ISE 108 Midterm 1 — Additional Practice Problems

1. **Field of Squares (20 points)**

The Processing sketch “problem_1” displays a large number of squares in a grid pattern. Modify the program so that pressing one of the following keys moves the entire grid of squares 5 pixels in the appropriate direction. Hint: use the two helper variables (`xOffset` and `yOffset`) for this purpose.

Key	Grid Moves This Way
i	Up
j	Left
k	Right
m	Down

2. **Debugging (20 points)**

The "DebuggingOne" program displays a simple simulated traffic light. When the program starts, the light is green. As the user clicks in the window, the light will turn yellow, then red, then green again (and keep going, changing color on ***every*** mouse click). Unfortunately, this program has a series of bugs in it, from syntax errors to logical errors. Fix the problems so that this program compiles and runs correctly. Note that, sometimes, fixing one syntax error may cause Processing to find a few more syntax errors, so don't be discouraged as you fix the bugs. Eventually, the syntax errors will end, and you'll be left with logical errors. There are a total of 5 errors in this code.

3. **Classes (20 points)**

The "LinearEquation" program contains a partial class definition for a class named `Linear` that represents a 2 x 2 system of linear equations:

ax + by = e
cx + dy = f

a. Complete the constructor, which assigns its arguments, in order, to the variables a, b, c, d, e, and f.

b. Complete the `isSolvable()` method, which returns `true` if ad - bc is not 0, and `false` otherwise.

c. Complete the `getX()` method. If the system of equations is solvable, return the solution

   x = (ed - bf) / (ad - bc)

   or -99999 otherwise.

d. Complete the `getY()` method. If the system of equations is solvable, return the solution

   y = (af - ec) / (ad - bc)

   or -99999 otherwise.

4. **Functions (3 points)**

   Define a function named `addUp()` that takes two integer arguments: *lo* and *hi*. This function returns an integer value that is the sum of all the integers from *lo* to *hi* (including *lo* and *hi*). For example, `addUp(3,6)` would return the integer value 18 (because 3 + 4 + 5 + 6 adds up to 18). Write a small Processing sketch that tests your function (your sketch does not need to collect any user input for your test; it is sufficient to call `addUp()` with test values of *lo* and *hi* from `setup()`). You may assume that the first argument (*lo*) is always less than or equal to the second argument (*hi*).

5. **Extra Credit (20 points)**

   Complete the Processing sketch "extra_credit", which contains a function named `fizzBuzz()`. This function prints out the numbers from 1 through n, one per line. For each value that is evenly divisible by 3, the function prints “fizz” on the line after the number. For each value that is evenly divisible by 5, the function prints “buzz” on the line after the number. If a value is evenly divisible by 3 and by 5 (for example, 15 or 30), the program prints both “fizz” and “buzz” on that line. **Hint:** Use the modulo (%) operator to test for divisibility.

   For example, `fizzBuzz(4)` would produce the following output:

   1
   2
   3fizz
   4