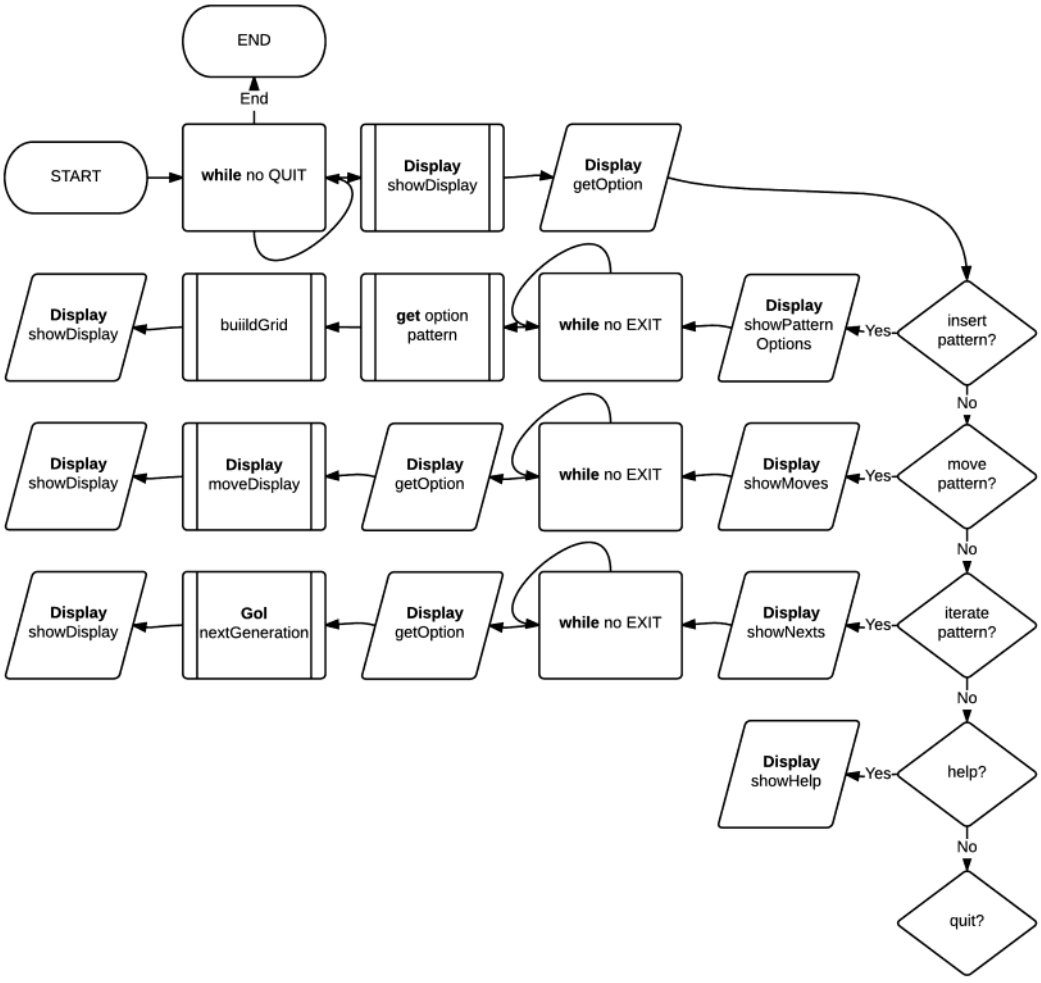
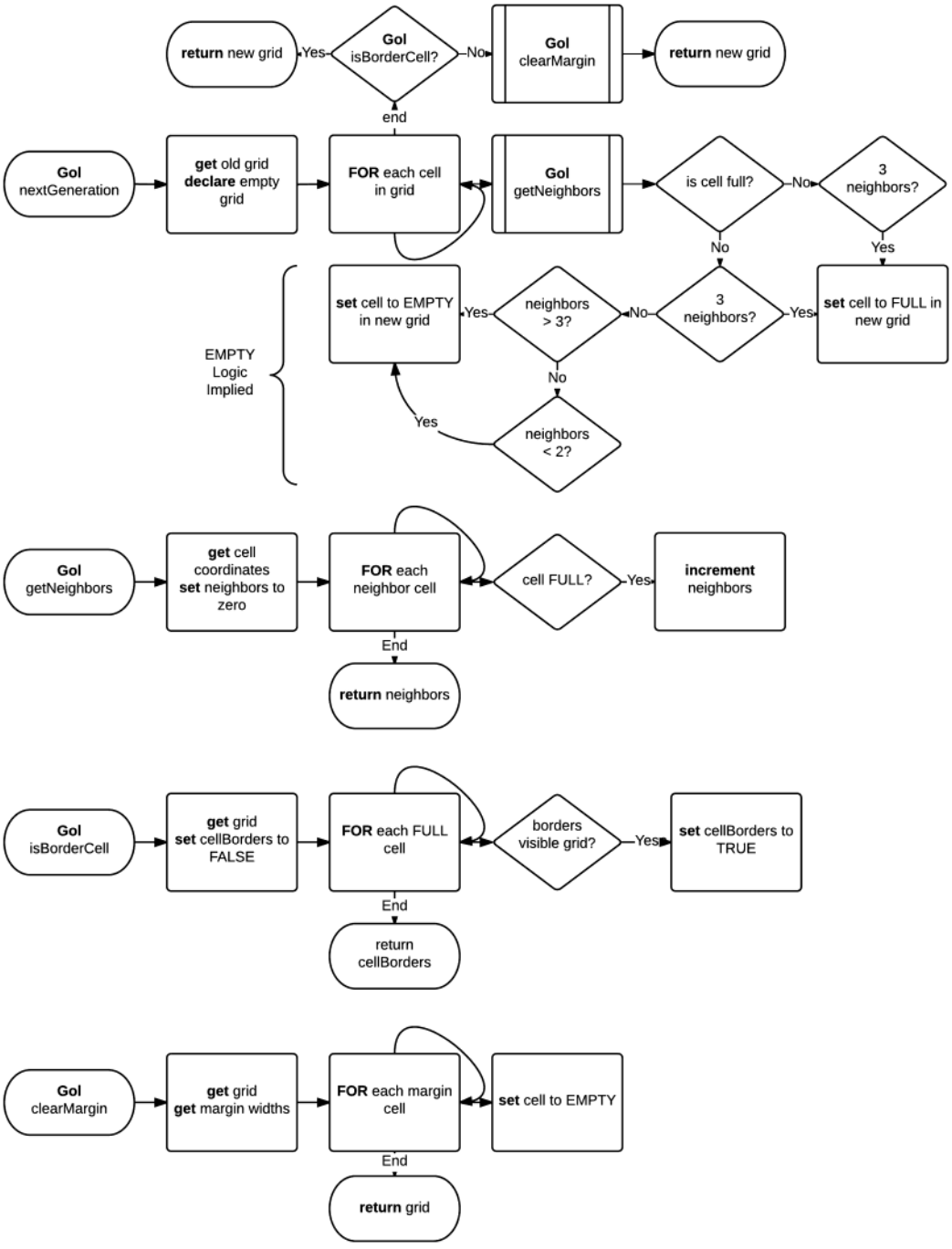
[1] Assignment (Reflection)

Saturday, April 04, 2015 12:15 AM

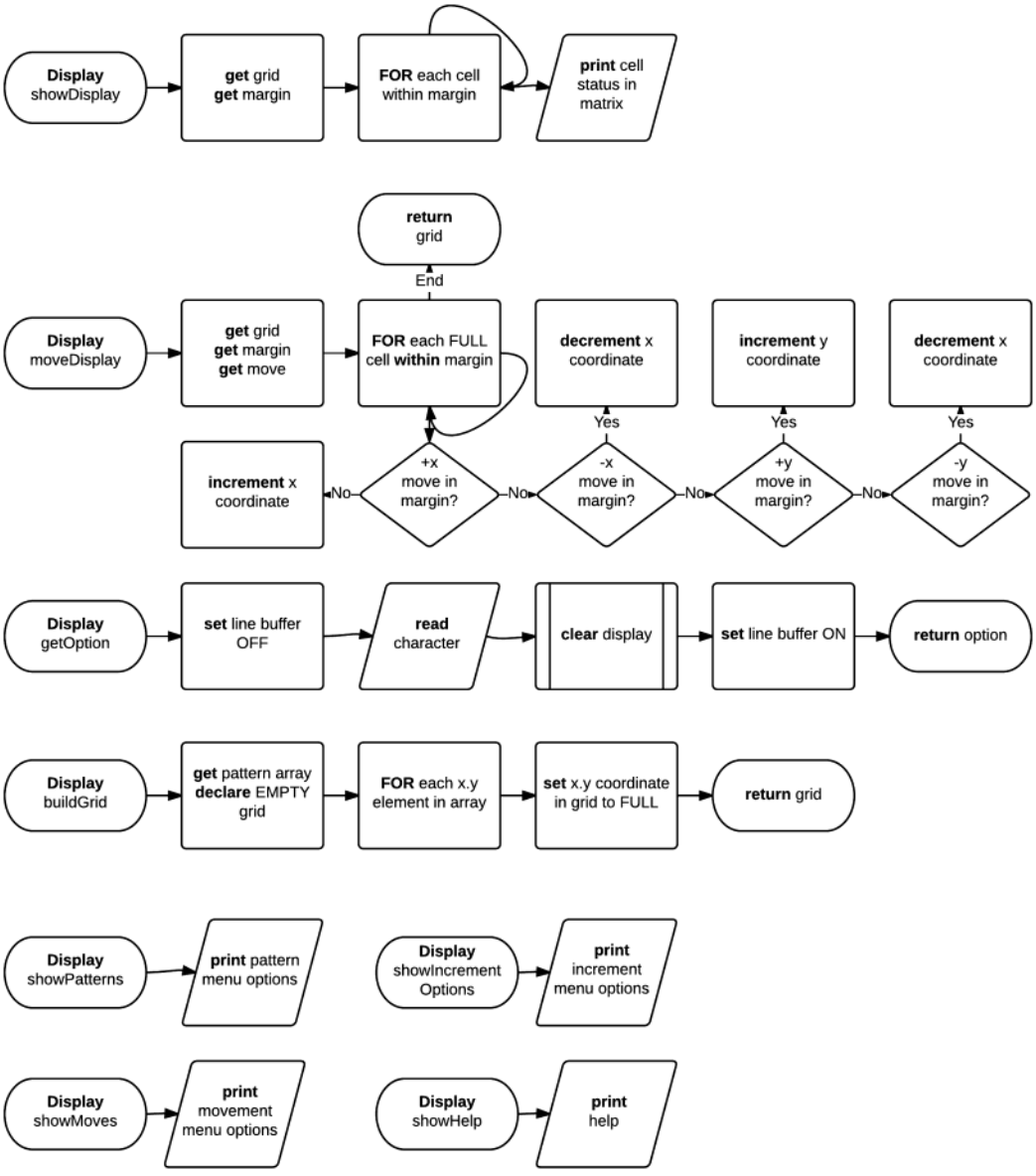
- Design
  - Classes/Files
    - Display - manipulation and display of viewable game of life grid
    - GoI - contains game of life logic
    - Menu - interfaces GoI and Display with user control
  - Flow Charts
    - Menu



    - GoI


return new grid
Yes
GoI isBorderCell?
No
GoI clearMargin
return new grid
end
GoI nextGeneration
get old grid declare empty grid
FOR each cell in grid
GoI getNeighbors
is cell full?
No
3 neighbors?
No
Yes
EMPTY Logic Implied
set cell to EMPTY in new grid
Yes
neighbors > 3?
No
3 neighbors?
Yes
set cell to FULL in new grid
No
Yes
neighbors < 2?
GoI getNeighbors
get cell coordinates set neighbors to zero
FOR each neighbor cell
cell FULL?
Yes
increment neighbors
End
return neighbors
GoI isBorderCell
get grid set cellBorders to FALSE
FOR each FULL cell
borders visible grid?
Yes
set cellBorders to TRUE
End
return cellBorders
GoI clearMargin
get grid get margin widths
FOR each margin cell
set cell to EMPTY
End
return grid

- Display



- Test Plan & Results

Test	Plan	Result
Display	Display turns on	Yep
Pattern Addition	Pattern coordinates added by hard-coded arrays	Yep
Pattern Movement	Patterns move within display whenever	Yep
Oscillator	Oscillator oscillates	Yep
Glider	Glider glides off screen smoothly without any funny business	Yep
Glider Gun	Glider gun shoots gliding gliders off the display indefinitely	Yep
Menu Options	Provides optimal user usability	Hopefully

- Deep Reflections

- Alterations
  - Much of the Display design was an alteration in and of itself. While design for the GoI class was aforementioned in flow chart form, not knowing the appropriate techniques needed to bring the intended Display to fruition led to straight coding and google searches, only latter leading to the design when knowing what was possible (e.g. ncurses and stty) and how to do it (cool stuff), then back to coding.
- Learnings
  - Realized linux terminal commands for capturing user input without line buffering
    - e.g. http://stackoverflow.com/questions/421860/capture-characters-from-standard-input-without-waiting-for-enter-to-be-pressed)
  - Introduced to ncurses library for capturing user input while looping
    - e.g. http://math.hws.edu/orr/s04/cpsc225/curses.html)
  - Utilized "gdb" to quickly find core-dumping functions from dereferencing non-dereferencable pointers.
    - e.g. http://stackoverflow.com/questions/2876357/determine-the-line-of-c-code-that-causes-a-segmentation-fault
- Failed Tests
  - Many failed tests occurred on programming the display in order to refresh on user command without line buffering for stop and play control of generating generations in the game of life. A common theme involved having "tabbed" line returns staggering my display to the right, resolved by clearing the screen just after reading input.
  - Attempted to hard-code array into classes without practical success. Arrays were hard coded into a heap memory as a function in Menu.
  - Got a single cell instead of three for the oscillator due to an incorrect size of the array when trying to use sizeof() for the determination. Pattern array size was simply hard-coded along with the array without expending any further neural glycogen.
  - Planned to fail testing for evolving patterns interacting with the display edges since it wasn't programmed in, as suggested in the assignment. A simple resolution of searching for any bordering cells and deleting the unseen game of life grid in the absence of any brought much relief compared to the alternative ideas in how to handle the concept of "infinity" required by the specifications.
  - A lot of segment faults also occurred due to the nature of dereferencing bad pointers. Most of this null dereferencing occurred when trying to initially move the pattern on display. "gdb" significantly aided the insanity.
  - Off course the cacophony of minor misplaced, mistyped, mishandled, mis-minded, and unsexy syntax joined in the ensemble of the other major mishaps already discussed.
- Regrets
  - I wish I had a better handle of the program structure in the design phase. Knowing the basic idea behind deconvoluting classes from the second week by stripping nouns from a narration of the problem space should ameliorate things considerably.
- Problem Solving Suggestions
  - The biggest problem solving virtue for the assignments objective was probably persistence. I never really ever considered persistence to be a key factor in problem solving before, but now it utterly makes complete sense that it is. In fact, I think it is the greatest attribute to problem solving, helping bridge the gap between what was seen in brain and what was being seen in the terminal (i.e. ncurses and stty manipulation). Though the idea of using untaught code and libraries may have been intimidating, using curiosity and simply chilling out to drive persistence galvanized in a steady and workable understanding of the needed Lego pieces.