Benjamin Berger
Bberger3@binghamton.edu

The purpose of the project was to get a better upstanding of simple transformations in images.

For the inverting the image, first I made sure to take the image in as greyscale. Then for each pixel, I subtracted that value from 255. The resulting values were the inverted pixels that I used for the modified image.

For the histogram equalization, for I created the initial histogram. Then I applied the cdf function to outline the point. Afterwards I applied the histogram equalization function as described by wiki
https://en.wikipedia.org/wiki/Histogram_equalization

For the window/level adjustment I didn't really know what to do.

For the connected component labeling, first I create a binary image from the original image. Then the idea what to go through each pixel and check it against the one above and just before it. It would always take the lower of the two if applicable, and create a map saying that the higher should become the lower. Then it would loop through that map until all cases were handled.

BUG REPORT
Part 2
Doesn't really do much.
Part 3.
Couldn't get the end result after creating a binary image to work right.

References:
The skeleton code I used was from Brad the TA.

I used wiki to set up the histogram equalization.
I used otsu method to create a binary image.
Otsu code found here http://www.dandiggins.co.uk/arlib-9.html

TO RUN CODE!
Open in terminal and type make.
It will make 4 files.
p1-1 corresponds to programming part 1 subsection 1 and 2.
p1 -3 corresponds to programming part 1 subsection 3 and 4.
p2 corresponds to programming part 2

p3 corresponds to programming part 3

Run them as ./p1-1 [name of image].

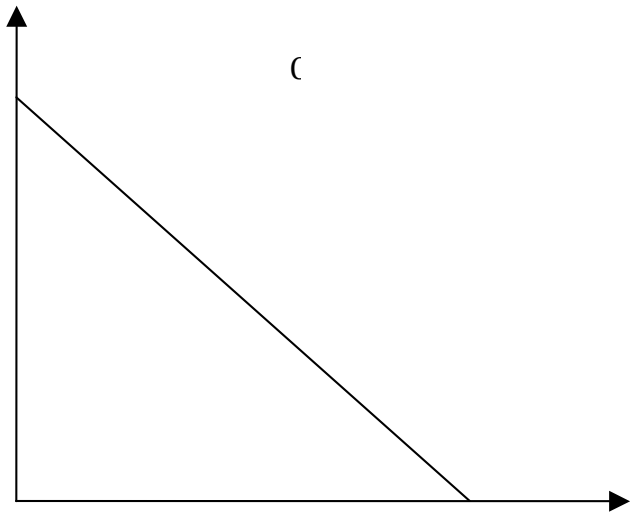
Press spacebar to cycle between original and modified..

Project 1 Write up.

Part 1:

Part 2:
Here is the function I used to achieve the inverted image.

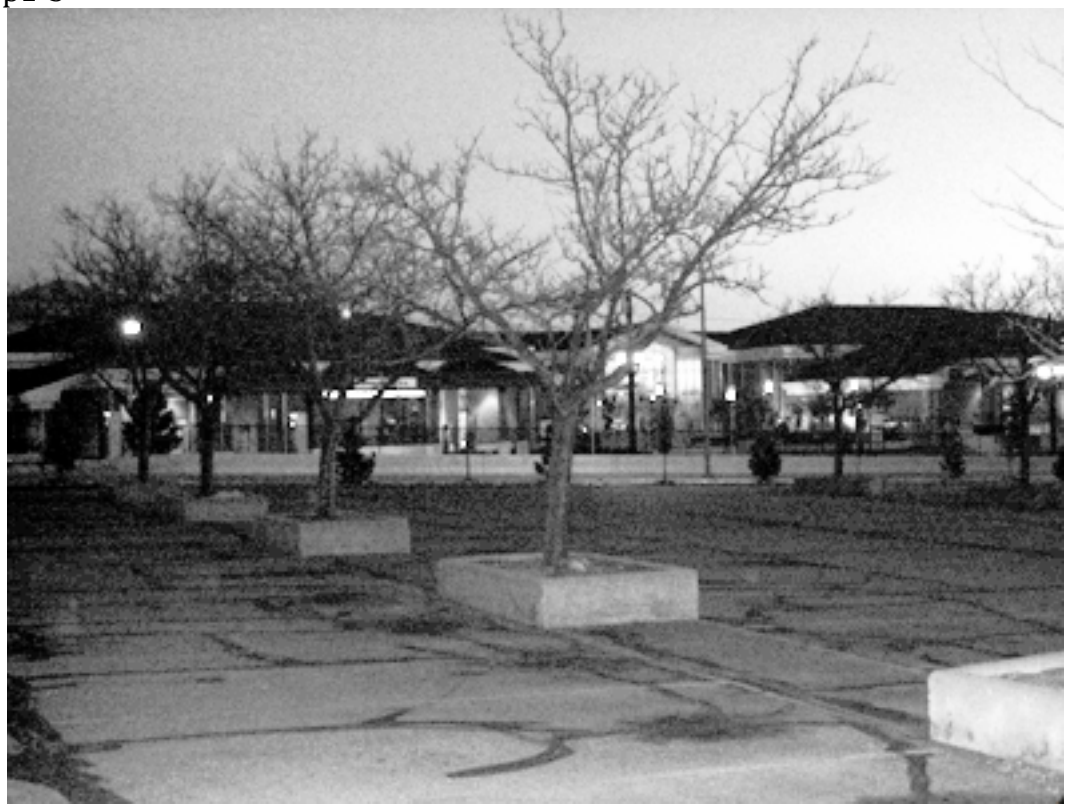
```
for(size_t y = 0; y < original_image.rows; y++){
   for(size_t x = 0; x < original_image.cols; x++){
         modified_image.data[y*original_image.cols +x] = 255 –
                        original_image.data[y*original_image.cols +x];
   }
}
```



Transformation curve

(2) Histogram for image and its negative are printed to the console when running p1-1

(3) Histogram for image and its negative are printed to the console when running p1-3



(4)

Part 3

The idea what to go through each pixel and check it against the one above and just before it. It would always take the lower of the two if applicable, and create a map saying that the higher should become the lower. Then it would loop through that map until all cases were handled.

It takes a few seconds to run.

(5)

(6) the threshold I got was 54.