Question Q2–1: (Solution, p 2) What is the motivation behind generalizing homogeneous coordinates to allow the last entry to be other than 1?

Question Q2–2: (Solution, p 2) Explain the concept of Gouraud shading.

Question Q2–3: (Solution, p 2) Complete the following code to draw a line from $(\texttt{x0}, \texttt{y0})$ to $(\texttt{x1}, \texttt{y1})$ (both point being given in window coordinates), where $\texttt{x0} < \texttt{x1}$, $\texttt{y0} < \texttt{y1}$, and $\texttt{H} < \texttt{W}$ where $\texttt{H} = \texttt{y1} - \texttt{y0}$ and $\texttt{W} = \texttt{x1} - \texttt{x0}$. While your code does not need to use Bresneham's line drawing algorithm precisely, it should result in the same basic line.

```
public static void drawLine(int x0, int y0, int x1, int y1) {
    int W = x1 - x0;
    int H = y1 - y0;


    int y = y0;
    for(int x = x0; x <= x1; x++) {
        drawPixel(x, y);




    }
}
```

Question Q2–4: (Solution, p 2) Supersampling often involves using a non-uniform filter to combine sub-pixels into the color of a pixel. The following is a common filter.

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

Explain some of the motivation behind using such a filter.

Question Q2–5: (Solution, p 2) Give a vector H representing a plane passing through the left plane of the perspective frustum. Recall that the left plane passes through the points $(0, 0, 0)$, $(left, bottom, -near)$, and $(left, top, -near)$.

Solution Q2–1: (Question, p 1) Using a perspective transformation to map a point from eye coordinates, the x-coordinate of the transformed point will depend on the reciprocal of its x- and z-coordinates in eye coordinates. Regular matrices do not allow expression of such a division. But by generalizing the last entry of homogeneous coordinates to represent a scaling factor, we can still represent this transformation transforming the z-coordinate to be the homogeneous coordinate.

Solution Q2–2: (Question, p 1) Frequently designers approximate complex curved objects with a set of polygons. Unfortunately, if we treat these surfaces as flat, the shading for the surfaces makes the polygons quite obvious, as light will reflect off each surface in different ways. In Gouraud shading, we interpolate brightness across each polygon, so that the abrupt shifts in brightness are not obvious at the edges of the polygons.

Solution Q2–3: (Question, p 1)

```
public static void drawLine(int x0, int y0, int x1, int y1) {
    int W = x1 - x0;
    int H = y1 - y0;
    double error = 0.0;
    int y = y0;
    for(int x = x0; x <= x1; x++) {
        drawPixel(x, y);
        error += (double) H / W;
        if(error > 0.5) {
            y++;
            error -= 1.0;
        }
    }
}
```

[Of course, Bresneham's algorithm involves scaling `error` so that it no longer needs to be a `double`.]

Solution Q2–4: (Question, p 1) Often, different pixels share the same subpixels. We would like to avoid giving these pixels undue weight, so we choose to distribute their weight across the pixels affected. In the given example, it may be that the upper right subpixel is shared by three other pixels, all of which assign it a $1/16$ weight, so that contributes a total weight of $1/4$, just as the center subpixel would if it is not shared by any other pixels.

Solution Q2–5: (Question, p 1) $(near, 0, left, 0)$, or any multiple thereof. [This problem is simply a matter of observing that H is of the form (A, B, C, D) where $H \cdot P = 0$ for all three points on the plane. From here, we solve the three equations for A, B, C, and D. Since three equations underspecify a system over four unknowns, you will end up having to make an arbitrary choice for one variable.]