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) camera coordinates
   (b) view plane coordinates
   (c) pixel-plane coordinates
   (d) viewport coordinates (in a framebuffer)
   (e) framebuffer coordinates

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

6. Briefly describe each of the following stages of the graphics rendering pipeline and put them in the correct order.
   - Clip
   - Rasterize
   - Project

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

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

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 using 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 point in the pixel-plane is the projected v transformed to by the pixel-plane transformation?

(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.)

(e) Write a minimal Java program that will compile and run and use its debugging output to verify your calculations.

11. Suppose that a framebuffer is 600 pixels wide by 400 pixels tall and it is stored in row-major form with a single `int` per pixel.

   (a) Where in the pixel array is the pixel with framebuffer coordinates (257, 188)? Show your calculation.

   (b) Write a brief, but complete, program to verify your result from part (a).

   (c) Suppose that the framebuffer has a viewport that is 200 pixels wide by 200 pixels tall with upper left hand corner at framebuffer coordinate (150, 150). Where in the pixel array is the pixel with viewport coordinates (88, 165)? Show your calculation.

   (d) Write a brief, but complete, program to verify your result from part (c).

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

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

   (c) 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.

   Note: You can check your answer using the renderer_5 by writing a small program and setting the following options on the pipeline.

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

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));
```

Suppose we render this model into the framebuffer with antialiasing truned off.

```
Scene scene = new Scene();
scene.addModel(m);
RasterizeAntialias.doAntialiasing = false;
Pipeline.render(scene, fb);
```
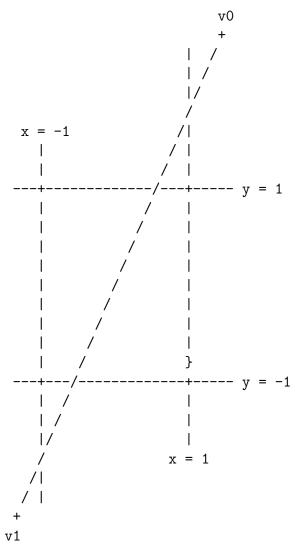
Which pixels in the framebuffer are turned on (that is, not black) by the rasterizer when it draws this line?

Hint: Notice that the above code is a complete program. You can check your answer by using renderer_4 with the following options set on the pipeline.

```
Pipeline.debug = true;
Clip.debug = true;
RasterizeAntialias.debug = true;
RasterizeAntialias.doAntialiasing = false;
Pipeline.render(scene, fb);
System.out.println(fb);
```

14. Redo the last problem with antialiasing turned on.

15. Suppose that the order of the edges used by the clipping algorithm is

```
x =   1,
y =  -1,
x =  -1,
y =   1.
```

(a) How many times would the line in the following picture from v0 to v1 be clipped? Label the new vertices in the order in which the clipping algorithm creates them.

```
                                 v0
                                  +
                               |  /
                               | /
                               |/
                               /
           x = -1             /|
              |              / |
              |             /  |
          ---+------------/---+----- y = 1
              |          /    |
              |         /     |
              |        /      |
              |       /       |
              |      /        |
              |     /         |
              |    /          |
              |   /           }
          ---+---/------------+----- y = -1
              | /             |
              |/              |
              /              x = 1
             /|
            / |
           +
          v1
```

(b) Using this order for the edges in the clipping algorithm, find an example of a line that will be clipped four times.

16. Here is a simple event driven program. What is missing from it? That is, what needs to be added to this program so that it compiles, runs, and noticeably responds to events? (Note: Do not delete any code from the program. Only add code to it.)

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class WhatIsMissing
{
   public WhatIsMissing()
   {
      JFrame jf = new JFrame("WhatIsMissing");
      jf.setLayout(new FlowLayout());
      JButton jb = new JButton("Hello");
      jb.addActionListener(this);
      jf.add(jb);
      jf.pack();
      jf.setSize(400, 400);
   }

   @Override public void keyTyped(KeyEvent e) {
      System.out.println(e);
   }
   @Override public void componentMoved(ComponentEvent e) {
      System.out.println(e);
   }
   @Override public void componentResized(ComponentEvent e) {
      System.out.println(e);
   }

}
```