Recitation 01

Focus

- strings, vectors, functions and I/O

Task

Some recitations, aka *labs*, will follow a "tutorial" style. Others, such as this one, will present you with a program to write. Regardless of the style of the assignment:

- Feel free to ask questions! Check the course notes! Even talk to your classmates! The purpose of the labs is for you to get hands on experience and produce good working code.
- If you are done before the end of the lab, ask a lab instructor to check over your code. It is very possible that even if your code works, the instructor may point out areas where your code could be improved, either to make it more readable, or because your code might have failed for other input.
- When you are done, either because you have shown it to a lab instructor and had it "checked off" or if you needed to finish it up at home, *submit* it on NYU Classes.

And now on to the description of today's task.

- Implement Conway's Game of life
- The rules of life say:
  - A living cell with two or three neighboring living cells survives into the next generation. A living cell with fewer than two living neighbors dies of loneliness and a living cell with more than three living neighbors will die from overcrowding.
  - Each empty/dead cell that has exactly three living neighbors-- no more, no fewer-- comes to life in the next generation.
- Input:
  - Get the initial state from a file, named "life.txt" in the current working directory.
  - Each row will have dashes for dead cells and asterisks for live ones.
  - To be nice, I will make all lines the same length.
- Output to standard output (i.e. the screen):

    - Display the original state of the world, as read in from the file and 10 additional generations.
    - In each generation, each row will have spaces and asterisks. (spaces look nicer than dashes)
        - Spaces represent empty/dead cells
        - Asterisks represent occupied/living cells
- Suggestion: For your internal representation, create an extra row on the top and bottom as well as an extra column to the right and the left of the actual data. Mark the extra cells as dead. This will simplify your code, as you won't need special checks for "boundary" cells.
- Sample input and output for the program are attached
- Good programming style:
    - Naming
    - Use of functions
    - Length of lines and functions
    - Comments are not required for labs *except b*e sure there is a comment at the top of the file identifying the author (you, hopefully), the name of the program file, and a [very] brief description of the program.
        - Note that all of your submissions for the semester should have such a comment at the beginning of each code file.

Submit

- Whether or not you have been "checked off" during the recitation, submit a single source file, `rec01.cpp`, on NYU Classes. If you do not, your grade may be penalized. Do *not* submit any other files, such as those generated for your development environment (IDE).

What you need to know

Since you are just starting out with C++ and with this course, I thought it might be handy to have links back to the course lecture notes on the necessary material. It is possible for this recitation that some item(s) may not have been covered in your lecture section. Hopefully, given your prior programming experience, none of these issues will seem complicated, but please feel free, as always, to ask for help from your lab instructors or your fellow students.

- Defining variables, using conditions and looping
- How to open a file, check that it opened correctly, read and close it
- How to define / use a vector

- How to use strings
- Ideally writing functions. If you have covered this, then certainly break the problem up into appropriate functions. However, if you haven't gotten to that in lecture, as happens in the fall semester, don't worry about it.