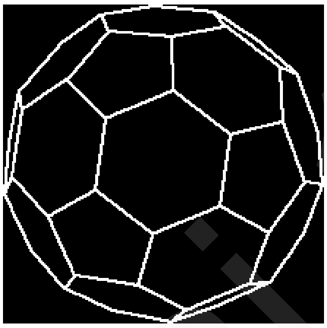
Batch: C2 Roll No.: 110

Experiment No. 10

TITLE: Write a program to draw “Buckyball” using openGL library.

AIM:

Write a program to draw “Buckyball” using openGL library.



Expected OUTCOME of Experiment:

To understand how to make 3d objects in openGL

Books/ Journals/ Websites referred:

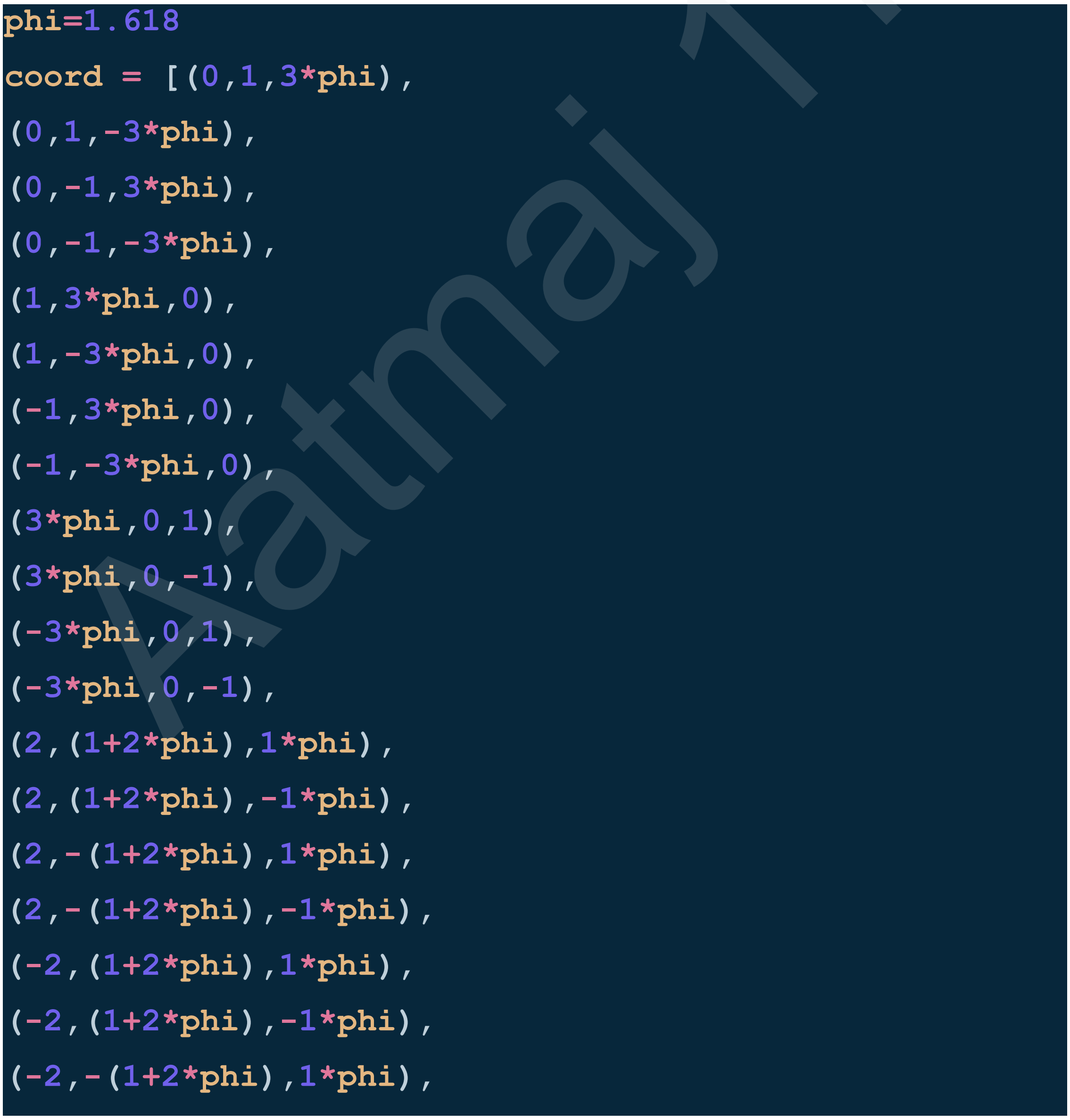
https://www.goldennumber.net/bucky-balls/

https://chat.openai.com

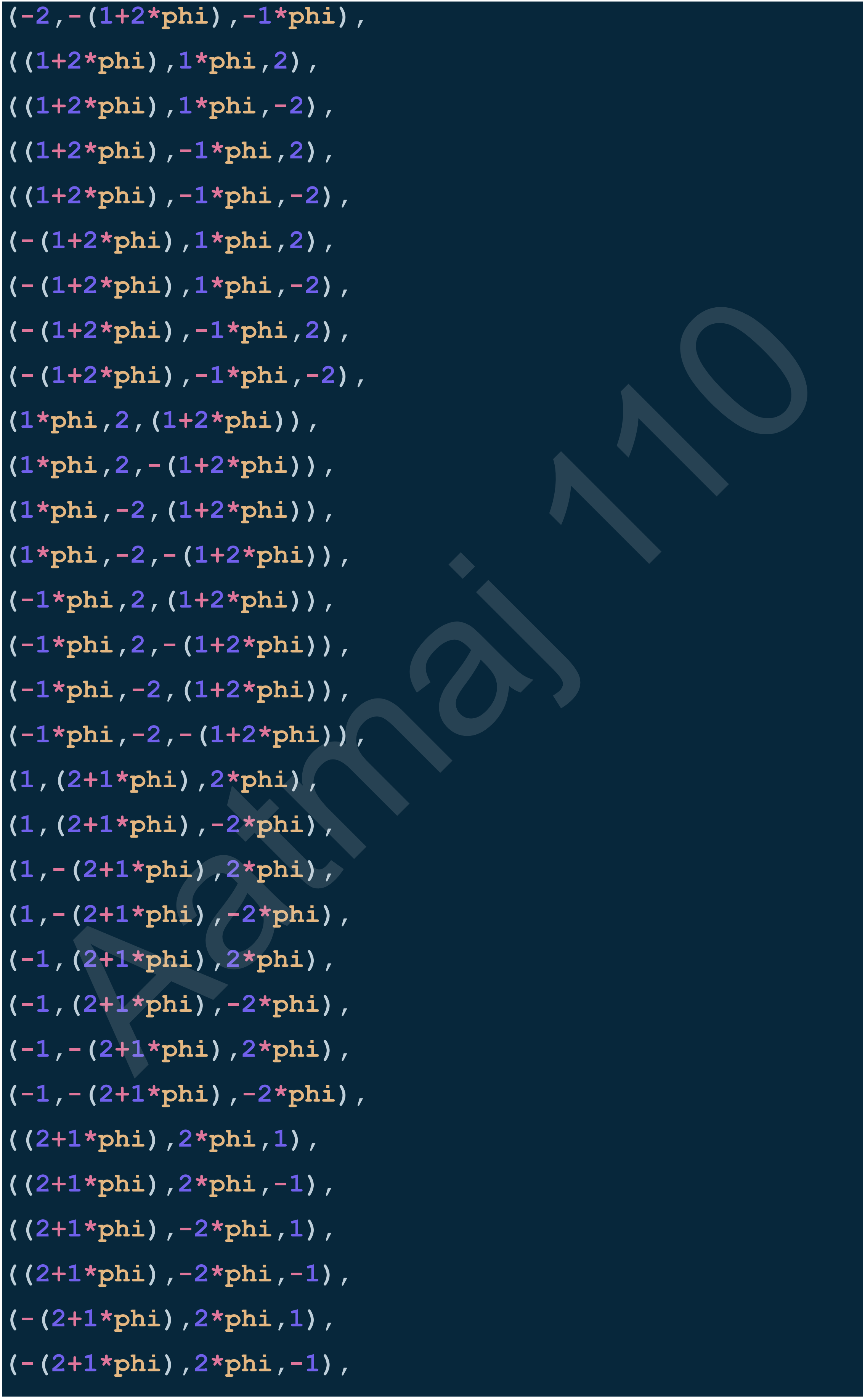
Algorithm/ Pseudocode for each process:

1) **Plot all coordinates of buckyball**
2) **Find coordinates that have distance less than phi/2**
3) **Draw lines of such coordinates**

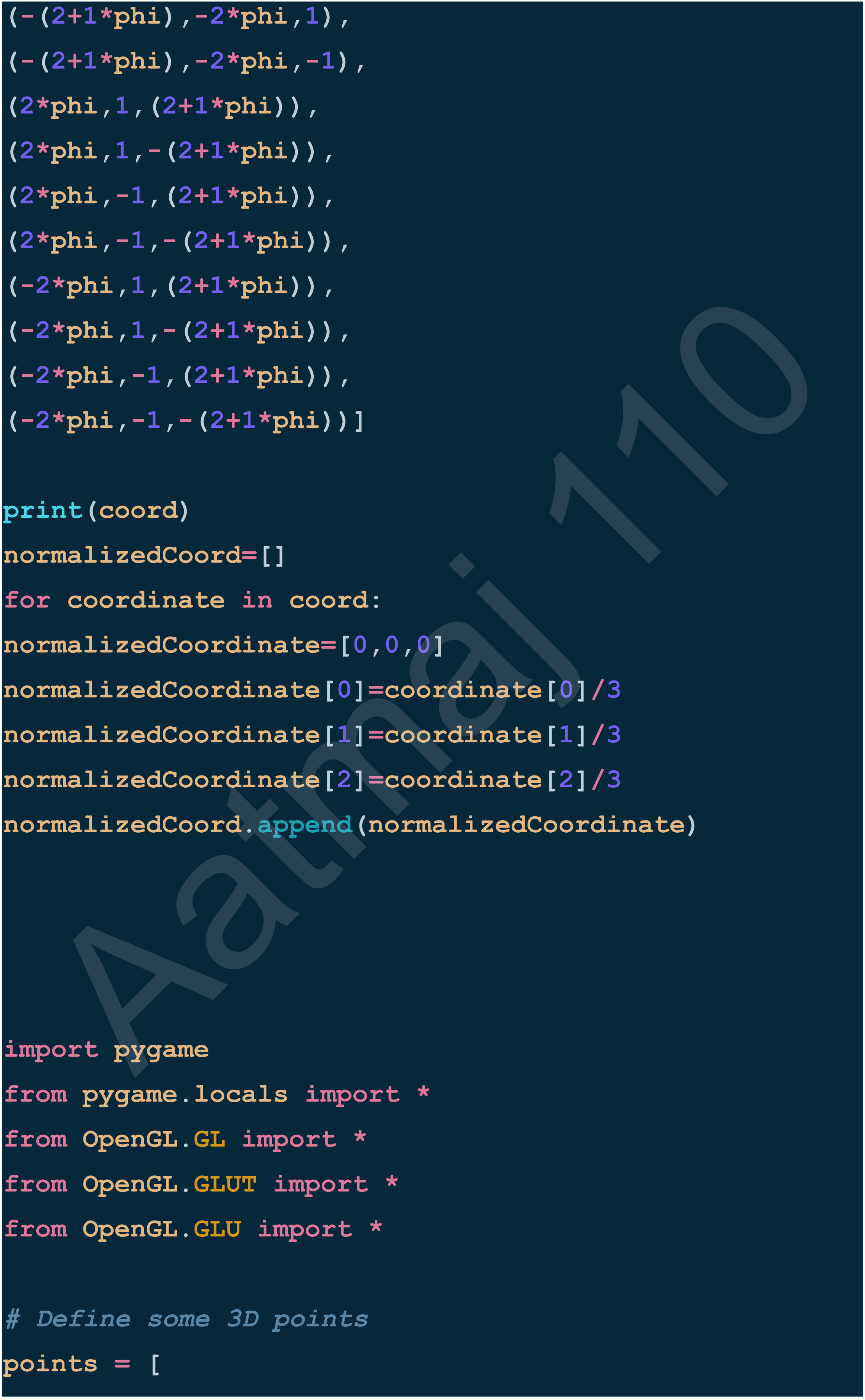
Implementation details:



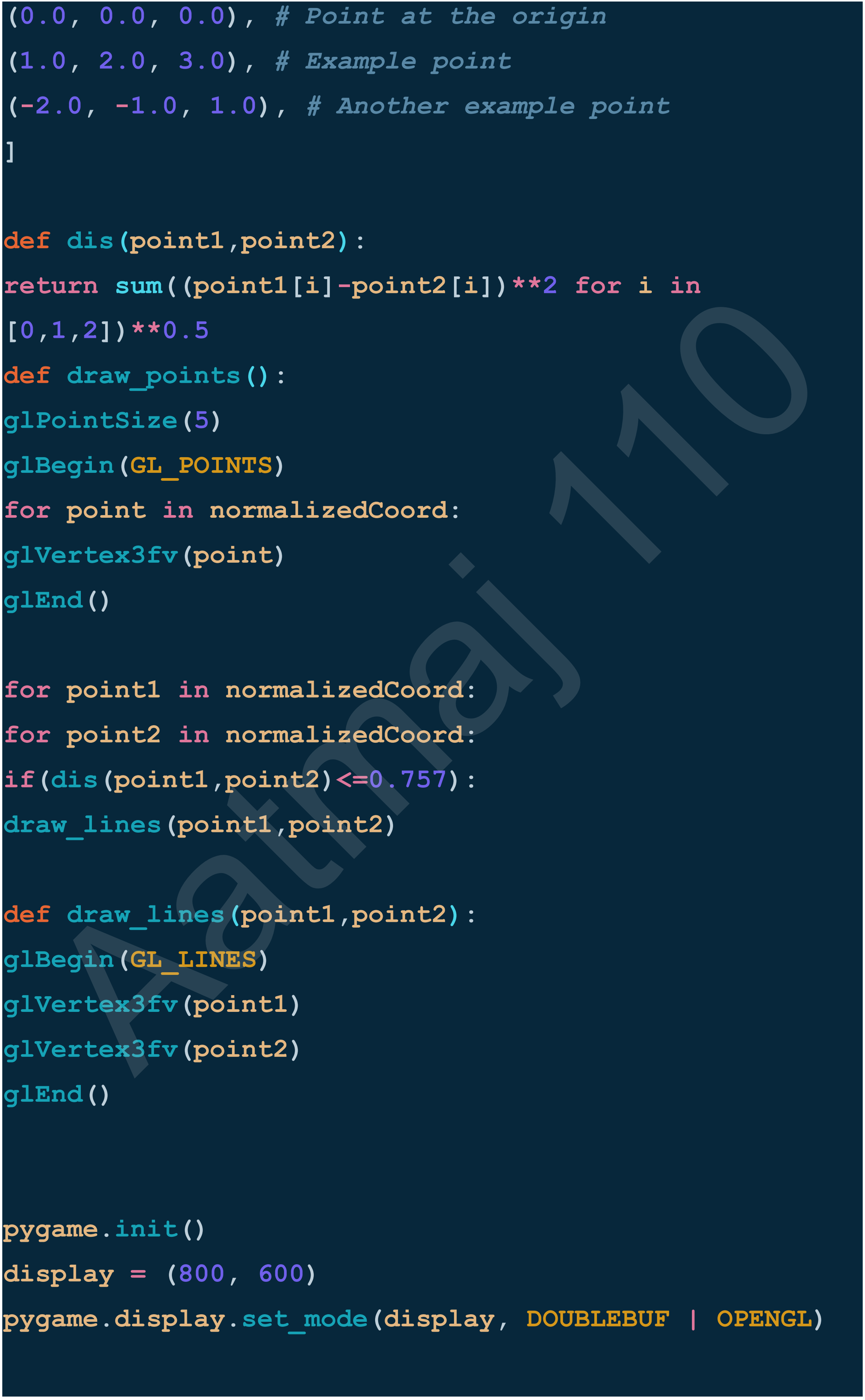
```
phi=1.618
coord = [(0,1,3*phi),
(0,1,-3*phi),
(0,-1,3*phi),
(0,-1,-3*phi),
(1,3*phi,0),
(1,-3*phi,0),
(-1,3*phi,0),
(-1,-3*phi,0),
(3*phi,0,1),
(3*phi,0,-1),
(-3*phi,0,1),
(-3*phi,0,-1),
(2,(1+2*phi),1*phi),
(2,(1+2*phi),-1*phi),
(2,-(1+2*phi),1*phi),
(2,-(1+2*phi),-1*phi),
(-2,(1+2*phi),1*phi),
(-2,(1+2*phi),-1*phi),
(-2,-(1+2*phi),1*phi),
```



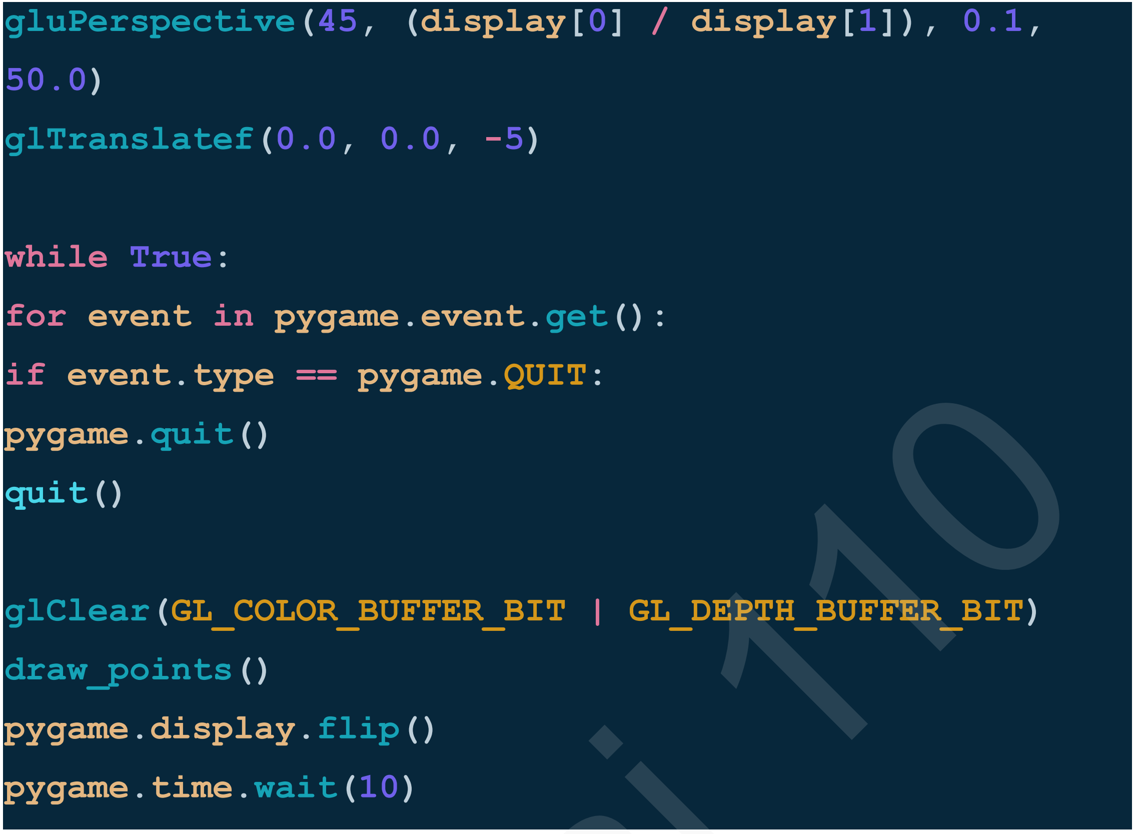
```
(-2,-(1+2*phi),-1*phi),
((1+2*phi),1*phi,2),
((1+2*phi),1*phi,-2),
((1+2*phi),-1*phi,2),
((1+2*phi),-1*phi,-2),
(-(1+2*phi),1*phi,2),
(-(1+2*phi),1*phi,-2),
(-(1+2*phi),-1*phi,2),
(-(1+2*phi),-1*phi,-2),
(1*phi,2,(1+2*phi)),
(1*phi,2,-(1+2*phi)),
(1*phi,-2,(1+2*phi)),
(1*phi,-2,-(1+2*phi)),
(-1*phi,2,(1+2*phi)),
(-1*phi,2,-(1+2*phi)),
(-1*phi,-2,(1+2*phi)),
(-1*phi,-2,-(1+2*phi)),
(1,(2+1*phi),2*phi),
(1,(2+1*phi),-2*phi),
(1,-(2+1*phi),2*phi),
(1,-(2+1*phi),-2*phi),
(-1,(2+1*phi),2*phi),
(-1,(2+1*phi),-2*phi),
(-1,-(2+1*phi),2*phi),
(-1,-(2+1*phi),-2*phi),
((2+1*phi),2*phi,1),
((2+1*phi),2*phi,-1),
((2+1*phi),-2*phi,1),
((2+1*phi),-2*phi,-1),
(-(2+1*phi),2*phi,1),
(-(2+1*phi),2*phi,-1),
```



```
(-(2+1*phi),-2*phi,1),
(-(2+1*phi),-2*phi,-1),
(2*phi,1,(2+1*phi)),
(2*phi,1,-(2+1*phi)),
(2*phi,-1,(2+1*phi)),
(2*phi,-1,-(2+1*phi)),
(-2*phi,1,(2+1*phi)),
(-2*phi,1,-(2+1*phi)),
(-2*phi,-1,(2+1*phi)),
(-2*phi,-1,-(2+1*phi))]

print(coord)
normalizedCoord=[]
for coordinate in coord:
normalizedCoordinate=[0,0,0]
normalizedCoordinate[0]=coordinate[0]/3
normalizedCoordinate[1]=coordinate[1]/3
normalizedCoordinate[2]=coordinate[2]/3
normalizedCoord.append(normalizedCoordinate)




import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

# Define some 3D points
points = [
```

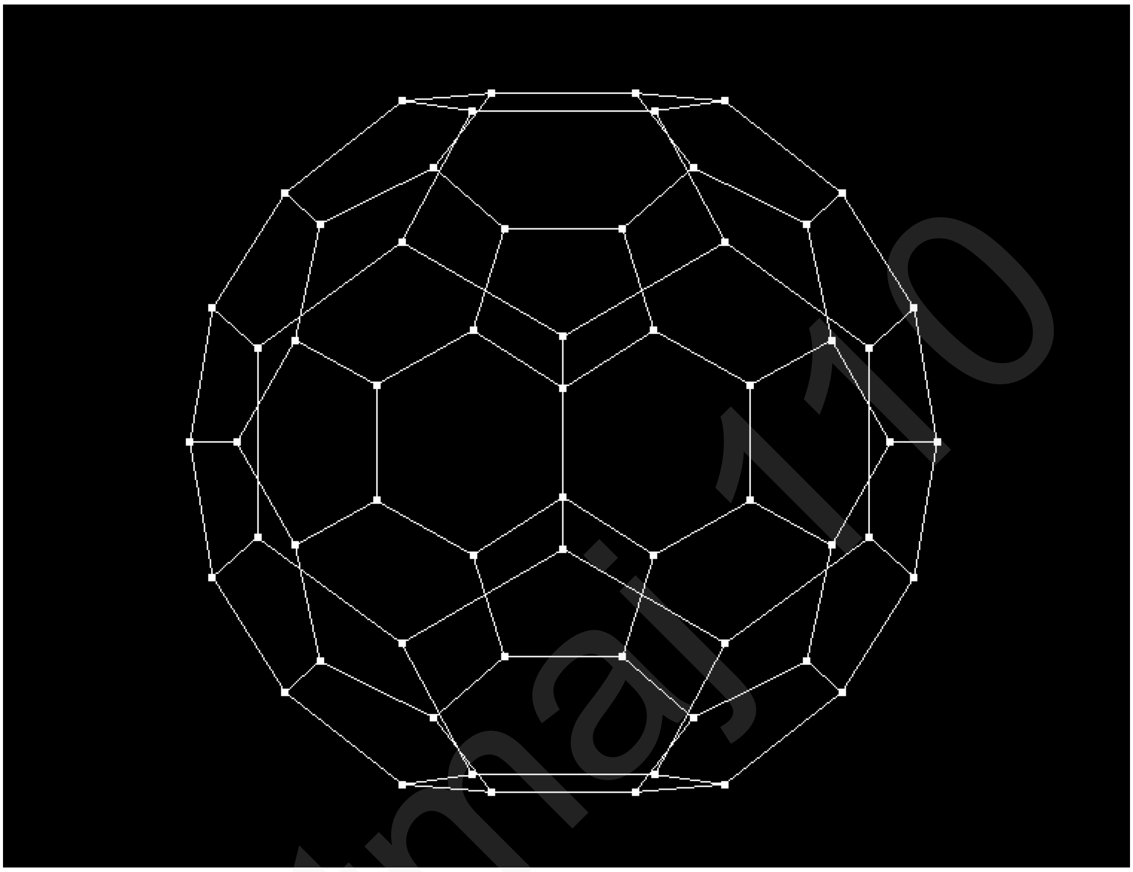


```
(0.0, 0.0, 0.0), # Point at the origin
(1.0, 2.0, 3.0), # Example point
(-2.0, -1.0, 1.0), # Another example point
]

def dis(point1,point2):
return sum((point1[i]-point2[i])**2 for i in
[0,1,2])**0.5
def draw_points():
glPointSize(5)
glBegin(GL_POINTS)
for point in normalizedCoord:
glVertex3fv(point)
glEnd()

for point1 in normalizedCoord:
for point2 in normalizedCoord:
if(dis(point1,point2)<=0.757):
draw_lines(point1,point2)

def draw_lines(point1,point2):
glBegin(GL_LINES)
glVertex3fv(point1)
glVertex3fv(point2)
glEnd()



pygame.init()
display = (800, 600)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
```



```
gluPerspective(45, (display[0] / display[1]), 0.1,
50.0)
glTranslatef(0.0, 0.0, -5)

while True:
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
draw_points()
pygame.display.flip()
pygame.time.wait(10)
```

Output(s) (Screen Shot):



Conclusion and discussion:

Thus we have implemented buckyball in opengl. We used the coordinates of buckyball that arise from golden ratio.

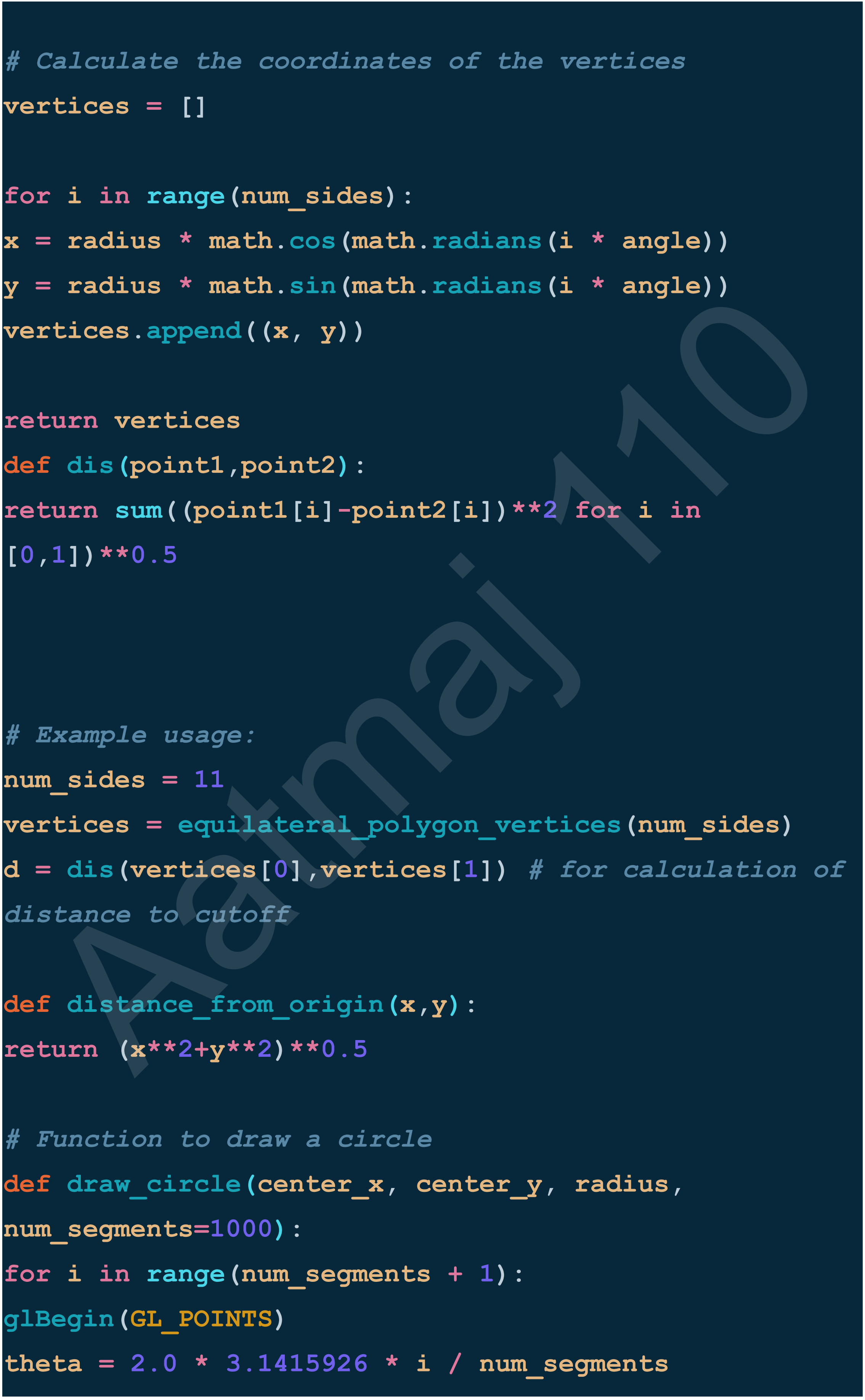
Date:4 oct 2023

Signature of faculty in-charge

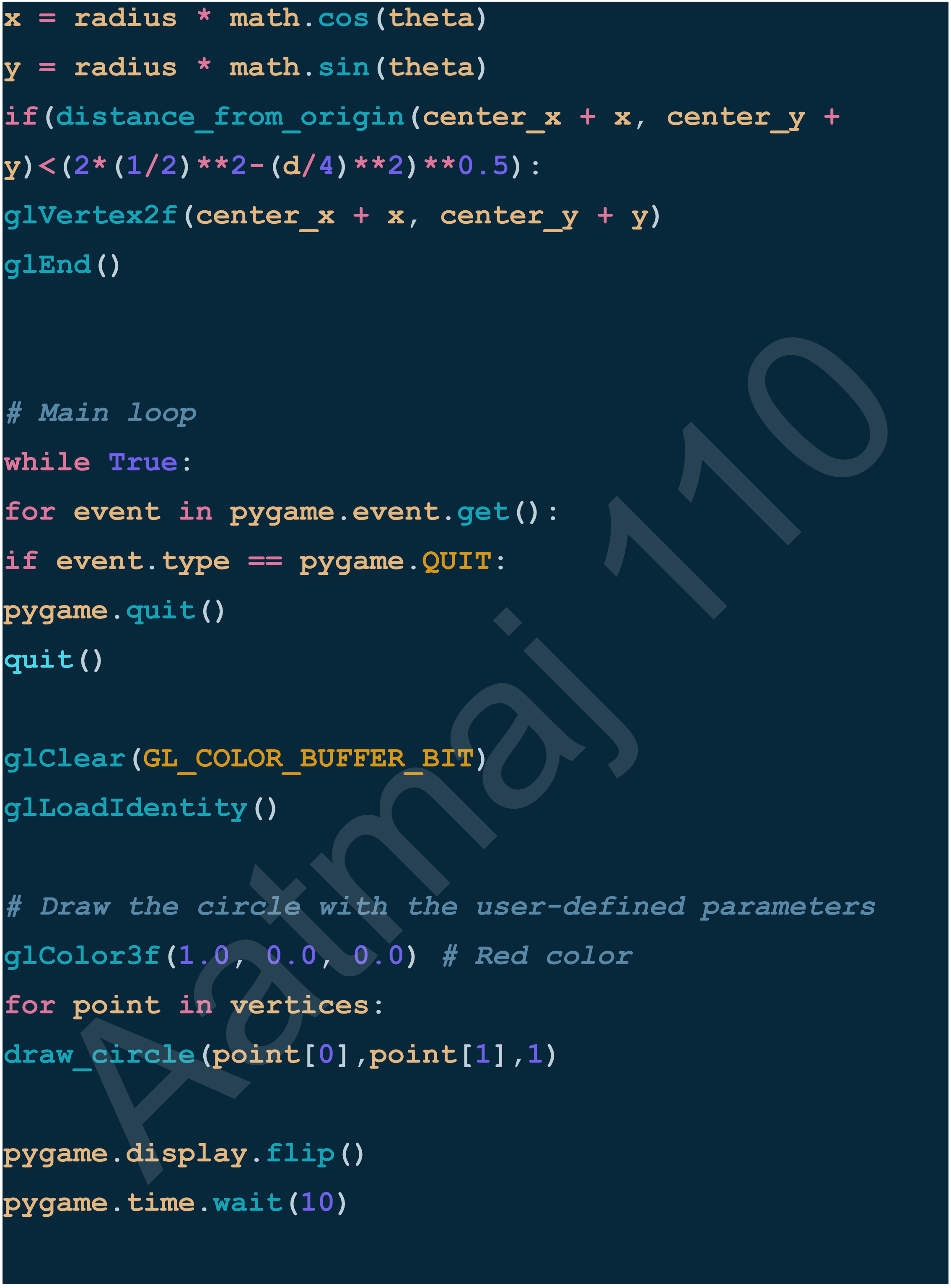
Post Lab

Draw The 5-, 11-, and 17-rosettes. using OpenGL.

```
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLUT import *
import math
# Initialize Pygame
pygame.init()
display = (800, 600)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)

grid = []
for x in range(-10,10):
x=x/10
for y in range(-10,10):
y=y/10
grid.append([x,y])
# vertices that are equidistant


def equilateral_polygon_vertices(num_sides):
import math

# Calculate the angle between each pair of consecutive
vertices
angle = 360 / num_sides

# Calculate the distance from the origin to each
vertex
radius = 1.0 # You can adjust this to change the size
of the polygon
```



```
# Calculate the coordinates of the vertices
vertices = []

for i in range(num_sides):
x = radius * math.cos(math.radians(i * angle))
y = radius * math.sin(math.radians(i * angle))
vertices.append((x, y))

return vertices
def dis(point1,point2):
return sum((point1[i]-point2[i])**2 for i in
[0,1])**0.5



# Example usage:
num_sides = 11
vertices = equilateral_polygon_vertices(num_sides)
d = dis(vertices[0],vertices[1]) # for calculation of
distance to cutoff

def distance_from_origin(x,y):
return (x**2+y**2)**0.5

# Function to draw a circle
def draw_circle(center_x, center_y, radius,
num_segments=1000):
for i in range(num_segments + 1):
glBegin(GL_POINTS)
theta = 2.0 * 3.1415926 * i / num_segments
```



```
x = radius * math.cos(theta)
y = radius * math.sin(theta)
if(distance_from_origin(center_x + x, center_y +
y)<(2*(1/2)**2-(d/4)**2)**0.5):
glVertex2f(center_x + x, center_y + y)
glEnd()


# Main loop
while True:
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()

glClear(GL_COLOR_BUFFER_BIT)
glLoadIdentity()

# Draw the circle with the user-defined parameters
glColor3f(1.0, 0.0, 0.0) # Red color
for point in vertices:
draw_circle(point[0],point[1],1)

pygame.display.flip()
pygame.time.wait(10)
```