**CSC 121 – Computer Science I**

**Programming Project #2 – Unistroke Character Recognition**

*Project due on Friday, November 10 at 11:59PM*

***The Setup:***

You are working for the CoryCo Company as a programmer and developer. Unfortunately, the company has been forced to issue a recall of over 100,000 of their latest CoryPads. It seems that there is a bug in the unistroke character recognition module. Whoever was responsible for encoding the digit recognition (not you, luckily!) did not do it. At all. So, no digits can be recognized by the CoryPad.

The company originally hired you due to your vast experience writing video games, and it has confidence in your ability to fix this problem. You will need to completely implement the recognition code for digits. CoryCo informed you that **all the missing code is contained in the Recognizer class. You should not modify any other classes in the project**.

***Administrative Details:***

You may work on this project in pairs, if you wish.

***Your Task:***

You will need to make sure that you understand the steps in the pattern recognition process (algorithm) that has been discussed in class. In a nutshell, the algorithm says:

* Translate the array of points captured by the mouse events.
* Scale the array of points
* Normalize the array of points
* Compare the array of points with a “standard” set of patterns – the base set – to find the pattern that best matches the normalized set of points.

***The Steps of the Process:***

First, you will need to copy the project file (Project2.zip) from the course Moodle site and expand it into your I: drive, then open it in BlueJ. You will find comments labeled TO DO at the points where you need to work. Remember that all your work should be done in the Recognizer class. Each step is described in the list below – successfully completing the items in the list will result in stock options in CoryCo.

1. Write the *findMinX* method. This method should return the smallest x value from the points in the userPoints array.
2. Write the *findMinY* method. This method should return the smallest y value from the points in the userPoints array.
3. Write the *findMaxX* method. This method should return the largest x value from the points in the userPoints array.
4. Write the *findMaxY* method. This method should return the largest y value from the points in the userPoints array.
5. Write the *translate* method. This method should move the points in the userPoints array by sliding them as far to the upper-left as possible.
6. Write the *scale* method. This method should scale the points in the userPoints array by stretching the user’s stroke character so it fills the canvas as nearly as possible while maintaining the aspect ratio of the stroke character.
7. Finish the *insertOnePoint* method.
8. Write the *computeScore* method. It computes and returns a “score” that is a measure of how closely the normalized userPoints array matches a given pattern array in the baseset array.
9. Write the *findMatch* method. It finds and returns the index (an int) of the base set pattern which most closely matches the user stroke.
10. Programming style: formatting (readability that includes indentation) and comments, especially those describing methods. (**Take out the TODO instructions too.**)

*Hint #1: You should find the project easier if you do these steps in the order listed.*

*Hint #2: You can use the replay( ) method in the Driver class to play the contents of the points array; this should help you test the translate( ), scale( ), and normalize( ) methods.*