Bilinear Interpolation in General

For this assignment, you'll make repeated use of bilinear interpolation. All bilinear interpolation involves interpolating a value between four known (point, value) pairs. The bilinear interpolation equation is always

$$v = a\,x + b\,y + c\,x\,y + d$$

If we evaluate this for a known (x,y) position, we should get the known value v at that position. This gives us four equations, which we use to solve the four unknowns a, b, c, and d.

Once we have solved for the coefficients a, b, c, and d, we can now evaluate this expression to get the value for any (x, y) position within the original four points. (You can use it for extrapolation as well, but it's not recommended.)

For bilinear interpolation of intensity (Question 2), the quantity that we're interpolating is intensity values for the pixels. For this, we can set up one system of four equations, four unknowns and solve it.

For bilinear warping (Question 3), the quantities we interpolate are the corresponding (x',y') locations of the transformed points. For this, we set up and solve *two* systems of four equations, four unknowns each: one for x' and one for y'.

Notice that each of these gives us a *general* solution for the interpolation---we then just plug in the desired (x,y) location we want to interpolate a value for.

Bilinear Interpolation within a Square or a Square Grid

The equations given in your text and above are the more general solution for interpolating between any four points. If these four points lie on a unit square, the problem simplifies greatly. (Try it yourself.) It's a simple matter then to solve this system once for a unit square with pixels values v_1, v_2, v_3, and v_4 at the corners. Now, just use this solution to interpolate any position within a square with any four values at the corners.

Furthermore, if you're interpolating pixel values that lie between points on an evenly-spaced grid, don't worry about the absolute location of the value you're interpolating. Just calculate its position with the square in which it lies, use the four pixel values at the corner of this square, and then interpolate as if they formed a unit square.

This means that when you code up bilinear interpolation, *you don't have to solve systems of equations for every* interpolation. Just solve once the equations necessary to interpolate within a unit square with four arbitrary (variable) values at the corners. Then just use this equation over and over, treating each 2x2 square of surrounding pixels as a unit square.

Example: Suppose you want to bilinearly interpolate to get the value of an image at (11.8,12.9). First, get the four surrounding pixels at (11,12), (11,13), (12,13), and (12,12). Now treat these values as the corners of a unit square and interpolate the position (0.8,0.9).

It's up to you to use the general form, solve for the solution for interpolation within a square, and to code up how to use this over and over within a grid of evenly-spaced pixels.