```

// convolve the given image with array of weights
PImage convolution(PImage img, float[][] wgt, float offset) {
  PImage result = createImage(img.width, img.height, RGB);
  for (int x=0; x < img.width; x++) {
    for (int y=0; y < img.height; y++) {
      result.set(x, y, convolvePixel(img, x, y, wgt, offset));
    }
  }
  return result;
}

// compute convolution result for pixel (x,y)
color convolvePixel(PImage img, int x, int y, float[][] wgt, float offset) {
  int h = wgt.length;      // number of rows in weight matrix
  int w = wgt[0].length;   // number of entries per row of weight matrix
  float r, g, b, total;    // total used for normalization
  r = g = b = total = 0;

  for (int i=0; i < w; i++) {
    int newX = x + i - w/2;
    if (newX >= 0 && newX < img.width) {  // otherwise out of bounds
      for (int j=0; j < h; j++) {
        int newY = y + j - h/2;
        if (newY >= 0 && newY < img.height) { // otherwise out of bounds
          color c = img.get(newX, newY);
          r += red(c) * wgt[i][j];
          g += green(c) * wgt[i][j];
          b += blue(c) * wgt[i][j];
          total += wgt[i][j];
        }
      }
    }
  }

  if (total == 0) {
    total = 1;  // avoid division by zero errors for neutral kernels
  }

  return color(offset + r/total, offset + g/total, offset + b/total);
}

```