ESS201: Programming-II
Module: C++

Jaya Sreevalsan Nair
jnair@iiitb.ac.in
International Institute of Information Technology, Bangalore

Lab-04: Lab assignment on October 30, 2018
Submission Deadline: 11:59:59 pm IST, November 09, 2018 (Friday)

1 Premise: Conway's Game of Life

Conway's Game of Life is the simplest two-dimensional cellular automaton, devised by John Horton Conway, in 1970. Cellular automaton, a discrete model studied in varied STEM (science, technology, engineering, mathematics) applications, is a regular grid of cells, where each cell is in one of the finite number of states, e.g. on or off. The grid changes state based on a *fixed* rule applied to its neighborhood.

In Conway's life of game, each cell has eight neighbors (Moore neighborhood of range-1 cell) and two states ("live" and "dead"). The rules applied in Conway's Game of Life allow transitions between live and dead states of the cell (i.e. between on and off states). At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbors dies, as if by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by starvation or overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

2 Multiverse

Assume the universe is a game of life. Then a *multiverse* contains several (parallel) universes. Thus, we introduce the notion of a *grid of grids* as a data structure for the multiverse, which we call as the *supergrid*. We refer to the multiverse as `multiverse(R,C)`, where all the grids in the multiverse are of size `R x C`. We assume the following about the multiverse:

1. Grids in the multiverse are laid out same as grid cells in the grid in the game of life, i.e. in the form of a grid.
2. There are grids which are boundary and non-boundary, similar to grid cells in a grid.
3. Each grid has a Moore neighborhood of range-1 grids.
4. Each grid in a multiverse is indexed using a tuple [r,c], where r is the row index, and c is the column index. $0 \leq r <$ `R` and $0 \leq c <$ `C`. The indexing starts at [0,0] from the top left corner of the multiverse, similar to that of the grid cells in the grid, as shown in Figure 1.
5. A "live" grid is a grid whose state is such that it has more than 50% of its grid cells as "live". Otherwise the grid is said to be in "dead" state.

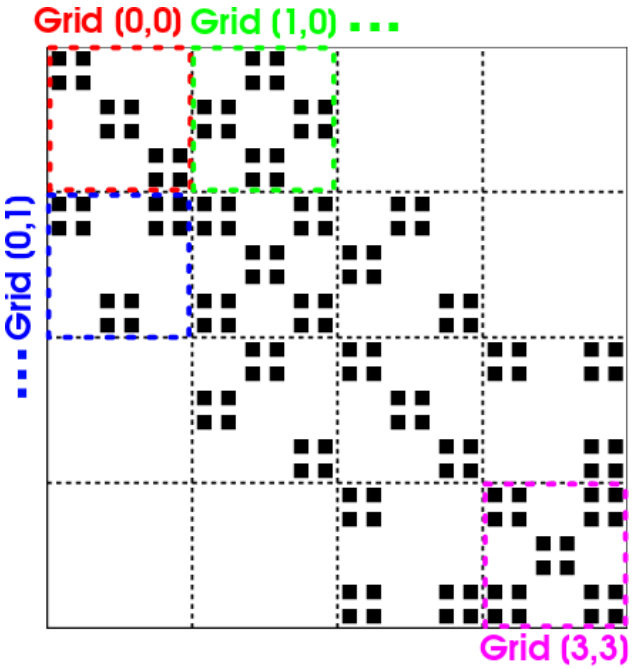

Figure 1: Output of an example of multiverse showing the positions of grids to indicate the format of output. The multiverse we cover in this assignment is however, different from the one in this figure, as in our case the boundary grid cells in each grid cannot be live. For this figure, black may be considered live, and white, dead cells.

6. Each of the grids are *valid* only if all of its boundary cells are in "dead" state.
7. A *synchronization event* is where the multiverse gets updated.
8. All the grids in the multiverse are simulated for a specific number of generations, say `k` generations. As we did in Lab-03, each grid progresses to next generation only if the new generation of the grid maintains the validity of the grid. Or else, the simulation of the grid terminates and the grid maintains its last valid configuration.
9. A **synchronization event** in a multiverse occurs analogous to a single simulation of game of life in a single grid. That means the synchronization event has two steps – the first step is to determine the state of each grid in the multiverse. In the second step, the state of each grid is updated as per the same rules as applicable to grid cells within each grid.
10. **Updating the state of the grid** means ensuring that the grid is in the state it is supposed to be, as per its neighborhood.
    - If a grid has to be converted from a "live" to a "dead" state, then change the state of enough number of randomly chosen "live" grid cells to "dead" state, so that the total number of "live" grid cells is 45% of all cells.
    - If a grid has to be converted from a "dead" to a "live" state, then change the state of enough number of randomly chosen "dead" grid cells to "live" state, so that the total number of "live" grid cells is 55% of all cells.
    - No change must be done to the grid if the grid is maintaining either its "live" or "dead" state.

11. Assume that the initial state of the multiverse is `synchronization-event-0`, which is the value of the attribute of *age* of the multiverse .

12. At the end of the i^{th} synchronization event, the age of the multiverse is `synchronization-event-i`.

13. The above model of the multiverse-grid uses *composition*, i.e. *the grid is a part-of the multiverse*.

14. The **simulation** of the multiverse is a sequence of *initialization equivalent to* `synchronization-event-0`, `k` simulations of each grid, `synchronization-event-1`, `k` simulations of each grid, ... `synchronization-event-S`, for `S` synchronization events in the multiverse.

15. The output of the multiverse is the same function as the output of the grid, except that instead of outputting grid cells, one would output grids. The expected outcome is that grids are in the form of matrix. Hence, care must be taken in calling appropriate grid cells to be printed in each row. Figure 1 shows how a multiverse(4,4) is printed out.

Assignment

The objective of this assignment is to write a C++ program, `multiverse_game_of_life.cpp`, which simulates the multiverse.

1. The user is expected to give inputs through command line, four integers, `R`, `C`, `k`, and `S`,

   ```
   ./multiverse_game_of_life R C k S
   ```

   which are number of rows in each grid, number of columns in each grid, number of generations between two consecutive synchronization events, and number of synchronization events, respectively.

2. The initial state of the multiverse is that of the chess board, where white cell implies "live" grid, and black cell is the "dead" grid. The grid of index (0,0) is white. Generate the multiverse where the grids are generated using the rules that:

   - the "live" grid must have 55% of the grid cells "live", and
   - the "dead" grid must have 45% of the grid cells "dead".

3. Your task is to print out the multiverse on the terminal console, when its age is `synchronization-event-S`.

Note: All %-ages used for computing state ("live" or "dead") of the grid may use the floor value for integer values.