 Name: _____

Answer each question clearly and concisely. **You must show work or explain your answer to receive partial credit**. There are a total of 52 points on the exam. You may use a calculator if you want.

1. Give a brief explanation of each of the following terms (2 points each)

a. World window

b. Viewport

c. 2D perp vector

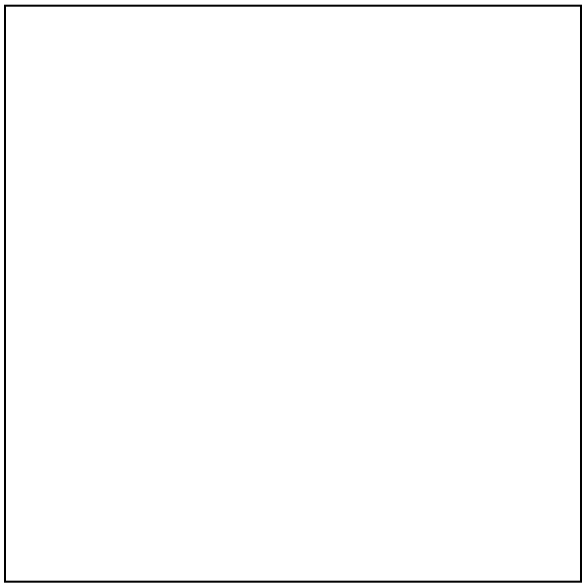
d. Cross-product

e. Homogeneous coordinates

2. Using *gluOrtho2D* and *glViewport*

a. Indicate what the screen window would look like after the following code is executed in the box representing the screen window shown below. Include color information and label the vertices with their **world coordinates**. The OpenGL control (named *canvas*) takes up the whole screen window. (6 pts.)

```
 1    Gl.glViewport (0, 0, canvas.Width, canvas.Height);
 2    Gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
 3    Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
 4    Gl.glColor3d(0.0, 0.0, 0.0);
 5
 6    Gl.glMatrixMode(Gl.GL_PROJECTION);
 8    Gl.glLoadIdentity();
 9    Glu.gluOrtho2D(-2, 2, -2, 2);
10
11    Gl.glBegin(Gl.GL_POLYGON);
12    for (int i = 0; i < 6; i++)
13    {
14      double x = Math.Cos(i * 60 * Math.PI / 180);
15      double y = Math.Sin(i * 60 * Math.PI / 180);
16      Gl.glVertex2d(x, y);
17    }
18    Gl.glEnd();
```



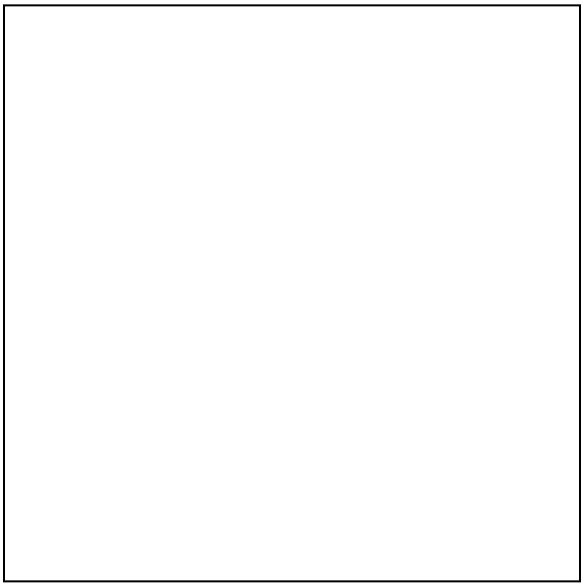
b. Describe how you could change a **single line** of the program so that the vertex generated when **i=0** is displayed at the **far right** edge of the screen window, and the generated when **i=3** is displayed at the **far left** edge. The shape of the object should remain the same. (3 pts)

Line # to change: _____

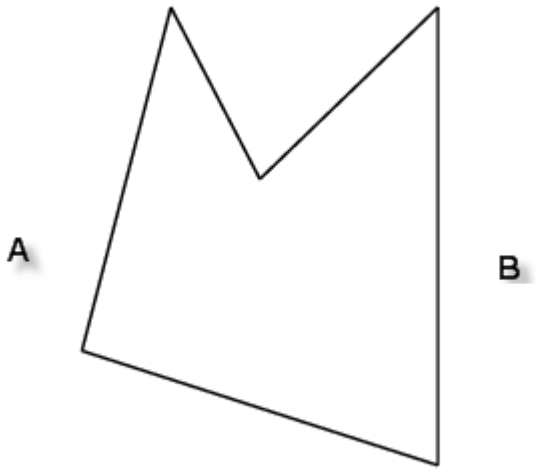
New contents: _____

c. Draw the contents of the screen window if the code from part (a) above were modified as follows (the modified lines are shown in **bold**). Again include color information, and label the vertices with their **world coordinates**. (6 pts)

```
1     Gl.glViewport (
        canvas.Width/4, canvas.Height/4,
        canvas.Width/2, canvas.Height/2
      );
2     Gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
3     Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
4     Gl.glColor3d(0.0, 0.0, 0.0);
5
6     Gl.glMatrixMode(Gl.GL_PROJECTION);
8     Gl.glLoadIdentity();
9     Glu.gluOrtho2D(0, 1, 0, 1);
10
11    Gl.glBegin(Gl.GL_LINE_STRIP);
12    for (int i = 0; i < 6; i++)
13    {
14      double x = Math.Cos(i * 60 * Math.PI / 180);
15      double y = Math.Sin(i * 60 * Math.PI / 180);
16      Gl.glVertex2d(x, y);
17    }
18    Gl.glEnd();
```

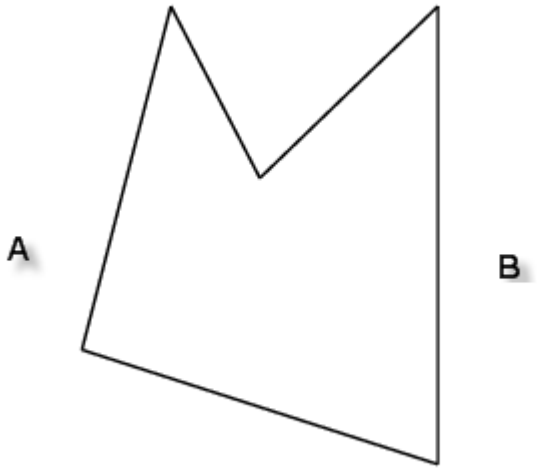


3. Suppose the Cyrus Beck clipping algorithm is applied to the figure shown below:



a. The line segment joining points A and B in the figure will not be clipped correctly. On the figure above, draw the clipped line segment that would result from connecting the points A and B in the figure. (3 pts)

b. On the figure below, indicate two points C and D such that the line segment joining C and D **would** be clipped correctly. (2 pts)



4. Consider the problem of transforming world coordinates into screen coordinates. The world window is a rectangle *W* whose lower left corner is at (W_L, W_B) and whose upper right corner is (W_R, W_T). The screen window is a rectangle *S* whose lower left corner is (S_L, S_B) and its upper right corner is (S_R, S_T)

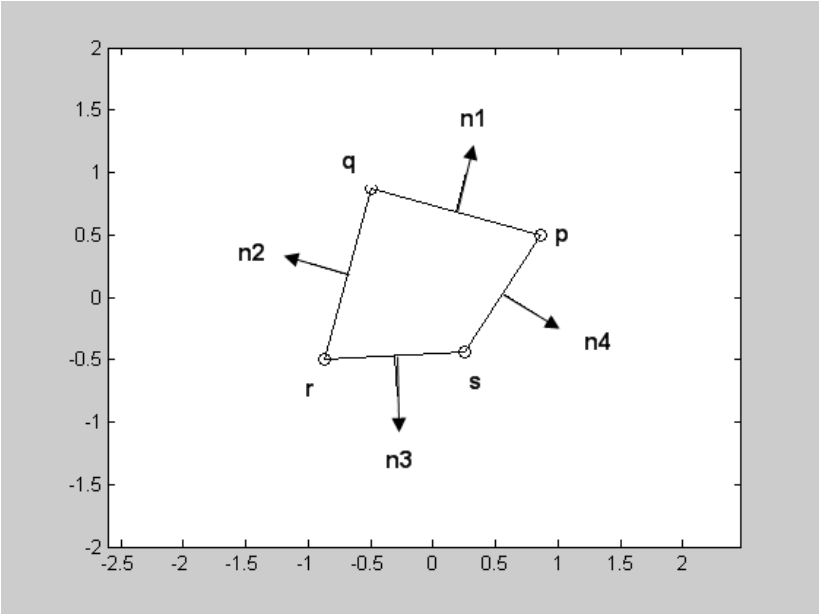
a. Write down the proportionality equation for used to compute S_X from W_X (you don't need to solve the equation for S_X) (4 pts)

b. Now we will use a combination of 3 affine transformation matrices to implement the world to viewport transformation. Let T be the final transformation matrix. Write T as the product of 3 matrices T_1, T_2 and T_3 that will implement the transformation of *W* into *S*. Label the matrices T1, T2 and T3 in your answer. Do not multiply them together at this step. T_1 is a translation matrix, T_2 is a scaling matrix, and T_3 is another translation matrix. (3 pts)

c. Compute the product of the left-most 2 matrices in your expression for T. (2 pts)

d. Describe how you could use the results from parts (a) and (c) to prove that the final matrix *T* can be used to implement the world-to-viewport transformation. (4 pts)

5. Consider the following figure:



The points and vectors labeled in the figure are given below:
p = (0.866, 0.5) q = (-0.5, 0.866) r = (-0.866, -0.5) s = (0.25, -0.433)
n_1 = (0.258819, 0.965926)

a. Verify that n_1 is perpendicular to the line joining the points p and q. (allow for round-off error) (3 pts)

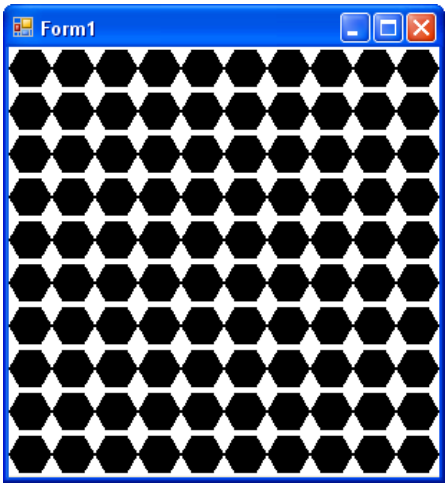
b. Compute the value of n_4 so that it points out of the polygon as shown in the figure. (3 pts)

c. Compute the point P where the ray `S = (1,0) + (-1, 0)t` hits the line segment joining s and p, with normal vector n_4. Recall that the formula to compute the hit time is:

$$t_{hit} = \frac{n \bullet (B - A)}{n \bullet c}$$

where A is the starting point of the ray, c is the direction of the ray, B is a point on the line joining *s* and *p* and *n* is the normal vector to the line joining *s* and *p*. (3 pts)

Parts (d) and (e) refer to the figure shown below. (Not actually given, but kept for reuse later)



This figure was drawn using 2 nested for loops as follows:

```
1     for (int row = 0; row < 10; row++) {
2        for (int column = 0; column < 10; column++) {
3           drawHexagon ()
4        }
5     }
```

The routine *drawHexagon* draws a unit hexagon; that is, the distance from the center of the hexagon to each vertex on the hexagon is 1. The control displaying the hexagons is named *canvas*. The values for *canvas.Width* and *canvas.Height* are the same.

d. Using the line numbers shown above, indicate where *gluOrtho2D* should be called to draw this figure (e.g "before line X), and then indicate the parameters that need to be used.

Where to call gluOrtho2D (2 pts): _____

Parameters for gluOrtho2D (2 pts):

`Left:` _____ `Right:` _____

`Bottom:` _____ `Top:` _____

d. **e.** Using the line numbers shown above, indicate where *glViewport* should be called to draw this figure, and then indicate the parameters that need to be used.

Where to call glViewport (2 pts): _____

Parameters for glViewport (lower left corner is x=Left, y=Bottom) (2 pts)

`Left:` _____ `Bottom:` _____

`Width:` _____ `Height:` _____