Homework 1 Report

Zack Fravel
010646947

Problem Statement:

The goal of this programming assignment was to create a C++ program that's able to take an uppercase character input and output the coordinates for a geometric model of that letter. Along with this, the user is also able to input scale information to determine the bounds of the letter in the X and Y directions. The assignment was to produce 2D coordinates for five letters in the English alphabet.

My program specifically is designed to handle the letters L, U, I, T, and C. There are a few areas where error handling needs to be performed, such as entering a character other than an uppercase letter as well as an uppercase letter who's coordinates are not saved within the program. Along with that I also made sure to validate for each size input. To reiterate, the program takes in an uppercase letter and outputs that letter's coordinates when represented as a 2D polygon.

Design:

There were a number of design decisions to make along the way. Again, my program handles the characters L, U, I, T, and C. I then had to figure out how I wanted to store the sets of coordinates for each letter. I decided to settle on creating a Point class and creating an array of Point objects to manipulate and print. The point class has a constructor which sets x and y equal to 0.0, along with set/get functions. The scaleCoordinates() and printCoordinates() functions use the set() and get() functions from the point class to pass information around. The pro's of this design is that it's modular so it will be very easy to add new functionality later on. The only con's I can think of in my design are maybe it uses more memory than it would if fully optimized because of the way I initialize my arrays.

The main program asks the user which character they'd like to input. After receiving an input, the program then runs input validation and recognition. Once a stored input is recognized, within the if-statement I set the number of sides of the polygon (n) and set the array of point objects to the respective points. Each letter is designed in a cell who's domain/range is X: [0, 7] Y: [0, 10]. The program then runs scaleCoordinates() on the array corresponding to the user's size inputs. Finally, the program runs printCoordinates(), which prints the model to the console.

Implementation:

The implementation process was pretty simple and follows the description laid out above in the design section. I started from scratch by implementing the user inputs and made sure those were working first. I then implemented the point class with a print() method and the array of point objects to make sure the data could be correctly put into the array. In the main validation for the user input I set the number of sides (n), which is sent

to the print() method at the end. At this point I could input a letter and output the coordinates, the final step was implementing the scaleCoordinates() method. The scaleCoordinates method takes in the xScale and yScale data inputted by the user and sets the model's coordinates to the specified size.

Testing:

I tested the program in a number of ways. Throughout each step of the process I would go ahead and implement error handling/input validation to make sure it functioned before moving on to a next step. For the uppercase letter input, I made sure and test for every other type of character other than an uppercase letter to make sure nothing tricked the program and caused a weird error.

Special cases I used included testing special characters, strings, other data types, etc. Everything works as expected.

```
Enter a single uppercase character to output [A-Z]: T

Generating data for the letter T

Enter desired X-axis scaling: 14
Enter desired Y-axis scaling: 30

poly2d 8
(0,30)
(14,30)
(14,24)
(10,24)
(10,0)
(6,0)
(6,24)
(0,24)
```

Conclusions:

The overall result of the assignment can be observed above! All possible areas a user could enter an error I made sure to handle. Something I might do differently next time would be to have the initial uppercase letter input allow for multiple attempts instead of booting out of the program. The project took around 3 to 4 hours of formulating an idea, re-familiarizing myself with C++, and implementing the program.