The problem was to use recursion to draw a series of images in python. There was a total of 4 images that needed to be drawn using code that was provided by the professor. By utilizing the matplotlib and numpy libraries one can plot geometric shapes using python.

I attempted to find a solution to this problem by breaking the problem into different parts. For example, for the first figure where a square’s vertex was the center of a smaller square, four in total, I tried to create just one smaller square to appear in one corner. Once this was completed, I used recursion to create another small square in another corner. A total of four recursion calls were made to def square which the initial call makes the big square and the four recursive calls afterwards make the four smaller squares in each corner. For the next figure, the circles within the circle, I followed the same method and broke it down. I first attempted to create one smaller circle within the circle. In this case, only one recursion call was made and by changing the n values of def draw\_circles I was able to obtain the figures requested. For the third figure, I also implemented the same idea and split the semi-completed triangles in two parts, the left side and the right side. I attempted to create the left side first which should be narrower from the top one. Two calls were made for this figure, one call make the first triangle and from there a call is made to the lower triangles. For the last figure, the same approach was used. From the first circle I created a smaller circle and went from there. There are two recursion calls for this method.

As far as how the experiments went, for the squares each call had a change in n and a change in the coordinates. The first smaller square which is the first recursion call changes the coordinates to the new q variable which multiplies the original coordinates, which is the original big square, by 0.5 which reduced the size of the square. This amount is then subtracted by 400 to move the smaller square to the third quadrant and have the vertex of the original square as its center. The second call creates an inverse on the q variable which makes a smaller square appear on the first quadrant. The same step is created for the last two calls with the only change being that the p coordinates are multiplied by 0.5 and then added by the first element of the p coordinates. This creates a mirror image on of the q coordinates in the x axis and this variable is named t. The last call is the inverse of t which makes the fourth smaller square appear in the 4th quadrant.

For the second figure, the same code provided by the professor is used with the only changes done to the initial call of the draw\_circles method. The arguments changed are the n and the w depending on how many circles one wants. Within the draw\_circles method, the first element of the center variable is changed by multiplying it by whatever value of w is selected. The arguments for the three required images are, (ax,10,[100,0],100,.6), (ax,50,[100,0],100,.9), and (ax, 110, [100,0], 100,.96) respectively.

For the binary tree image two method calls are used. The first call, draw\_triangles, crates the first image which would be considered the root of a binary tree. The next call, triangles, creates the child of the root all the way to the leafs. The p array is used to create the coordinates and then calls draw\_triangle with the same variables the professor’s code used with the only changes being on n and w. Draw\_triangle then calls the triangles method where the narrower triangles are made. The recursion for the triangles depends on n and q variables. The method itself plots the neg and pos representing the third quadrant and fourth quadrant respectively are the ones plotted within each recursive call.

For the final image, draw\_circles is the method called with only n and w variables changing within each recursion call. There are two recursion calls and the first element of the center variable is changed by multiplying it by w. The first call creates the big circle original circle the first and second call create the smaller circles within he big circle. By multiplying the first element of the center by .58 moves the smaller circle from the center to the side but still keeping the smaller circles within the bigger circle.

What I learned from this project is that recursion can come in handy when a big problem can be broken down to smaller parts. When creating the same image multiple times one must determine how many recursion calls one must make to create the desired image. Sometimes it is easier with one recursive call while sometimes it is needed to make several to solve the problem.

Appendix:

Line 12 to 31 figure 1

Line 33 to 56 figure 2

Line 58 to 77 figure 3

Line 33 to 39 and 79 to 92 figure 4