1.
    a.
        i.

```
double* getSmallestValue(double* pixel_data, int width, int height);
```

        ii.

```
double* getSmallestValue(double* pixel_data, int width, int height)
{
  int value;
  double ref[2];
  for (int i = 0; i < width; i++) {
          for (int j = 0; j < height; j++) {
                  if (i == 0 && j == 0) {
                          value = pixel_data[i,j];
                          ref[0,1] = i,j;
                  }
                  else if (pixel_data[i,j] < value) {
                          value = pixel_data[i,j];
                          ref[0,1] = i,j;
                  }
          }
  }
  return ref;
}
```

    b. High-pass filter alters an image so that there are no values close to 0 (black). This means any black in the image is changed to a dark grey (0.3-0.4), though white values will be left mostly alone, depending on how high the filter is.
    c. Unsigned bytes can be quickly used to represent any value from 0 to 255. Whereas floating point can represent a much larger range of numbers with a maximum value exceeding the millions. This also includes decimal and negative values, though it always requires 4 bytes of space regardless of the number currently held.
    d. Floating point would be better at maintaining the best dynamic range, though it may be better to use unsigned bytes as more dynamic range would be lost storing a floating point image into an unsigned byte file format, than just using unsigned byte for the image from the start.

2.
    a.
        i. A high-pass filter was used to remove low frequencies, like when using the Roberts Operator. It has a vertical gradient.
        ii. A gaussian filter was used to create this, as the details around the edges of the shapes appear more blurred than the original.
        iii. A high-pass filter was used to isolate the details around the edges of the shapes, like the Laplacian Operator.
    b.
        i. High-pass filter
        ii. Low-pass filter
        iii. High-pass filter

3. A first step would be to get an image from the camera's view when no people are present by the cash machine, as this would give a good background to help detect people in the area. This could be updated every hour or every time the image as a whole has changed

significantly by light level (due to weather or time), during another period where there are no people. Once this is established, it can recognise when a person is present at a cash machine. Queues are likely to form, so it would have to tell differences as people move along and are replaced by others. Should the camera detect someone by the machine, and they don't move for 20 minutes, then the alarm should be triggered as this is a highly extended time period to spend at a machine, and likely means they are asleep. If someone moves significantly (i.e. not mistaken for just moving in their sleep) during these 20 minutes, it should reset the timer immediately as to not trigger the alarm when someone who was accessing the machine 15 minutes ago is replaced by someone at the machine for more than 5 minutes.