**CS2302 - Data Structures**

**Spring 2019**

**Lab Report**

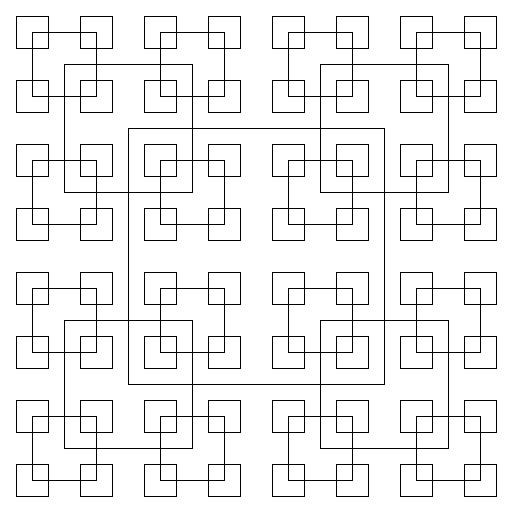
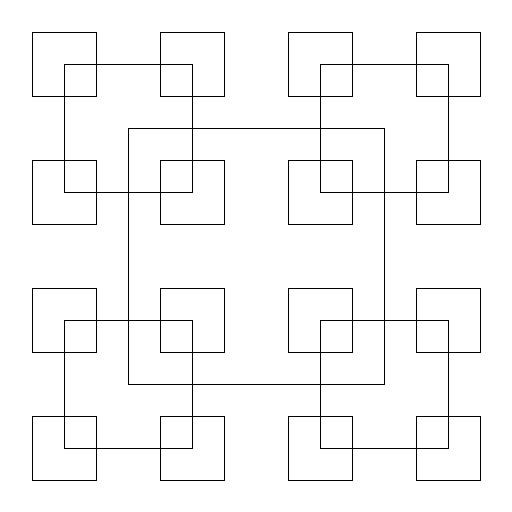
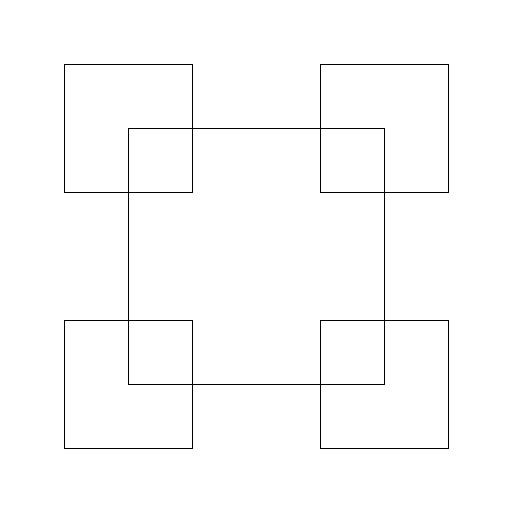
February 12, 2019

Sebastian Gomez

**Introduction:**

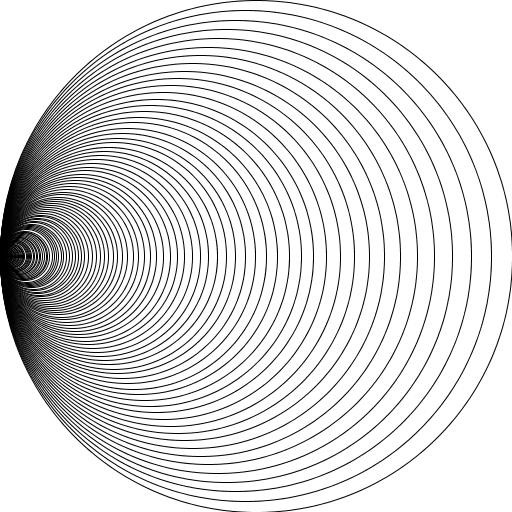
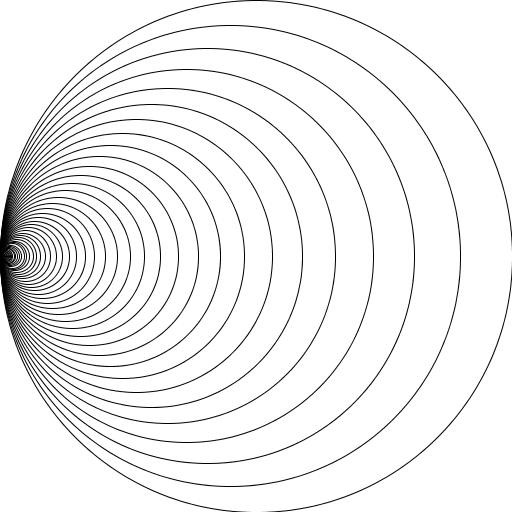
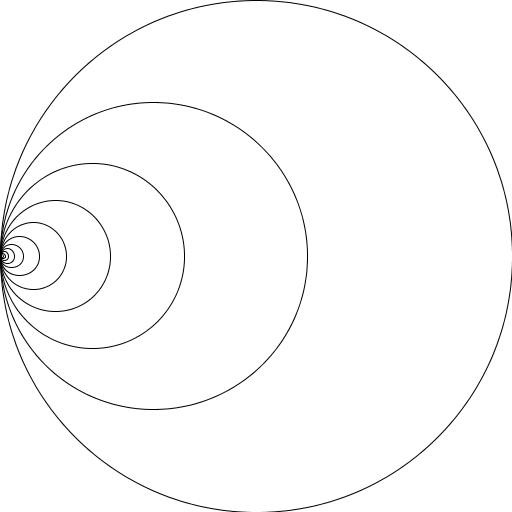
Based on the given code of the nested squares and concentric circles, create the following 12 figures using recursion.

1. Write a recursive method to draw the following figures:



* 1. b) c)

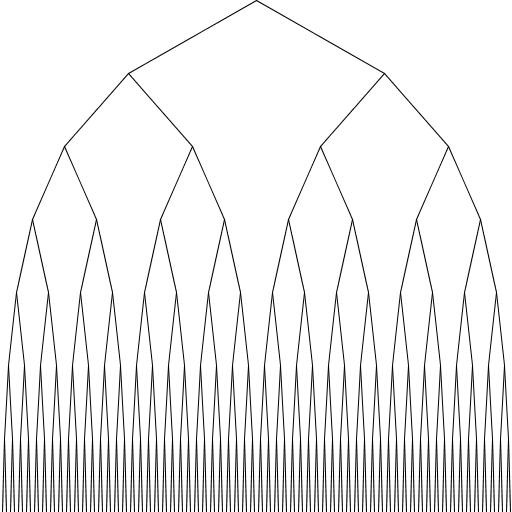
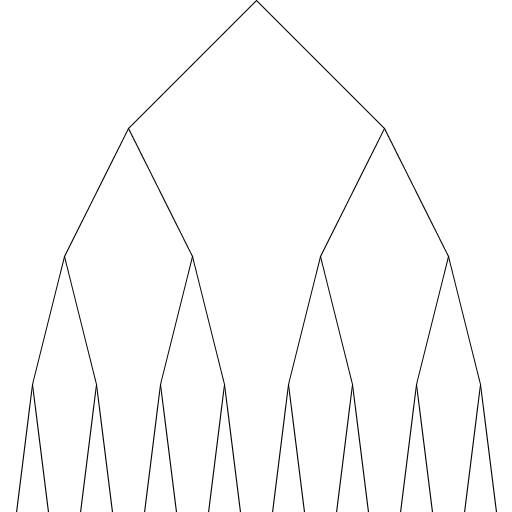
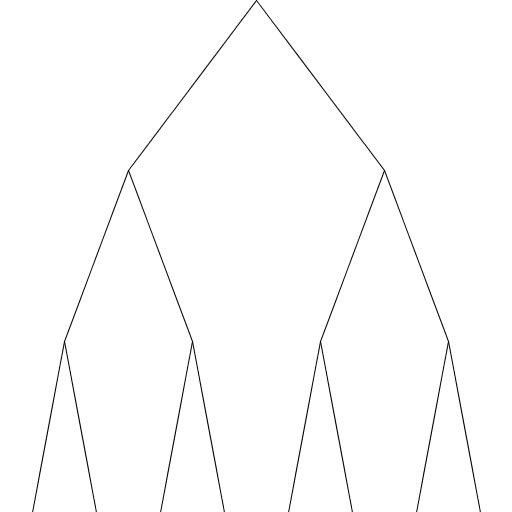
I noticed that by making the first figure, the other ones will be done by incrementing the number of times the program called the recursive method. I broke down this program into creating the first square and knowing the center and the radius of the square. Then the coordinates of the corners of the square were obtained by doubling the radius in x-coordinates and y-coordinates. Once completing the drawing of the square, by looking at the figure, I noticed that the corners are the new center for the yet to be drawn squares and as you wanted to increment the squares the pattern repeats itself. The size of the square was the radius divided by two and as you wanted to create more squares, you just kept dividing the radius by two. In my case, I did not know how to assign a center and a radius so I could not do the task assigned. I knew that it was similar to the last figure because of the changing center and radius but I could not figure it out.



* 1. b) c)

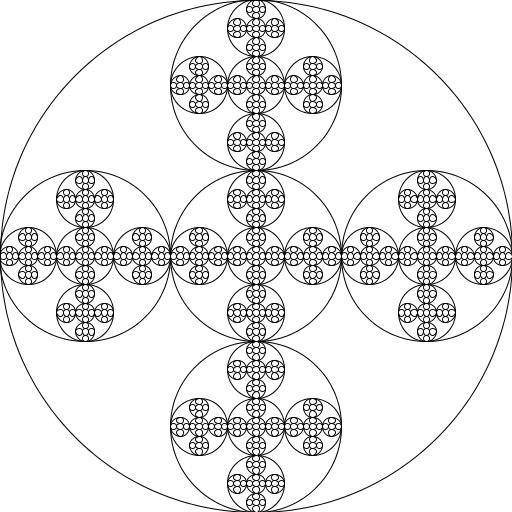
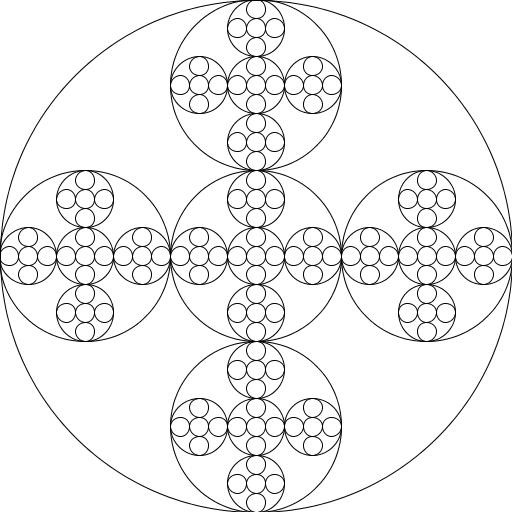
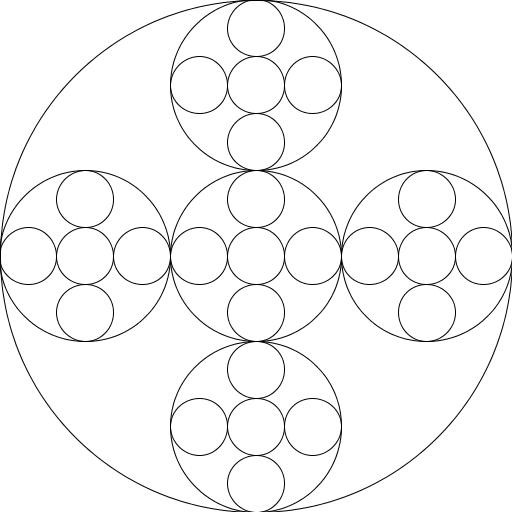
The figure above I divided it into creating one circle and then changing the center and the radius by multiplying it by the percentage. Every time a circle was created, the radius changed. The center was changed only by the x-coordinate and was by subtracting from the original center the radius times the percentage and with every recursive call it changed. These changes were made on the method draw\_circles in order to do the task.

The inputs for getting the first figure are: n = 8, center = [1000,0], radius = 100, w = .6. The run time is 0.13842129707336426. The inputs for getting the second figure are: n = 50, center = [1000,0], radius = 100, w = .9. The run time is 0.3436703681945801. The inputs for getting the last figure are: n = 100, center = [1000,0], radius = 100, w = .94. The run time is 0.8007369041442871



* 1. b) c)

For these one, the origin was the one on the top, and the number of recursive calls was the depth or levels of the binary tree. The parameter were x-coordinates, y-coordinates, the average of x and y, and n which is the number of times the recursive call will call itself. For these one I needed to draw the first two branches and when you went to the left branch, you subtracted to the x-coordinate the average of x, and to the y-coordinate the average of y and as you keep going down, I needed to divide the average by two, as for the right branch it was the similar but instead of subtracting on the x-coordinate, you added the average. To change from figure to figure you just incremented the number of recursive calls. For these one I did not know how to coded.



* 1. b) c)

In these one I divided the problem into drawing the big circle, identify its radius and center and then start the calculations to get the other circles. I also noticed that the circles on a horizontal way, were always on the same x-coordinate and the vertical in the middle were on the same y-coordinate. By doing some calculations, I noticed that to draw the 5 circles inside the big circle, I needed to change the radius and the center, the fixed operations was that to get the left circle you needed to subtract from the center 2/3 times the radius because from the center to the new center of the left circle you moved 2 radiuses out of 3. For the circle at the middle I did not change the coordinates, for the right, instead of subtracting the product of 2/3 times the radius, I added it. It applied the same for the y coordinate, to do the upper circle, you added 2/3 times the radius but in the y-coordinate, not on the x-coordinate, and for the lower circle you subtracted the 2/3 times the radius. In every circle made, you also divided your radius by 3. Now that you knew how to draw the 5 circles inside the big circle, the rest of the circles were the same but smaller, so just by calling the 5 recursive calls again changing every time the center and radius, the figures were possible. I did manage to create the 5 circles but I did not know how to create a method to just change the number of recursive calls to de the figures.

**Conclusions:**

I had never before drawn on a programming language. This was my first time and unfortunately, I did not do well on this lab. I learned the concept of recursion but not how to implement it completely on all of the figures. I hope that I can learn more about how to draw so I can improve.

**I certify that this project is entirely my own work, I wrote, debugged, and tested the code being presented, performed experiments, and wrote the report. I also did not certify that I did not share my code or report or provided inappropriate assistance to any student in the class.**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Feb 12 17:35:14 2019

@author: Sebastian Gomez

Course: Data Structure 2302

Assignment: Lab 1

Instructor: Olac Fuentes

T.A: Anindita Nath and Maliheh Zargaran

Purpose: Use of recursive calls to plot and create complex figures

"""

import numpy as np

import matplotlib.pyplot as plt

import math

import time

global start

global end

def draw\_squares(ax,n,p,w,center,radius):

if n>0:

i1 = [1,2,3,0,1]

q = p\*w + p[i1]\*(1-w)

ax.plot(p[:,0],p[:,1],color='k')

center = [0,0]

radius = p[:,0] / 2

draw\_squares(ax,n-1,q,w,center,radius)

start = time.time()

plt.close("all")

p = np.array([[-400,-400],[400,-400],[400,400],[-400,400],[-400,-400]])

fig, ax = plt.subplots()

center = [0,0]

draw\_squares(ax,1,p,.2,center,50)

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('squares.png')

end = time.time()

print(end - start)

def circle(center,rad):

n = int(4\*rad\*math.pi)

t = np.linspace(0,6.3,n)

x = center[0]+rad\*np.sin(t)

y = center[1]+rad\*np.cos(t)

return x,y

def draw\_circles(ax,n,center,radius,w):

if n>0:

x,y = circle(center,radius)

ax.plot(x,y,color='k')

draw\_circles(ax,n-1,[center[0] - (radius\*(1-w)),0],radius\*w,w)

start = time.time()

plt.close("all")

fig, ax = plt.subplots()

draw\_circles(ax,14,[200,0],200,.8)

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('circles.png')

end = time.time()

print(end - start)

start = time.time()

plt.close("all")

fig, ax2 = plt.subplots()

r = 200

center = [0,0]

draw\_circles(ax2,1,center,200,.8)

draw\_circles(ax2,1,[center[0] - (2/3)\*r,0],r/3,.1)

draw\_circles(ax2,1,[center[0] + (2/3)\*r,0],r/3,.1)

draw\_circles(ax2,1,[center[0],0],r/3,.1)

draw\_circles(ax2,1,[center[0],center[0] - (2/3)\*r ],r/3,.1)

draw\_circles(ax2,1,[center[0],center[0] + (2/3)\*r ],r/3,.1)

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('circles.png')

end = time.time()

print(end - start)

def draw\_tree(ax,n,p,w):

if n>0:

i1 = [1,2,3,0,1]

q = p\*w + p[i1]\*(1-w)

ax.plot(p[:,0],p[:,1],color='k')

draw\_tree(ax,n-1,q,w)

start = time.time()

plt.close("all")

size = 100

p = np.array([[0,0],[size,-size],[0,0],[-size,-size],[0,0]])

p = np.array([[0,0],[size/2,-size/2],[0,0],[-size/2,-size/2],[0,0]])

fig, ax = plt.subplots()

draw\_tree(ax,1,p,.1)

ax.set\_aspect(1.0)

ax.axis('on')

plt.show()

fig.savefig('squares.png')

end = time.time()

print(end - start)