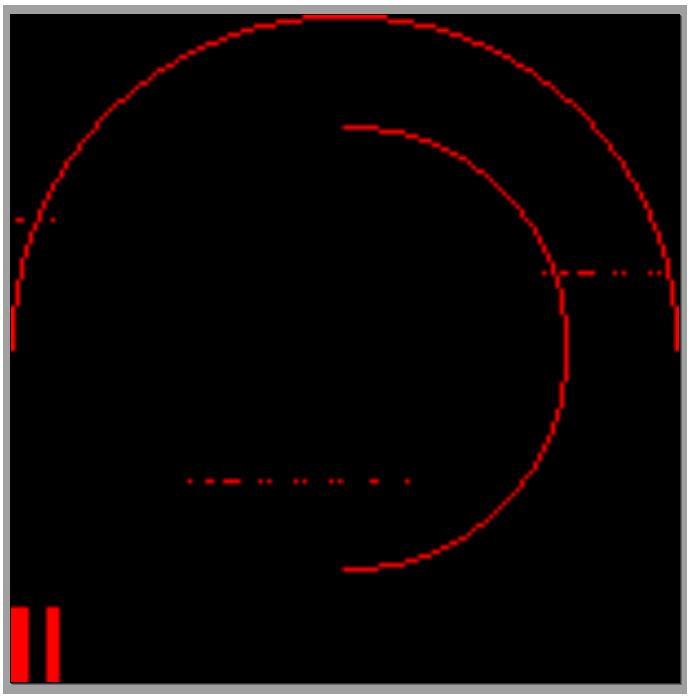
Hw1 required me to create two half circles one with the radius of 150 pixels and the other with a radius of 100 pixels. To create this, I used the midpoint circle algorithm. The algorithm takes a point, lights up the pixel then increments while choosing the best choice of pixel to the ideal circle and lighting up. To The algorithm calls a function (called renderPixel in this case) to light up the pixels. In addition to lighting up the coordinate given, it also lights up any symmetrical point needed. Which in this assignment the needed points are (y,x), (-x,y), (-y,x), (y,-x,), and (x,-y).

However, multiple issues arise in the code given, one being the size of the output file is set to the radius, so scaling of the circles are needed, and the other being the negative x's and y's cannot be used. To fix the first issue we divide all the r's in the midpoint circle algorithm besides for y = r; this scales down the circles. So now that the circle is scaled down properly the origin needs to be moved to the center of the output file which can be explained along with the fix to the second issue. The alternative to using negative x's and y's is to use basic math to adjust the location of the points. We can group the math used to get the renderPixel function to light up the correct pixels. For example instead of using image[-x][y] we use image[x][r-y] (r being the radius given by the algorithm) and for the second circle instead of image[x][-y] we use image[r-x+25][y+25] the addition of 25 to the x and y is used to center the second circle. The result is:



I believe the horizontal dotted lines along with the two columns in the bottom left is an error on my end of things. But the circles are rendered how intended with the larger circle in quadrant 1 and 2 being the circle with the scaled radius of 150, and the smaller circle in quadrant 1 and 4 being the circle with the scaled radius of 100.