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

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

7. Consider the following block of code.

```
Model m = new Model("Problem 7");
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));
```

   (a) Draw a picture of what 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).

8. 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.

9. 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).

10. 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`.
   (a) Draw a simple (2D) picture of the viewplane, the view rectangle, and this line segment.
   (b) Write a program that renders the line segment into an 800 by 800 pixel framebuffer.
   (c) Use your program to determine, from the debugging information, the exact pixels where the line leaves the framebuffer.

11. 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.

```
Scene scene = new Scene();
scene.addModel(m);
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_2 with the following options set on the pipeline.

```
Pipeline.debug = true;
Rasterize_Clip.debug = true;
Pipeline.render(scene, fb);
System.out.println(fb);
```

12. 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.)

```
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);
   }

}
```