1. What is a FrameBuffer data structure? What does it contain? What does it represent? How is it used in a graphics rendering pipeline?

2. What is a Scene data structure? What does it contain? What does it represent? How is it used in a graphics rendering pipeline?

3. Briefly describe the contents of a ppm file. As an image file format, what advantage does it have? What is its disadvantage?

4. Briefly describe each of the following coordinate systems.
   (a) model coordinates
   (b) camera coordinates

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

6. When is it preferable to use orthographic (parallel) projection? When is it preferable to use perspective projection? Explain why.

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

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

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

```
model.model2Identity();
model.modelTranslate(1.5, 2.0, -3.0);
model.modelScale(3, 2, 0.5);
```

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

```
model.model2Identity();
model.modelRotate(90, 0, 0, 1);
```

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

    ```
    model.model2Identity();
    model.modelTranslate(3.0, 0.0, 0.0);
    model.modelScale(3.0, 1.0, 1.0);
    ```

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

    ```
    model.model2Identity();
    model.modelScale(3.0, 1.0, 1.0);
    model.modelTranslate(3.0, 0.0, 0.0);
    ```

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

    ```
    model.model2Identity();
    model.modelScale(3.0, 1.0, 1.0);
    model.modelTranslate(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?

12. Consider the following block of code.

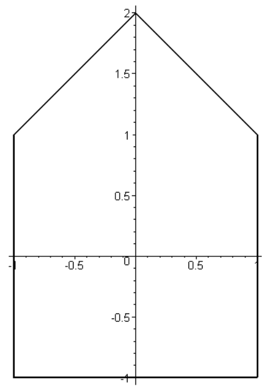
```
Vertex[] v = {new Vertex(0,  1, -1),
              new Vertex(1, -1, -1),
              new Vertex(1,  1, -1)};
LineSegment ls1 = new LineSegment(v[0], v[1]);
LineSegment ls2 = new LineSegment(v[1], v[2]);
LineSegment ls3 = new LineSegment(v[2], v[0]);
Model m = new Model();
m.addLineSegment(ls1);
m.addLineSegment(ls2);
m.addLineSegment(ls3);
```

    (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 appropriate points in the order v[0] to v[2].

    (b) How many Java objects does this code instantiate (count the objects that are composed in objects like `Model` and `LineSegment`)? 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).

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



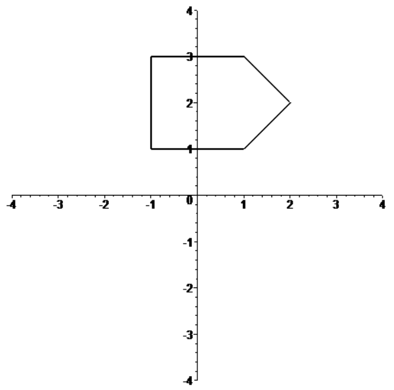
```
Model m1 = new Model();
Model m2 = new Model();
Model m3 = new Model();
Model m4 = new Model();
Model m5 = new Model();
Model drawModel = new DrawModel();

m1.addSubModel(m2);
m1.addSubModel(m3);
m3.addSubModel(m4);
m3.addSubModel(m5);
m2.addSubModel(drawModel);
m4.addSubModel(drawModel);
m5.addSubModel(drawModel);

m3.modelTranslate(1.0, 0.0, 0.0);
m4.modelTranslate(0.0, 3.0, 0.0);
m4.modelRotate(45.0, 0.0, 0.0, 1.0);
m5.modelTranslate(1.0, 0.0, 0.0);
m2.modelTranslate(-2.0, -2.0, 0.0);
m2.modelRotate(-90.0, 0.0, 0.0, 1.0);

Scene scene = new Scene();
scene.addModel(m1);
```

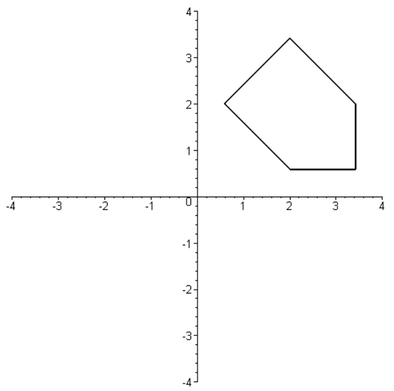
14. Using the same `DrawModel` class from the previous problem, write code that would generate the following image. Do this two different ways, one way with a translation preceding a rotation, and the second way with a rotation preceding a translation.



(a) Translation and then rotation.

(b) Rotation and then translation.

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, and the second way with a rotation preceding a translation.



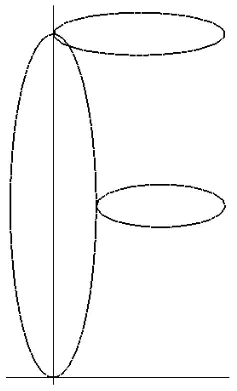
(a) Translation and then rotation.

(b) Rotation and then translation.

16. 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?

```
Model m1 = new Model();
Model m2 = new Model();
Model m3 = new Model();
Model m4 = new Model();
Model circle = new DrawUnitCircle();
m1.addSubModel(m2);
m1.addSubModel(m3);
m1.addSubModel(m4);
m2.addSubModel(circle);
m3.addSubModel(circle);
m4.addSubModel(circle);
m2.modelScale(2.0, 2.0, 2.0);
m3.modelRotate(theta1, 0, 0, 1);
m3.modelTranslate(3.0, 0.0, 0.0);
m4.modelRotate(theta2, 0, 0, 1);
m4.modelTranslate(1.5, 0.0, 0.0);
m4.modelScale(0.5, 0.5, 1.0);
```

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. Use a *single* instance of this model, several instances of `Model`, and any of the methods `addSubModel()`, `modelTranslate()`, `modelRotate()`, and `modelScale()`, 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.



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

```
LineSegment ls = new LineSegment(new Vertex(-1, 0, 0), new Vertex(1, 0, 0));
Model m1 = new Model();
m1.addLineSegment(ls);
Model m2 = new Model();
Model m3 = new Model();
Model m4 = new Model();
m2.addSubModel(m3);
m2.addSubModel(m4);
m3.addSubModel(m1);
m4.addSubModel(m1);
m3.modelTranslate(1, 0, 0);
m4.modelTranslate(2, 0, 0);
m4.modelRotate(90, 0,0,1);
m4.modelScale(0.5, 1, 1);
Model m5 = new Model();
Model m6 = new Model();
Model m7 = new Model();
Model m8 = new Model();
m5.addSubModel(m6);
m5.addSubModel(m7);
m5.addSubModel(m8);
m6.addSubModel(m2);
m7.addSubModel(m2);
m8.addSubModel(m2);
m7.modelRotate(90, 0,0,1);
m8.modelRotate(-135, 0,0,1);
```

19. 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.)

20. 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` ls. 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 `LineSegment` ls 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: Solve for t in the 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.

21. Suppose we have a `FrameBuffer` object `fb` that is 1000 pixels by 1000 pixels and this `FrameBuffer`'s viewport is set to be all of the `FrameBuffer`.

```
FrameBuffer fb = new FrameBuffer(1000, 1000);
```

Suppose we have a `Model` object `m` and this model contains a `LineSegment` object with the following two vertices.

```
Model m = new Model();
Vertex v0 = new Vertex(1, 1, 0);
Vertex v1 = new Vertex(1, 1, -1);
m.addLineSegment( new LineSegment(v0, v1) );
```

Suppose we render this model into the framebuffer.

```
m.model2Identity();
m.modelTranslate(0, 0, -2);
m.modelScale(0.5, 1.5, 1.0);
Scene scene = new Scene();
scene.addModel( m );
Pipeline.render(scene, fb);
```

(a) What vertex in camera space is each vertex `v0` and `v1` transformed to by the `Model2Camera` pipeline stage?

(b) What vertex in the viewplane $z = -1$ is each transformed vertex `v0` and `v1` projected to?

(c) What (2-dimensional) vertex is each projected vertex `v0` and `v1` transformed to by the `Viewport` transformation pipeline stage?

Hint: Notice that the above code is a complete program. You can check your answers to this problem (and also the previous two problems) by using the actual renderer with the following options set on the pipeline.

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

22. 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();
Vertex v0 = new Vertex(-1, -1, -1);
Vertex v1 = new Vertex( 1,  1, -1);
m.addLineSegment( new LineSegment(v0, v1) );
```

Suppose we render this model into the framebuffer.

```
m.model2Identity();
Scene scene = new Scene();
scene.addModel( m );
Pipeline.render(scene, fb);
```

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.

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