CSE 1321L: Programming and Problem Solving I Lab
Assignment 7 – 100 points

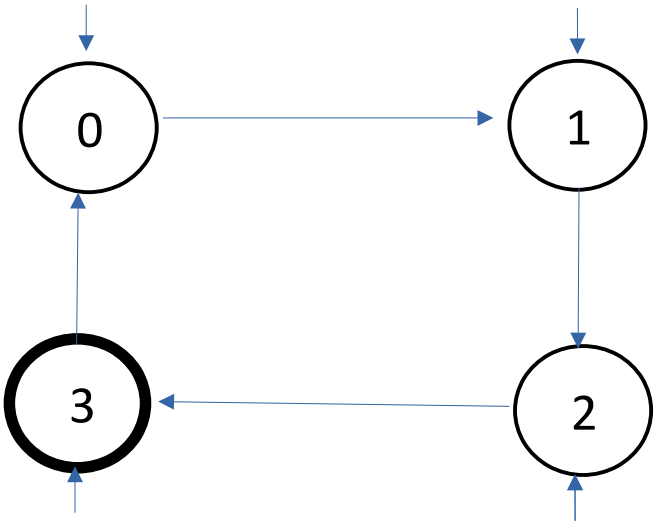
Object Oriented Programming and Classes

What students will learn:

1) Designing classes including attributes, constructors and methods
2) Creating objects of those class types

Overview: The programs you've written up to this point have started at “main” and basically executed from "top to bottom". In the last assignment, functions/methods were introduced, where you learned that the execution of code could jump from main, to a function, then back to main. For this assignment, we're going to practice OOP, or Object-Oriented Programming. OOP is a completely different way to think about programming. The concept originated in the 1960's, but it wasn't widely adopted until the mid 1980's.
So, what is it? At its core is the concept of a "class" which is simply a **grouping of variables and methods**; the behavior of those methods usually change based on the state (also known as values) of the variables.

Assignment 7A: *Finite State Automata.*
In computer architecture (a class you might take in the future) we teach a concept called “Finite State Automata”. Also known as “State Machines”, these models show different states of a system and how you can move from one to other. We are going to simulate one using objects. First, review the following diagram:



You will create a **FSA** class with one attribute: int state. The valid states are shown in the diagram above. It should also have the following methods:

- overloaded constructor: Takes in one integer for the state attribute. If the state is not one of the values shown, it should be initialized to 0 and the constructor should print out “This is an invalid state. Starting at state 0”

- *int goToNextState()*: Takes in no parameters and advances the state variable to the next state based on the diagram above. It should then return the state variable. **It should not print anything**.

- *bool/Boolean end()*: Takes in no parameters. It returns TRUE if the state is currently 3, and returns FALSE otherwise.

You will then create a separate class (and file), **Assignment7A**. In its main method, you should do the following:

- Prompt the user to enter a state

- Create a new FSA object using the overloaded constructor

- Create a loop and prompt the user with two choices
  - "Go to next state" – This should call the goToNextState() function on the FSA object and print the current state based on the value returned from the function
  - "End" – This should call the end() function. If the returned value is TRUE, end the loop and program. Otherwise, tell the user they can only end the program at state 3.

Sample Output:

```
[Finite State Automata]
What state do you want to start at? 1

What will you do?
>Go to next state
>End
Go to next state

You're now at state 2.
What will you do?
>Go to next state
>End
END

You can't stop here; you can only stop at state 3.

What will you do?
>Go to next state
>End
Stop

[Invalid command!]

What will you do?
>Go to next state
>End
Go to NEXT state

You're now at state 3.
```

```
What will you do?
>Go to next state
>End
End

The FSA has shut down.
```

Assignment 7B: *Art Program!* For our last assignment in CSE 1321L, we'll learn how to create an image. Most images are saved in binary format – this means they are not human readable without special software like a hex editor. However, a few are saved in ASCII/text mode, and you can read and edit them with just a text editor. One such type is the Portable Pix Map image format. You will create a program that generates the contents of a PPM file for a square image (we'll learn how to actually save a file in CSE 1322).

You will create a **Pixel** class which has three attributes:

- red: int
- green: int
- blue: int

You will create a default constructor that initializes those values to 255, and an overloaded constructor that takes user input to assign the values. The class will also have the following functions:

- **changeRGB** (): Takes in three integers to update the red, green, and blue attributes. Returns nothing.
- **printRGB** (): Takes in nothing. Prints the red, green, and blue attributes in order with a single space in-between each value. Returns nothing.

In a separate Assignment7B driver class, do the following:

- Prompt the user to enter a width and height value
- You will create a 2D array of Pixel object based on the width and height. *Don't be scared!* This is similar to creating a 2D array of strings.
- Prompt the user for the initial fill color – this will be used to initialize all the Pixel objects in the 2D array to the same RGB
- Repeatedly prompt the user to make one of the following choices:
    - Print the PPM file to the screen. Note that every valid PPM file starts with a "header", which looks like this:
      P3
      <WIDTH VALUE> <HEIGHT VALUE>
      255
      <PIXEL VALUES>
      The width and height values should be what the user entered, and the pixel values should come from the 2D array's objects.

- Change a pixel to a new color. Prompt the user for the pixel's coordinates and the new RGB value, then update the Pixel object at that location in the array. Refer to your prior work with the level map creator – the logic is similar.
- Change a line of pixels to a new color. Prompt the user for the starting coordinates, the length, and the new RGB value. Then, update the colors of all Pixel objects in that horizontal line. Refer to your prior work with the level map creator – the logic is similar.
- Quit. Print the final image to the screen and stop the program.

If you're interested in actually seeing the image itself, check the appendix of this assignment.

Sample Output:

```
[Portable Pix Map Art Program]
Enter an image width: 3
Enter an image height: 4
Enter the fill color's red value: 21
Enter the fill color's green value: 25
Enter the fill color's blue value: 200

What will you do?
1) Fill in a pixel
2) Fill in a line
3) Print the image
4) Quit
Choice? 3

P3
3 4
255
21 25 200 21 25 200 21 25 200
21 25 200 21 25 200 21 25 200
21 25 200 21 25 200 21 25 200
21 25 200 21 25 200 21 25 200

What will you do?
1) Fill in a pixel
2) Fill in a line
3) Print the image
4) Quit
Choice? 1

Row: 2
Column: 1
New Red Color: 123
New Green Color: 98
New Blue Color: 5

What will you do?
1) Fill in a pixel
2) Fill in a line
3) Print the image
4) Quit
Choice? 2
```

```
Row: 0
Column: 0
Length: 3
New Red Color: 6
New Green Color: 78
New Blue Color: 15

What will you do?
1) Fill in a pixel
2) Fill in a line
3) Print the image
4) Quit
Choice? 4

P3
3 4
255
6 78 15 6 78 15 6 78 15
21 25 200 21 25 200 21 25 200
21 25 200 123 98 5 21 25 200
21 25 200 21 25 200 21 25 200
Closing...
```

Submission:

1. You will submit all required object and driver class code files
2. File names and class names must be correct.
3. Upload all files (simultaneously) to the assignment submission folder in Gradescope.

APPENDIX

To actually see your image, copy and paste the contents of the PPM file (starting with the P3 line) into a text editor like Notepad or TextEdit.

Save it with a .ppm extension (making sure to save as “All Files” rather than as a Text file). If you did this correctly, the file should have a blank icon (or at least, not look like a text file).

You can then view the image by dragging it onto this page:
http://paulcuth.me.uk/netpbm-viewer/

If you’d like to edit it further and make your own PPM files, you can use image software like the free GNU Image Manipulation Program.