1. Pseudo-Code for one implementation of a $3 \times 3$ convolution:

$$[s_x, s_y] = \text{Dimensions(image)}$$

Create an output array the same size as the input array.

while $i < s_x$ do
  while $j < s_y$ do
    if $(i > 0) \& (i < s_x - 1) \& (j > 0) \& (j < s_y - 1)$ then
      All values are defined so the convolution can be computed normally.
    else
      This is an edge location.
      Compute the convolution with the available information.
      Example techniques are zero padding, setting the output to zero, and edge wrapping.
    end if
  end while
end while

Return the convolved image.
2. All convolutions were implemented as being zero padded.

**Figure 1:** (a) Filter, (b) Original Image, (c) Output Image, and (d) difference Image (0 difference = 128).
Figure 2: (a) Filter, (b) Original Image, (c) Output Image, and (d) difference Image (0 difference = 128).
Figure 3: (a) Filter, (b) Original Image, (c) Output Image, and (d) difference Image (0 difference = 128).
Figure 4: (a) Filter, (b) Original Image, (c) Output Image, and (d) difference Image (0 difference = 128).
3. All convolutions were implemented with zero padding.
**All images on this page are scaled so that the resultant image has value from 0-255.**
To compute:

\[
\text{OutputImage} = \sqrt{\left(\text{InputImage} \ast \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)^2 + \left(\text{InputImage} \ast \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right)^2}
\] (1)

** The top two images on this page are scaled so that the resultant image has value from 0-255. **
4. Extra Credit

The $3 \times 3$ median filter blurs the image while preserving the edges.