CS14 – Lab 2

Warmup. From now on, at the beginning of lab you will have a warmup exercise from Project Euler (https://projecteuler.net/). This website has a list of problems designed to hone your basic programming skills. There are solutions for all of these problems available on the internet; however, I highly recommend you do not look for them. You need to learn to solve these types of problems on your own.

1. (a) Write the FizzBuzz program using loops:

   > Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

   (b) Rewrite the FizzBuzz program recursively.

   (c) Read http://www.codinghorror.com/blog/2007/02/why-cant-programmers-program.html about how this is a common interview question that many people fail.

Exercises: For the next few labs, you will be implementing Conway's Game of Life. You can find a complete description of the rules on Wikipedia (https://en.wikipedia.org/wiki/Conway's_Game_of_Life). We will use different data structures in each lab. Every structure has its own strengths and weaknesses. In this lab, you will be implementing Life using a matrix (i.e. a vector of vectors) data structure to store the game. Follow these steps:

1. Every game has a starting position that comes from a file. You have been provided with three of these files: “blinker,” “glider,” and “pulsar.” You have also been given a working version of the game. Test the game by running:

   ```
   ./conway-matrix blinker 100000
   ```

   The file format for the game is a grid of spaces. Each space will be displayed in the console as a gray square. Wherever there is an 'x' instead of a space, that position will be marked with a red square. Your first task is to create a new file for any of the patterns shown on wikipedia's page. Then run conway-matrix on it and watch the pattern change. (Try creating your own patterns too. It's fun!)

2. Next, read through the code in the GameBoard and Matrix board classes. Implement the GameBoard::iterate() function. Your program should work fully if you've done this properly.

3. Implement the ToroidBoard class. This game board has no edges at all. Instead, the game board “wraps around.” The conway-toroid executable gives an example. Use the glider file and watch the glider travel over the edges of the board.

4. Modify the main function so that your program takes an additional, non-optional argument that specifies which game board to use (i.e. MatrixBoard or ToroidBoard).