ASSIGNMENT 1

DUE DATE: FRIDAY, OCTOBER 6, 2000 – BEFORE MIDNIGHT

You may develop your application at home, HOWEVER, you must compile and run it on the SUN workstations in CSIL and submit this version via the on-line submission process. The TA has provided you with a sample Makefile which you can use to help compile your program. Details concerning the on-line submission process will be sent out via email.

Your assignment has three parts:

1. Write a procedure (`drawLine(`x_0`,`y_0`,`x_1`,`y_1`)`) to draw a line using the mid-point algorithm (Bresenham's Algorithm). The code should not have any floating point operations, and should handle lines of any slope using reflections. Note: you should mathematically derive the extensions to the algorithms presented in class (e.g. m<=1) and NOT duplicate code with `case` statements and/or multiple `if` statements.

2. Write a procedure (`fillPolygon()`) to fill a polygon - this should fill only the interior of the polygon. You should implement an active-edge table using the incremental algorithm described in the text (text p.98). It is permissible to use floating point numbers in calculating the slope of the lines. Your implementation should handle concave and self-intersecting polygons. Dynamic data structures (linked lists as opposed to arrays) are preferable.

3. Modify the pseudocode for the ellipse scan-conversion algorithm so that it takes into account the second-order differences as well as optimizing the computations (e.g.: use no floating point variables or calculations, precompute quantities such as a^2, etc.).

Note: you do NOT have to implement this! (just handwritten pseudocode). The original pseudocode is listed in the course notes, and on page 90 of the text book (discussed September 19th).

Note:

- OpenGL and GLUT will be used for this assignment. You are not required to use GLUI.
- Endpoints for the lines and polygons should handle input in the following ways:
    1. Using the mouse to select points on the window
    2. Via the keyboard to input pairs of integer (x,y) coordinates. Make sure to test the points to see if they are within the window.
    3. An input file containing start and endpoints defining a series of lines (the TA will email out a template for the input file).

  In all cases, the user should be able to select new points for the next line or polygon until she/he selects to exit.
- Your window size should be 500x500 pixels and your coordinate system should be from 0 to 100 in both the horizontal and vertical dimensions (use an orthographic projection). Pixel size should be passed as a command line argument to your program.
- For parts 1 and 2 (implementation), the only graphics routines you are allowed to use are: `glBegin(GL_POINTS), glVertex2i(x,y), and glEnd()` where `x` and `y` are integer coordinates of the pixel in our window (the origin is at the bottom lefthand corner).

Documentation:

- Algorithms should be described with comments in the source code only, where appropriate. Tell us what your code is doing!
- A short README file should be included, explaining how to run your program and how to input data.

Assignment Integrity:

- A similar assignment was given in a previous class, and copies of those assignments have been saved, so please resist the temptation to borrow them.
- These are **individual** assignments. You may discuss problems with fellow classmates but there is to be no sharing of code or written documentation/answers.