**1. Problem Statement:**

The goal of this programming project is to develop software to create geometric models of letters of the alphabet that we can include in our future graphics applications. For example, the letters A B C below are made up of sequences of very small line segments that define the outline of each letter, and a collection of planes that connect the front surface to the back surface.

In computer graphics, a collection of connected line segments is called a *polygon*. They are typically defined using a sequence of N points, and the assumption is that there is a line segment between adjacent points in the list. A *closed* polygon also has a line segment connecting the first point and the last point.

The goal of your programming project is to write a program that produces 2D models of FIVE upper case letters of the alphabet, for example your name or initials. Your program should prompt the user for which letter they want to generate, and then prompt the user for the *size* of the geometric model in the X and Y dimensions. Using this information, your program should transform your coordinate data as needed and print out the geometric model as a 2D polygon in the format below.



poly2d N <- the number N is the number of points in the polygon

x1 y1

x2 y2  <- the (x,y) coordinates of each point are separate lines

...

xN yN

**2. Design:**

the number N is the number of points in the polygon the (x,y) coordinates of each point are separate lines

Your first design task is to draw the five letters of the alphabet you will be modeling on a piece of graph paper using a sequence of line segments. To keep things simple, make “blocky” looking letters instead of letters with smooth curves. Now, calculate

the (x,y) coordinates of all of the points. You can use any range of (x,y) values you are comfortable working with. Starting with the (x,y) point that is closest to (0,0), go around your letter in a *clockwise* order, and write down the (x,y) coordinates of the remaining points. Congratulations, you now have a geometric models of your letters.

Your next task is to design a data structure to store this model in your program. One option is to store (x,y) point coordinates in “parallel arrays”, with one array for the x coordinates, and another array for y coordinates. The other option is to define a point class that stores (x,y) coordinates, and then have an array of point objects.

Somewhere in your program, you will need to initialize this data structure to contain the (x,y) coordinates for the three letters you calculated above. You can do this by “hard coding” (x,y) coordinates, or by reading one or more input files that contain the (x,y) coordinate values you figured out above.

Your final design task is to work out the math to stretch/shrink the (x,y) coordinates you have in your arrays, so you can create letters of different sizes. This is most easily done by finding the minimum and maximum coordinate values for each of your letters, and using this information multiply/divide all (x,y) coordinates in your data structure by a scale factor that will produce an output letter of the size specified by the user.

For example, if you look at your letter A and the x coordinates go from [1..16], then the original width is (16-1)=15. If you wanted to create an output letter that was 5 wide, you would have to multiply all x coordinates by 5 / 15 = 0.33 to shrink your initial model to the desired size. A similar calculation is needed for y coordinates.

**3. Implementation:**

Since we will be using C++ and OpenGL in several programming projects this semester, it would be a good idea to implement this project in C++ even though we are not using OpenGL yet. As always, you should break the code into appropriate classes / functions, and then add code to each class / function incrementally writing comments, adding code, compiling, debugging, a little bit at a time.

Remember to use good programming style when creating your program. Choose good names for variables and constants, use proper indenting for loops and conditionals, and include clear comments in your code. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

**4. Testing:**

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report. For example, if your program has a model for the letter L, you might get output that looks like this:

Enter letter to output [A..Z]: L

Enter X size: 20

Enter Y size: 30

poly2d 6

0 0

0 30

2 30

2 3

20 3

20 0