1. Briefly describe each of the following stages of the graphics rendering pipeline and put them in the correct order.
   (a) Clip
   (b) Rasterize
   (c) Project
   (d) Model2Camera

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

3. Which is better? Why is it better than the other two?
   - clipping in camera coordinates (just before projection)
   - clipping in view plane coordinates (just after projection)
   - clipping while rasterizing

4. Draw a simple diagram that explains how to derive the projection formula for the x-coordinate.

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

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

7. 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.identity();
position.matrix.mult(Matrix.rotateZ(90));
```

8. (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?

9. Consider the following block of code.

```
Model m = new Model();
m.addVertex{new Vertex(0,  1, -1),
            new Vertex(1, -1, -1),
            new Vertex(1,  1, -1)};
m.addLineSegment(new LineSegment(0, 1),
                 new LineSegment(1, 2),
                 new LineSegment(2, 0));
m.addColor(Color.white, Color.white, Color.white);
```

(a) What picture would be drawn if we rendered the model `m` built by this code (assume an orthographic projection looking down the z-axis)? In your picture, label the vertices in order from vetrex 0 to vertex 2.

(b) How many Java objects does this code instantiate, counting all the objects that are composed in other objects? Draw a detailed picture of what the objects "look like" in the Java heap (which objects refer to which objects?).

(c) Write a minimal(!) Java program that will compile and run and draw the above model (by "draw" I mean write the `FrameBuffer` to a file that can be viewed).

10. Suppose we have a `FrameBuffer` object that is 200 pixels by 200 pixels. Suppose we define a 100 pixel high by 200 pixel wide viewport within the `FrameBuffer` so that the viewport is centered in the `FrameBuffer`.

Let $v = (4, -3, -6)$ be a vertex in the camera coordinate system.

(a) What vertex in the viewplane $z = -1$ does the vertex v project to?

(b) What (2-dimensional) vertex is the projected v transformed to by the `Viewport` transformation pipeline stage?

(c) Which pixel in the `FrameBuffer`'s viewport represents the vertex v? (Be sure to use the correct viewport pixel coordinate system.)

(d) Which pixel in the `FrameBuffer` represents the vertex v? (Be sure to use `FrameBuffer` pixel coordinates.)

11. Let $v_1 = (0.4, 0.8, -1)$ and $v_2 = (2.2, -0.3, -1)$ be two vertices in the viewplane $z = -1$ and suppose that they define a `LineSegment`. Suppose that v_1 has the color $(r, g, b) = (0.5,\ 0.5,\ 0.5)$ and v_2 has the color $(r, g, b) = (0.0,\ 1.0,\ 0.7)$.

    Let $v_c = (1, y_c, -1)$ denote the new vertex created when the line segment is clipped on the line $x = 1$.

    (a) Compute the value y_c in the new vertex v_c.

    (b) Compute the new color at the new vertex v_c.

    Hint: First, solve for t in the x-coordinate of this equation

    $$(1 - t)v_1 + tv_2 = (1, y_c).$$

    Then use t to compute y_c. Then use t to linearly interpolate each of the three new color components.

12. Let $v_0 = (1.5, 1.0, -1.0)$ and $v_1 = (0.0, -2.0, -1.0)$ be two vertices in the viewplane $z = -1$ and suppose that they define a `LineSegment`. Suppose that v_0 has the color $(r, g, b) = (0.5,\ 0.5,\ 0.5)$ and v_1 has the color $(r, g, b) = (0.0,\ 1.0,\ 0.7)$.

    (a) Draw a simple (2D) picture of the viewplane, the view rectangle, and this line segment.

    (b) Compute the vertices of the new line segment that this line segment will be clipped to.

    (c) Compute the value of the red color component at one of the new vertices. Indicate in your picture above for which vertex you are computing the new red value.

13. Suppose we have a `FrameBuffer` object `fb` that is 10 pixels wide by 3 pixels high and this `FrameBuffer`'s viewport is set to be all of the `FrameBuffer`.

```
FrameBuffer fb = new FrameBuffer(10, 3);
```

Suppose we have a `Model` object `m` and this model contains a single `LineSegment` that is a diagonal across the view rectangle in the view plane.

```
Model m = new Model();
m.addVertex(new Vertex(-1, -1, -1),
            new Vertex( 1,  1, -1));
m.addLineSegment(new LineSegment(0, 1));
m.addColor(Color.white, Color.white);
```

Suppose we render this model into the framebuffer.

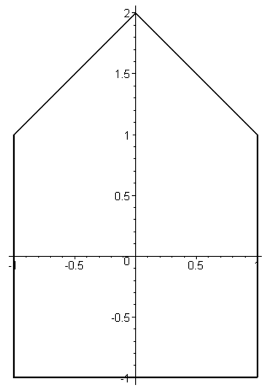
```
Scene scene = new Scene();
scene.addPosition(new Position(m));
Pipeline.render(scene, fb.vp);
```

Which pixels in the framebuffer are turned on by the rasterizer when it draws this line?

Hint: Notice that the above code is a complete program. You can check your answer by using the actual renderer with the following options set on the pipeline.

```
RasterizeAntialias.debug = true;
Pipeline.debug = true;
Pipeline.render(scene, fb.vp);
```

14. 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.



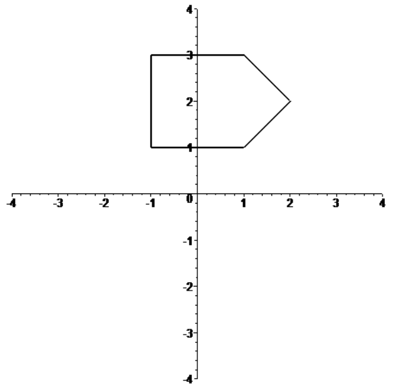
```
Model drawModel = new DrawModel();
Position p1 = new Position();
Position p2 = new Position(drawModel);
Position p3 = new Position();
Position p4 = new Position(drawModel);
Position p5 = new Position(drawModel);

p1.addNestedPosition(p2);
p1.addNestedPosition(p3);
p3.addNestedPosition(p4);
p3.addNestedPosition(p5);

p3.matrix = Matrix.translate(1.0, 0.0, 0.0);
p4.matrix = Matrix.translate(0.0, 3.0, 0.0).times(
            Matrix.rotateZ(45.0));
p5.matrix = Matrix.translate(1.0, 0.0, 0.0);
p2.matrix = Matrix.translate(-2.0, -2.0, 0.0).times(
            Matrix.rotateZ(-90.0));

Scene scene = new Scene();
scene.addPosition(p1);
```

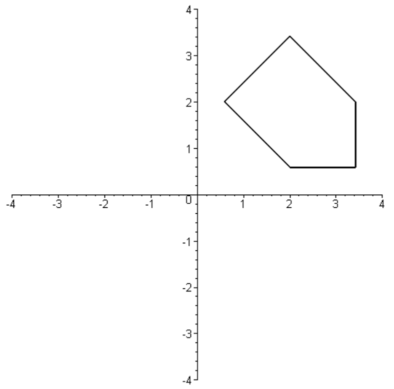
15. Using the same `DrawModel` class from the previous problem, write code that would generate the following image. Do this two different ways, one way with a translation preceding a rotation (in the code), and the second way with a rotation preceding a translation (in the code).



(a) Translation and then rotation.

(b) Rotation and then translation.

16. 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.

17. 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();
Position p2 = new Position(circle);
Position p3 = new Position(circle);
Position p4 = new Position(circle);
p1.addNestedPosition(p2);
p1.addNestedPosition(p3);
p1.addNestedPosition(p4);
p2.matrix = Matrix.scale(2.0, 2.0, 2.0);
p3.matrix = Matrix.rotateZ(theta1).times(
            Matrix.translate(3.0, 0.0, 0.0));
p4.matrix = Matrix.rotateZ(theta2).times(
            Matrix.translate(1.5, 0.0, 0.0)).times(
            Matrix.scale(0.5, 0.5, 1.0));
```

18. What does the following code draw in the xy-plane? Hint: Draw a picture of the scene graph data structure built by this code. Then determine what the transformations do to the geometry.

```
Model m1 = new Model();
m1.addVertex(new Vertex(-1, 0, 0),
             new Vertex( 1, 0, 0));
m1.addLineSegment(new LineSegment(0, 1));
Position p2 = new Position();
Position p3 = new Position(m1);
Position p4 = new Position(m1);
p2.addNestedPosition(p3);
p2.addNestedPosition(p4);
p3.matrix = Matrix.translate(1, 0, 0);
p4.matrix = Matrix.translate(2, 0, 0).times(
            Matrix.rotateZ(90)).times(
            Matrix.scale(0.5, 1, 1));
Position p5 = new Position();
Position p6 = new Position();
Position p7 = new Position();
Position p8 = new Position();
p5.addNestedPosition(p6);
p5.addNestedPosition(p7);
p5.addNestedPosition(p8);
p6.addNestedPosition(p2);
p7.addNestedPosition(p2);
p8.addNestedPosition(p2);
p7.matrix = Matrix.rotateZ(90);
p8.matrix = Matrix.rotateZ(-135);
```