2.1

The required edge location algorithm is:

shrink objects in image;
subtract shrunken objects from original image;
// the result is to leave the edges

To prove this, note that the EDGE table in Chapter 2 has a single 1 in the lower left of the output section. On the other hand, the corresponding SHRINK table would have a single 1 in the lower right of the output section. Combining the two outputs would yield 1's in both of the lower outputs, but leave 0's in both of the upper outputs. Thus the total output is equal to A0, i.e. the same as for the original image. Hence the edge is the original image minus the shrunken image.

2.2

(a) For shrinking, it is probably best to take the off-image values as 1's, so that objects that are partly occluded at the boundary of the image remain contiguous with the boundary.

(b) For expanding, it is best to take the off-image values as 0's, so that apparent object areas do not move inwards from the boundary with each expansion phase. (The two cases of shrinking and expanding ought in principle to be duals of each other. However, it seems better not to follow this line of thought because in Chapter 2 we envisage that images will have a number of small, separate binary picture objects with value 1 against a 0's background, and the chosen strategies need to give the least error in this type of situation.)

(c) For a blurring convolution, taking the off-image values as 0's or 1's, or any other fixed values, would mix significantly different values into the current pixel value, and the local grey-scale would change abruptly. The smoothest variation arises when the current pixel value and/or its nearest neighbors within the image are used to estimate the new current pixel value. I.e. it is better to seek a locally acceptable value than to impose a fixed value that will be a bad choice over much of the image boundary. The simplest choice is to use the intensity of the nearest pixel within the image. Of course, some averaging of neighboring pixel intensities might serve to minimize noise.

2.3

The NOIZE algorithm looks for a 1 with a single 1 adjacent to it, and allows it to be changed to 0. However, the 1 could be at the end of a line of single pixel width. Hence the algorithm could remove the end 1, and subsequently keep on removing further 1's until the whole line had been eliminated. This cannot happen with the NOISE algorithm, since each 1 that is removed is isolated, and therefore a 'chain reaction' cannot occur.

3.3

This type of filter emulates the mode filter, and as such will enhance images by taking a strong decision at each pixel as to which side of an edge the pixel lies. However, the two extreme values in a distribution are very poorly defined because they are outliers, and have absolutely no immunity to impulse noise. They are also subject to the extremes of Gaussian noise, and as such they are again inaccurately defined, and poor examples to use as paradigm intensity values for the new output.

3.5

(a) See the main text in Chapter 3.

(b) Clear the whole histogram initially; then clear only the elements that have been filled, after finding each median. Find the window minimum; then set *sum* equal to the minimum before proceeding to find the median. Not having to clear the whole histogram essentially saves 256 operations, and there are still 128 × 2 to be done, so there is a saving by a factor two. Setting *sum* at the window minimum value can at best save ½ × 128 × 2 operations, but in fact the saving will typically be ~half of this figure.

(c) See the main text in Chapter 3.

(d)

```
original:      1 2 1 1 2 3 0 2 2 3 1 1 2 2 9 2 2 8 8 8 7 8 8 7 9 9 9
3 × 3 median:  ? 1 1 1 2 2 2 2 2 2 1 1 2 2 2 2 2 8 8 8 8 8 8 8 9 9 ?
5 × 5 median:  ? ? 1 2 1 2 2 2 2 2 2 2 2 2 2 2 8 8 8 8 8 8 8 9 ? ?
```

The runs of constant value are obvious. The spike of value 9 has shifted the 5 × 5 median but not the 3 × 3 median. The rule is that shifts occur when a spike and a point adjacent to an edge appear within an *n*-element window.

3.6

(a) On the background side of any edge, the background intensity dominates, so the mode filter comes up with a central background intensity value. Similarly, it comes up with a central foreground value on the foreground side of any edge. There is a critical point on any edge where the dominant peak in the local intensity distribution changes over from the background peak to the foreground peak or vice versa. Thus the mode filter tends to enhance edges. The mean filter blurs edges since it produces an intensity profile in which background and foreground intensities are mixed together, thereby reducing the local contrast between the two regions.

(b) When a max filter is applied to an image, light background regions expand, and will tend to reduce the size of objects, or even to eradicate small objects. It will also tend to produce regions of constant (high) intensity in the image. In general, the mode filter will tend to preserve edges and not move them substantially to reduce the size of objects. On the other hand, at the corners of objects, or on high curvature boundaries, the mode filter will tend to cut into objects. However, unlike the max filter, it will act symmetrically

between foreground and background, so it will also cut into small regions of background on high curvature boundaries.

(c) The purpose of the median filter is to suppress noise by eliminating local outliers in intensity. In addition, it aims (successfully) to achieve this while not producing any local image blurring. The median filter is highly computation intensive, with computation proportional (at least) to the area of the window it operates in: to offset this, it is common to implement a 2-D median filter as two 1-D filters, with a result that is commonly almost 100% effective.

(d)
```
orig:  0 1 1 2 3 2 2 0 2 3 9 3 2 4 4 6 5 6 7 0 8 8 9 1 1 8 9
mean:  ? ? 1 2 2 2 2 2 3 3 4 4 4 4 4 5 6 5 5 6 6 5 5 5 ? ?
max:   ? ? 3 3 3 3 3 3 9 9 9 9 9 6 6 6 7 7 8 8 9 9 9 9 9 ? ?
med1:  ? ? 1 2 2 2 2 2 2 3 3 3 4 4 4 5 6 6 6 7 8 8 8 8 8 ? ?
med2:  ? ? ? ? 2 2 2 2 2 3 3 3 4 4 4 5 6 6 6 7 8 8 8 ? ? ? ?
med3:  ? ? ? ? ? ? 2 2 2 3 3 3 4 4 4 5 6 6 6 7 8 ? ? ? ? ? ?
```

Clearly, the mean filter averages all the noisy values into the signal, while the median excludes them.

(e) Further applications of a median filter in this case have no further effect. The max filter spreads all high values over a wide range and also broadens any light spots.

3.7

(a) The 3 × 3 median filter removes the noise points (including the bump) and also removes one corner point. The 5 × 5 median filter removes the noise points (including the bump) and three corner points.

(b) To obtain a corner detector, first use the 3 × 3 median filter to remove the noise points; then use the 5 × 5 median filter to find the corner points. This method has numerous variations. It does not depend on orientation so is intrinsically better than template matching. However, it requires significant computation in its own right.

3.8

(a) Mean filters average and return the mean intensity in the window; median filters return the median intensity in the window. Mean filters average in all values blindly, without discriminating against outliers. Median filters ignore outliers and hence don't blur images.

```
original: 1 1 1 1 2 1 1 2 3 4 4 0 4 4 4 5 6 7 6 5 4 3 3
median:   1 1 1 1 1 1 1 2 3 4 4 4 4 4 4 5 6 7 6 5 4 3 3
```

(b) The median algorithm is bookwork (see the main text in Chapter 3). It operates slowly mainly because it has to go steadily through the histogram entries for something like half of the 256 possible intensity values before it arrives at the median.

(c) By cascading max operations, the maximum value can be found in 8 operations. One operation then sets the maximum to zero. Repeating this a further 4 times, but not in the last case setting the result to zero, gives the median. This involves a total of $9 \times 5 - 1 = 44$ operations. This should be compared with $\sim 9 + 256/2 + 9 = 146$ operations by the standard method, so the max approach is much faster.

(d) However, finding the max in a 1×3 window takes 2 operations, and then finding it again takes 1 operation, after setting the first max to zero. Thus the overall operation takes 6 operations—vastly quicker than 146 operations. The splitting up of the median does not affect its capability for eliminating single intensity impulses, though things get more complicated when their density is high.

3.9

(a) Results:

(i)	0 0 0 0 0 1 0 1 1 1 1 1 1 1	(input)
	0 0 0 0 0 0 1 1 1 1 1 1 1 1	(output)
(ii)	2 1 2 3 2 1 2 2 3 2 4 3 3 4	(input)
	? 2 2 2 2 2 2 2 2 3 3 3 3 ?	(output)
(iii)	1 1 2 3 3 4 5 8 6 6 7 8 9 9	(input)
	1 1 2 3 3 4 5 6 6 6 7 8 9 9	(output)

(b) General lessons:
- (i) Median filters can eliminate impulse noise, but may introduce edge shifts. This also applies to 2D images.
- (ii) There is a tendency to produce runs of constant value.
- (iii) Apart from removing impulse noise, median filters tend to leave monotonically increasing/decreasing signals unchanged.

(c) See the main text in Chapter 3.