so if you want to build the executable on your personal computer, you'll need to modify Makefile to direct it to the local locations of `lib` and `include`.

Your task

- Improve the performance of the simulation by modifying the `evolve` method. Do this by implementing various optimization techniques. Refer to the lecture slides.
- You can modify any code in *life.c*, but make sure your edits don't break the simulation run.
- Please name your implementation the same as your CS username. In other words, replace "Simple" with your userID, so that when the program runs, it might print the following:

```
jagodzf:  5.61
jobss: 27.22
gatesb: 16.44
turinga: 10.68
lovelaa:  7.37
```

Hints and suggestions

- You are free to add as many test methods as you want, and you are encouraged to do this. Copy the code, give it a new name, and add the new `evolve` method to the test harness (e.g, `add_method`). This will let you see if you are making improvements.

Grading

Your implementation will be evaluated against Simple, jobs, gatesb, turinga, and lovelaa. It will be tested using the silent (default) mode.

Rubric and Submission

Upload your *life.c* file to Canvas*.* If for some reason you also develop other (custom) .c and .h files, upload them as well.

Correctness	Points
Beating Simple (the provided solution)	7
Beating jobss	6
Beating gatesb	5
Beating turinga	1
Beating lovelaa	1
Coding style, including informative comments, well structured code, meaningful function and variable names, and helper functions when necessary	3
Total	22 points