# CSE 1321L: Programming and Problem Solving I Lab

## Assignment 6 – 100 points

## Classes

**Program 1:** WAKA, WAKA. Pac-Man was a big hit back in the 80s. One of the things he could do was "teleport" from one side of the screen to the other, and that's what we're going to implement here. For this program, you're going to write a class that only has two attributes: an X and Y location. Imagine the player is on a 10x10 grid. If the player exceeds the bounds of the screen, their location teleports to the other side of the screen (e.g. far-right players go to the far-left). The class should include methods for going up, down, left and right. Finally, you should make a "driver" that brings an instance of this class to life in the middle of the "screen" and enables the player to move up, down, left and right. Design (pseudocode) and implement (source code) this program.

Sample run

```
Current location – X: 5 Y:5
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
U
Current location – X : 5 Y :4
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
U
Current location – X : 5 Y :3
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
U
Current location – X : 5 Y :2
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
U
Current location – X : 5 Y :1
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
U
Current location – X : 5 Y :0
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
U
Current location – X : 5 Y :9
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
R
Current location – X : 6 Y :9
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
R
Current location – X : 7 Y :9
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
R
Current location – X : 8 Y :9
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
R
Current location – X : 9 Y :9
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
R
Current location – X : 0 Y :9
(U)p, (D)own, (L)eft, (R)ight or (Q)uit:
Q
```

**Program 2:** BACK AWAY FROM THE TV! If you sat close enough to an old TV, you could see individual (R)ed, (G)reen and (B)lue (or RGB) "phosphors" that could represent just about any color you could imagine (using "additive color model"). When these three color components are at their maximum values, the resulting color is white from a distance (which is amazing – look at your screen with a magnifying glass!). When all are off, the color results in black. If red and green are fully on, you'd get a shade of yellow; red and blue on would result in purple, and so on. For computers, each color component is usually represented by one byte (8 bits), and there are 256 different values (0-255) for each. To find the "inverse" of a color (like triple-clicking your iFone® button), you subtract the RGB values from 255. The "luminance" (or brightness) of the color = (0.2126*R + 0.7152*G + 0.0722*B).

For this program, you need to design (pseudocode) and implement (source code) a Color class that has R, G and B attributes (which can be ints). The constructor should take three parameters representing the initial color of (R:254, B:2, G:100). You should include 6 methods to increase and decrease each component, not to exceed 255 or be less than 0. You should include a method (usually called "toString") that returns a string representing the current values for each component as well as the luminance. Finally, you should include a method that calculates and prints the inverse color.

Next, create a "driver" (or main) that creates a default color, prints its values to the screen, and enables the user to increase/decrease values as well as print the inverse. It should behave like below.

Sample run:

```
R:254 G:2 B:100 L:62.6508
Do you want to:
1) Increase Red, 2) Decrease Red
3) Increase Green, 4) Decrease Green
5) Increase Blue, 6) Decrease Blue
7) Print the inverse
or 8) Quit
1
R:255 G:2 B:100 L:62.8634
Do you want to:
1) Increase Red, 2) Decrease Red
3) Increase Green, 4) Decrease Green
5) Increase Blue, 6) Decrease Blue
7) Print the inverse
or 8) Quit
1
R:255 G:2 B:100 L:62.8634
Do you want to:
1) Increase Red, 2) Decrease Red
3) Increase Green, 4) Decrease Green
5) Increase Blue, 6) Decrease Blue
7) Print the inverse
or 8) Quit
7
Inverse is R:0 G:253 B:155
R:255 G:2 B:100 L:62.8634
Do you want to:
1) Increase Red, 2) Decrease Red
3) Increase Green, 4) Decrease Green
```

```
5) Increase Blue, 6) Decrease Blue
7) Print the inverse
or 8) Quit
8
R:255 G:2 B:100 L:62.8634
```

## Submission:

### Part 1: Pseudocode:

1. Review the assignment submission requirements and grading guidelines.
2. Upload the pseudocode files (Word doc or PDF) to the assignment submission folder in D2L.
3. The files must be uploaded to D2L by the due date.
4. The Pseudocode must be complete and following the standards listed at http://ccse.kennesaw.edu/fye/Pseudocode.php

### Part 2: Source Code:

1. Review the assignment submission requirements and grading guidelines.
2. Upload the source code files to the assignment submission folder in D2L.
3. The files must be uploaded to D2L by the due date.