Maryfrances Umeora
CSCI 433 Section 2
Project Report

"Provide a convincing explanation of your method and rationale for design and implementation decisions."

Since the main problem of this program was to grow a region around a pixel, the first thing I did was create a function that does the specific act of returning a queue of the eight pixels surrounding a given pixel. Basically, I checked each possible neighbor one by one, resulting in eight separate if-statements. Maybe not the prettiest code, but it does the job. At first, this list only contained the value of those pixels, rather than the pixels (including coordinates) themselves.

Next, I continued to the main function to try and implement the rest of the algorithm. After initializing a boolean 2D array the same size as the image (also a 2D array), I loop through the entire image array, searching for an unvisited pixel of the target value. Once I find one, I mark its coordinates as "visited" and increase my number of regions variable by one.

Since I am in a new region at this point, I initialize a queue of seedpoints containing all the neighbors of the seed. It is at this point that I realized that I need not the value of the neighboring pixels, but their coordinates, and after finding out that *tuples* don't exist in Java, I decided to go the object oriented programming route and create a Seed object. A seed's only attributes are its coordinates, seedR for y and seedC for x.

Now that I have a working queue of neighbors, I dequeue and examine the neighbors one by one, marking them as visited and growing the region around them as I go along. I also make sure to increment my number of pixels variable, which represents the size of the region, as I go along.

Once the queue is empty, I know that it means there is nothing left in that region to explore, so I add my region size to an array list of region sizes and start the whole process over again.

After the entire image has been processed, I sort my array list of sizes and print out my results.