CMSC 201 Fall 2017

Project 2 – Game of Life

Assignment: Project 2 – Game of Life
Due Date:
Design Document: Friday, November 3rd, 2017 by 8:59:59 PM
Project: Friday, November 10th, 2017 by 8:59:59 PM
Value: 80 points

Collaboration: For Project 2, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

For Project 2 you will have to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberate construction of your program and how it will work, rather than coding as you go along, or starting without a plan.

Instructions

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on manipulating lists, calling functions, and handling mutability appropriately.

The design for Project 2 is entirely up to you – suggestions are provided within the project description, but you are not required to use them.

At the end, your Project 2 file must run without any errors.
It must also be called proj2.py (case sensitive).

Additional Instructions – Creating the proj2 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 2 files. We recommend calling it **`proj2`**, and creating it inside a newly-created directory called **`Projects`** inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

Project 2 is designed to give you practice with two-dimensional lists, mutability, and creating and calling functions. You'll need to use practically everything you've learned so far, and will need to do some serious thinking about how all of the pieces you need to create should fit together.

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Task

For this project, you will be coding a simple cellular automata game, called Conway's Game of Life. In this game, you have a grid where pixels can either be on or off (dead or alive). In the game, as time marches on, there are simple rules that govern whether each pixel will be on or off (dead or alive) at the next time step.

Coding Standards

Prior to this assignment, **you should be familiar with the entirety of the Coding Standards**, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

You should be commenting your code, and using constants in your code (not magic numbers or strings).
Any numbers other than 0 or 1 are magic numbers!
Any strings with a meaning (*e.g.*, alive or dead) should be constants!

You will **lose major points** if you do not follow the 201 coding standards.

If you have questions about commenting, whitespace, or any other coding standards, please come to office hours.

Additional Specifications

For this assignment, you must create and call **at least five individual functions**, not including `main()`. All other design decisions are up to you.

You may *not* import any libraries or use any library functions! Doing so will lose you a *very* large number of points.

Input Validation

For this project, we will require that you validate input from the user. You can assume that the user will enter the right type of input, but not that they will enter a correct value. In other words, a user will always give an integer when you expect one, but it may be a negative or otherwise invalid value.

You will need to validate the following things:

- Getting the numbers of rows and columns
  - Rows must be 1 or greater
  - Columns must be 1 or greater
- Getting the cells that will start the game off as "alive"
  - The row must be either
    - The character "q" (for quit)
      - *If the user enters any other character, it is fine for the program to crash*
    - A valid index for the number of rows your board has (*i.e.*, it starts at index 0, and goes up to index row_size – 1)
  - The column must be
    - A valid index for the number of columns your board has (*i.e.*, it starts at index 0, and goes up to index column_size – 1)
- Getting the number of iterations to run
  - Must be 0 or greater
    (They can choose to run no iterations, in which case you would only print the starting board)

See the sample output for examples of how this should work in your program.

Details

For this project, you will be coding a simple cellular automata game, called Conway's Game of Life. In this game, you have a grid where pixels can either be on or off (dead or alive). In the game, as time marches on, there are simple rules that govern whether each pixel will be on or off (dead or alive) at the next time step. These rules are as follows:

Any **live** cell with fewer than two live neighbors dies.
Any **live** cell with two or three live neighbors lives on to the next generation.
Any **live** cell with more than three live neighbors dies.
Any **dead** cell with exactly three live neighbors becomes a live cell.

"Live" cells are to be represented with the character "`A`" (capital 'a'), and "dead" cells are to be represented with the character "`.`" (a period).

To begin, you will ask the user for the size of the game board (rows first, then columns). Next prompt them for any cells they would like to be "alive" when the game begins. Finally, ask the user how many iterations of the game (number of time steps) they would like to see run. You should then display these iterations, showing each iteration separately.

Hints and Advice

It would be a good idea to:

- Store your board in a 2D list (make sure you don't mix up column and row!)
- Have a function called `nextIteration()` that takes the current board in as a parameter, and that returns a new board with the next iteration
- Have a function called `printBoard()` that takes in the current board as a parameter and prints out the board's contents

TIP: This would be a very good time to use incremental programming! Incremental development is when you are only working on a small piece of the code at a time, and testing that the piece of code works before moving on to the next piece. This makes it a lot easier to fix any mistakes.

Project

The project is worth a total of 80 points. Of those points 10 will be based on your design document, 10 will be based on following the coding standards, and the other 60 will be based on the functionality and completeness of your project.

Design Document

The design document will ensure that you begin seriously thinking about your project early on. This will not only give you important experience doing design work, but will help you gauge the number of hours you'll need to set aside to be able to complete the project. **Your design document must be called design2.txt.**

For Project 2, you are creating the design entirely on your own.
You **may NOT work with another student** to "brainstorm" a solution or discuss any general approaches or requirements. If you need assistance with the design document, come to office hours.

Your design document must have four separate parts:

1. A file header, similar to those for your assignments
2. Constants
    a. A list of all the constants your program will need, including a short comment describing what each "group" of constants is for (*e.g.,* menu options, meaning of indexes, etc.)
3. Function headers
    a. A complete function header comment for each function you plan to create, including the name, description, inputs, and outputs
4. Pseudocode for main()
    a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under the relevant comment (if applicable)

Your design can follow the same general format as the design for Project 1.

Your **`design2.txt`** file will be compared to the **`proj2.py`** file that you submit. Minor changes to the design are allowed. A minor change might be the addition of another function, or a small change to **`main()`**.

Major changes between the design and your project will lose you points. This would indicate that you didn't give sufficient thought to your design.
(If your submitted design doesn't work, it is generally better to lose the points on the design, and to have a functional program, rather than turning in a broken program that follows the design. The decision is ultimately up to you.)

To submit your design document, use

```
linux1[4]% submit cs201 PROJ2_DESIGN design2.txt
Submitting design2.txt...OK
linux1[5]%
```

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard, and is called "sample2.txt".

(Yours does not have to match the sample output exactly, but it should be similar.)

Submitting

Once your **`proj2.py`** or **`design2.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the design or project in multiple times, as you reach new milestones or complete each piece. To do so, run **`submit`** as normal.)

To submit your design file (which is due Friday, November 3rd, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ2_DESIGN design2.txt
Submitting design2.txt...OK
linux1[5]%
```

To submit your project file (which is due Friday, November 10th, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ2 proj2.py
Submitting proj2.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**