1. Suppose I want to apply the filter \([0.5 \ 0\ 0.5]\) vertically and \([1\ 2\ 1]\) horizontally. What would be the 3x3 filter kernel for this filter?

\[
\begin{bmatrix}
0.5 \\
0 \\
0.5
\end{bmatrix}
\begin{bmatrix}
1 & 2 & 1
\end{bmatrix} =
\begin{bmatrix}
0.5 & 1 & 0.5 \\
0 & 0 & 0 \\
0.5 & 1 & 0.5
\end{bmatrix}
\]

2. Given the filter from question 1, what pixel value will be computed for location \((2, 3)\)?

\[
12 \times 0.5 + 14 + 22 \times 0.5 + 22 \times 0.5 + 36 + 40 \times 0.5 = 98
\]

3. (right top) Write a filter that enhances contrast. Use whatever constants you are comfortable with in this filter.

```cpp
for(int r=0; r<m_image2.Height(); r++)
{
    // c is columns of the array
    for(int c=0; c<m_image2.Width() * 3; c++)
    {
        double p = m_image[r][c] / 255.;
        if(p < 0.25)
            p = p * 0.5;
        else if(p < 0.75)
            p = 0.125 + ((p - 0.25) / 0.5) * 0.75;
        else
            p = 0.875 + ((p - 0.75) / 0.25) * 0.125;
        m_image2[r][c] = BYTE(p * 255);
    }
}
```

I find it much easier to do these tasks if you normalize pixels to the range 0-1.
4. If I apply the highpass filter $[-1 \ 0 \ 1]$ to the rows of an image, you get a highpass filtered image with all lows removed. Create a filter that will enhance the edges rather than showing only the edges. Write the code for computing the pixel value for this filter (ignoring boundary conditions):

```c
double enhance = 0.1;
double filter[3] = {-1 * enhance, 1, 1 * enhance};

m_image2[r][c] = BYTE(m_image1[r][c-3] * filter[0] + m_image1[r][c] * filter[1] + m_image1[r][c+3] * filter[2]);
```

For the following, assume this affine transformation is from the source to the destination:

$$x' = 2x + 15 \quad y' = 2y + 45$$

5. If $(10, 100)$ is a pixel in the source image, what is the equivalent location in the destination image?

$$x' = 2x + 15 = 2(10) + 15 = 35$$
$$y' = 2y + 45 = 2(100) + 45 = 245$$

6. If $(10, 100)$ is a pixel in the destination image, what is the equivalent location in the source image?

$$10 = 2x + 15$$
$$2x = -5$$
$$x = -2.5$$

$$100 = 2y + 45$$
$$2y = 55$$
$$y = 27.5$$

7. This affine transformation will make the objects in the destination:
   - [ ] Twice as big as the source
   - [ ] Half as big as the source
   - [ ] The same size as the source
   - [ ] Invisible

   Twice as big as the source