1. Give one important reason for using homogeneous coordinates in a graphics system.

2. Demonstrate, using algebra or pictures, that order matters when composing (concatenating) two transformations.

3. What would be the 16 entries in the 4×4 homogeneous matrix in the `Position` object after executing the following lines of code?

```
position.matrix = Matrix.translate(1.5, 2.0, -3.0);
position.matrix.mult(Matrix.scale(3, 2, 0.5));
```

4. What would be the 16 entries in the 4×4 homogeneous matrix in the `Position` object after executing the following line of code?

```
position.matrix = Matrix.rotateZ(90);
```

5. (a) Compute the 16 number that make up the 4×4 homogeneous matrix in the `Position` object after executing the following lines of code.

```
position.matrix = Matrix.translate(3.0, 0.0, 0.0)
            .times(Matrix.scale(3.0, 1.0, 1.0));
```

   (b) Compute the 16 number that make up the 4×4 homogeneous matrix in the `Position` object after executing the following lines of code.

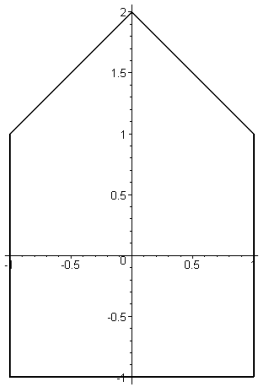
```
position.matrix = Matrix.scale(3.0, 1.0, 1.0)
            .times(Matrix.translate(3.0, 0.0, 0.0));
```

   (c) Compute the 16 number that make up the 4×4 homogeneous matrix in the `Position` object after executing the following lines of code.

```
position.matrix = Matrix.scale(3.0, 1.0, 1.0)
            .times(Matrix.translate(1.0, 0.0, 0.0));
```

   (d) How would you explain to someone, without doing the actual calculations, why the matrices in parts (a) and (b) are different, but the matrices in parts (a) and (c) are the same?

6. Suppose we have defined a subclass `DrawModel` of `Model` that represents the following image in the xy-plane. Draw a picture of the scene graph that the following code constructs. Draw a sketch of the scene that the code would produce when it is rendered. For each of the three model images, give the coordinates of the model's "origin" point.

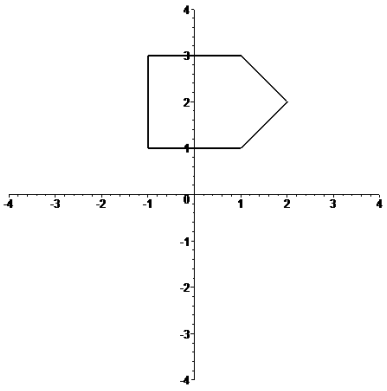



```
Model drawModel = new DrawModel();
Position p1 = new Position(drawModel);
Position p2 = new Position(drawModel);
Position p3 = new Position(drawModel);

p2.matrix = Matrix.translate(1.0, 3.0, 0.0).times(
            Matrix.rotateZ(45.0));
p3.matrix = Matrix.translate(1.0, 0.0, 0.0).times(
            Matrix.translate(1.0, 0.0, 0.0));
p1.matrix = Matrix.translate(-2.0, -2.0, 0.0).times(
            Matrix.rotateZ(-90.0));

Scene scene = new Scene();
scene.addPosition(p1, p2, p3);
```

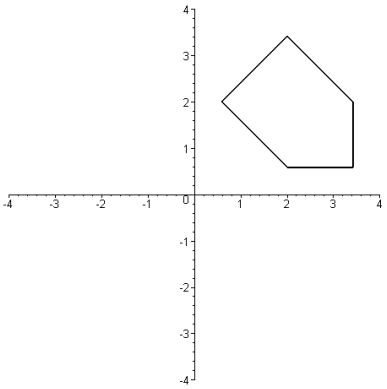
7. Using the same `DrawModel` class from the previous problem, write code that would generate the following image. Do this two different ways, one way with a translation preceding a rotation (in the code), and the second way with a rotation preceding a translation (in the code).



   (a) Translation and then rotation.

   (b) Rotation and then translation.

8. Using the same `DrawModel` class from the previous problem, write code that would generate the following image. Do this two different ways, one way with a translation preceding a rotation (in the code), and the second way with a rotation preceding a translation (in the code).



   (a) Translation and then rotation.

   (b) Rotation and then translation.

9. Suppose you have available to you a subclass `DrawUnitCircle` of `Model` that draws in the xy-plane a circle of radius one centered at the origin. What would the following code draw in the xy-plane? What does its scene graph look like?

```
Model circle = new DrawUnitCircle();
Position p1 = new Position(circle);
Position p2 = new Position(circle);
Position p3 = new Position(circle);
p1.matrix = Matrix.scale(2.0, 2.0, 2.0);
p2.matrix = Matrix.rotateZ(theta1).times(
            Matrix.translate(3.0, 0.0, 0.0));
p3.matrix = Matrix.rotateZ(theta2).times(
            Matrix.translate(1.5, 0.0, 0.0)).times(
            Matrix.scale(0.5, 0.5, 1.0));
Scene scene = new Scene();
scene.addPosition(p1, p2, p3);
```

10. Suppose you have available to you a subclass `DrawUnitCircle` of `Model` that draws in the xy-plane a circle of radius one centered at the origin. Use a *single* instance of this model, several instances of `Position`, and any of the methods `addPosition()`, `translate()`, `rotate()`, and `scale()`, to draw the following "F-shape". The vertical ellipse has its bottom at the point $(0, 0)$ and its top at the point $(0, 4)$, and at its widest it is 1 unit across. The upper horizontal ellipse is 2 units long. The lower horizontal ellipse is 1.5 units long. Its right hand end point has coordinates $(2, 2)$. The two horizontal ellipses are 0.5 units tall.