UNIVERSITY OF RWANDA
COLLEGE OF SCIENCE AND TECHNOLOGY
soICT
COMPUTER SCIENCE
LEVEL 3
MODULE: Computer graphics

ASSIGNMENT 2

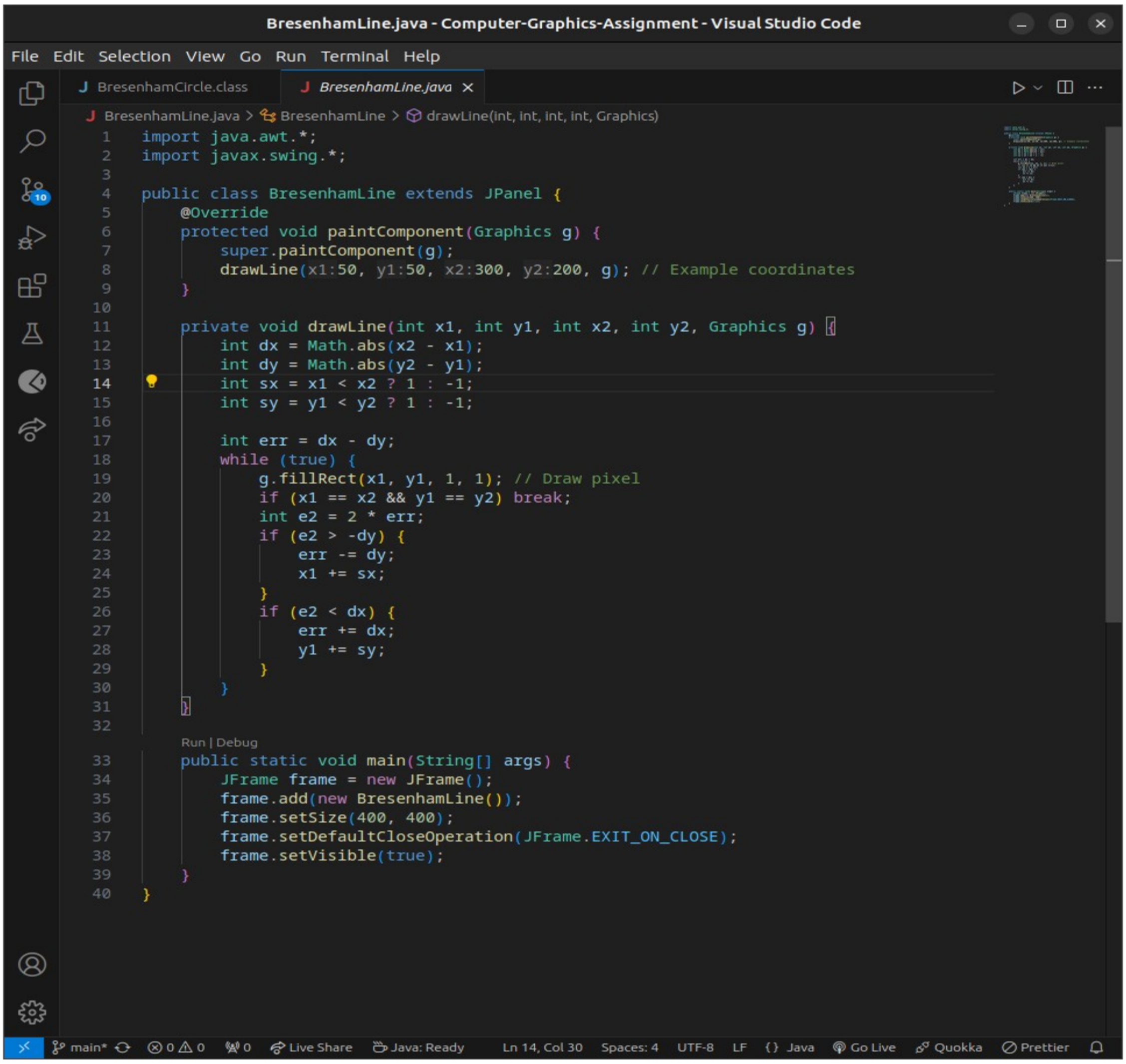
Group7:

USANASE Emeline	222008798
NISINGIZWE Marie Claire	222008836
IZERE BUGINGO Vainqueur	222019837
NSENGIYUMVA Clement	222005281
MUKUNZI Emmanuel	222019777

Note: All code snippets referenced in this document are available on the GitHub repository Computer graphics.

QUESTION 1:

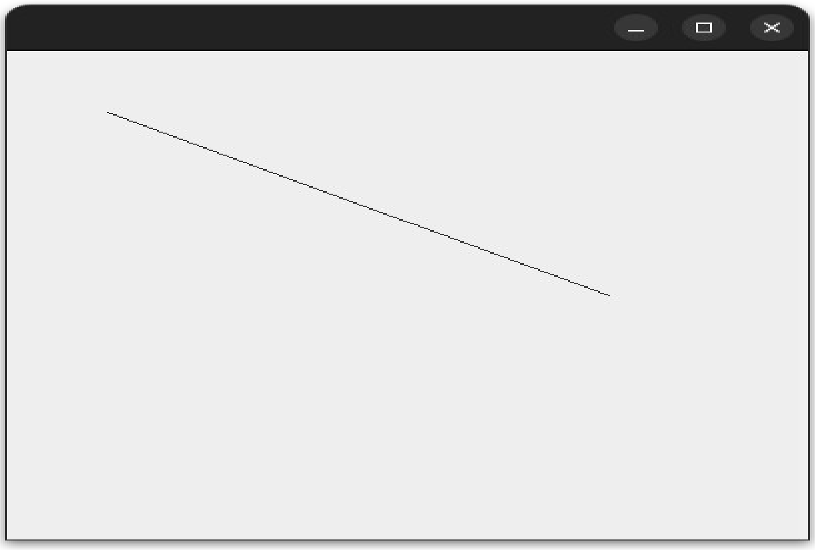
A. The following are java codes draw line in a swing using bresenham’s line algorithm.



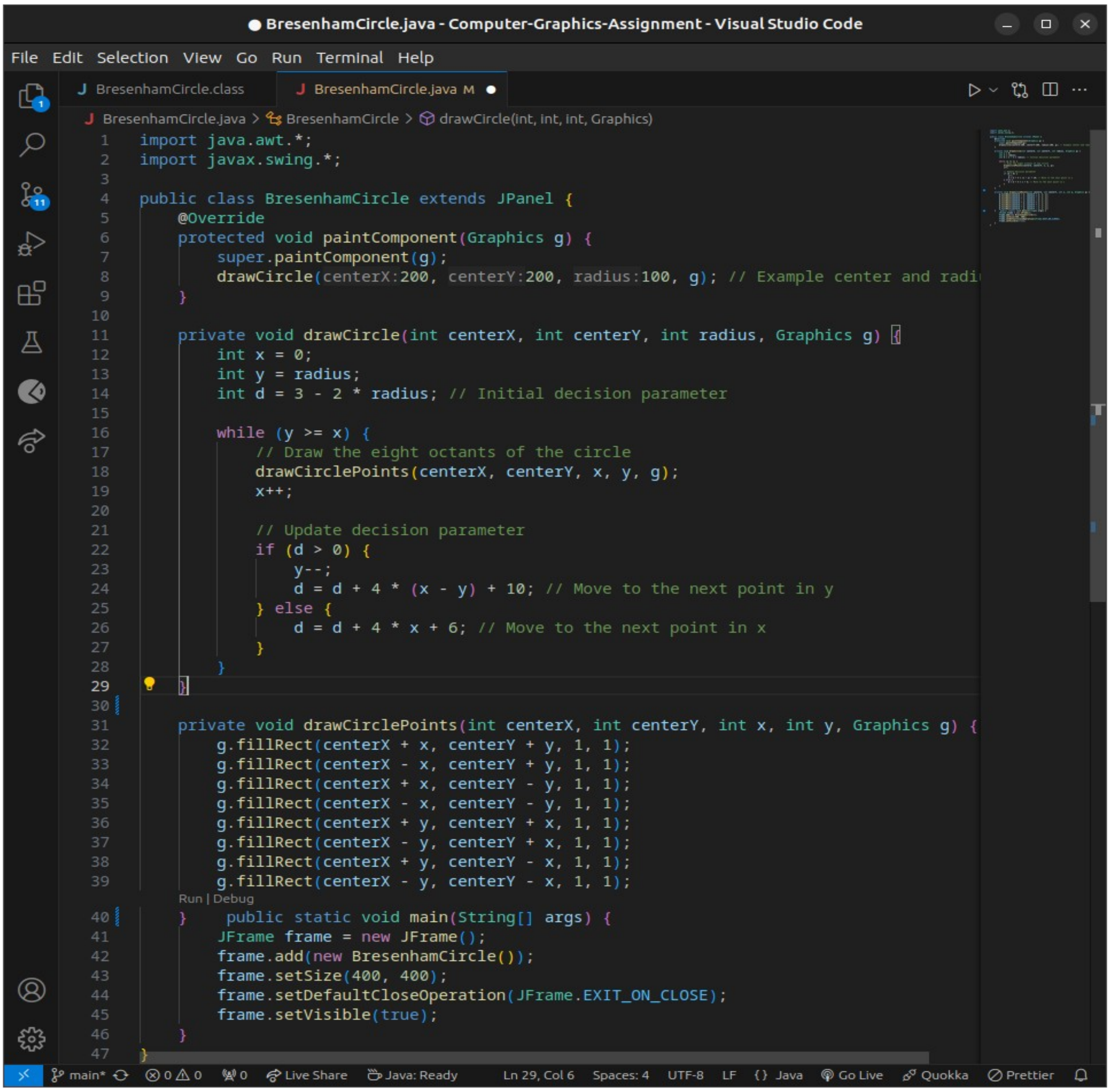
BresenhamLine.java - Computer-Graphics-Assignment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
BresenhamCircle.class
BresenhamLine.java
BresenhamLine.java > BresenhamLine > drawLine(int, int, int, int, Graphics)
Run | Debug
main* 0 0 0 Live Share Java: Ready Ln 14, Col 30 Spaces: 4 UTF-8 LF {} Java Go Live Quokka Prettier

```java
import java.awt.*;
import javax.swing.*;

public class BresenhamLine extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        drawLine(x1:50, y1:50, x2:300, y2:200, g); // Example coordinates
    }

    private void drawLine(int x1, int y1, int x2, int y2, Graphics g) {
        int dx = Math.abs(x2 - x1);
        int dy = Math.abs(y2 - y1);
        int sx = x1 < x2 ? 1 : -1;
        int sy = y1 < y2 ? 1 : -1;

        int err = dx - dy;
        while (true) {
            g.fillRect(x1, y1, 1, 1); // Draw pixel
            if (x1 == x2 && y1 == y2) break;
            int e2 = 2 * err;
            if (e2 > -dy) {
                err -= dy;
                x1 += sx;
            }
            if (e2 < dx) {
                err += dx;
                y1 += sy;
            }
        }
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.add(new BresenhamLine());
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

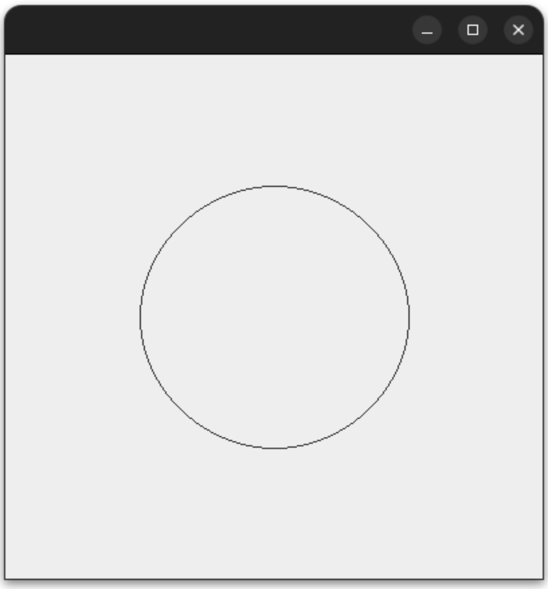
Here is the output



B. In figure bellow is the java codes that draw a circle following Berensham’s circle algorithm, alongside their output.

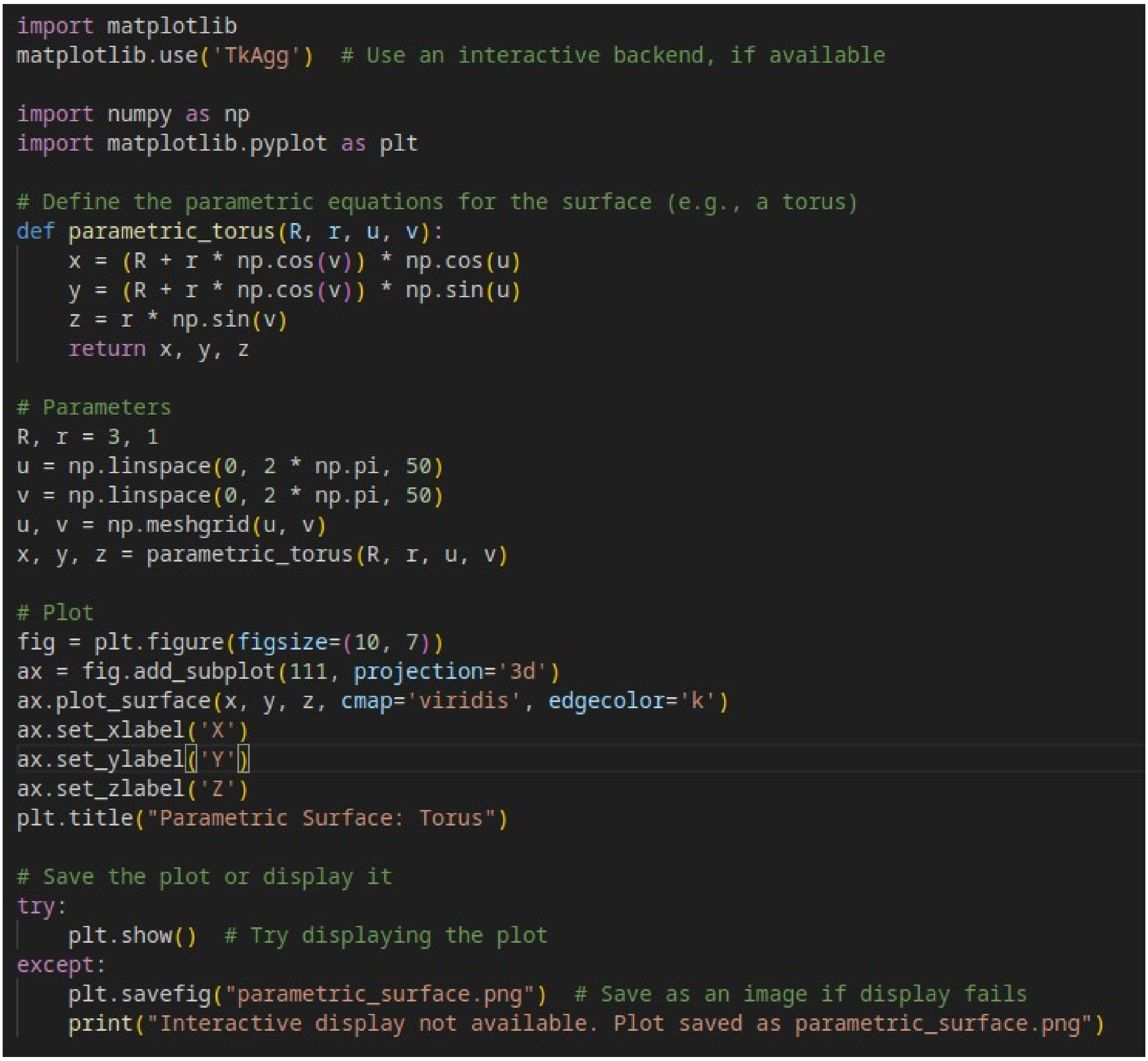

● BresenhamCircle.java - Computer-Graphics-Assignment - Visual Studio Code
File Edit Selection View Go Run Terminal Help
BresenhamCircle.class
BresenhamCircle.java M ●
BresenhamCircle.java > BresenhamCircle > drawCircle(int, int, int, Graphics)
Run | Debug
main* ⊗ 0 ⚠ 0 0 Live Share Java: Ready Ln 29, Col 6 Spaces: 4 UTF-8 LF {} Java Go Live Quokka Prettier

```java
import java.awt.*;
import javax.swing.*;

public class BresenhamCircle extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        drawCircle(centerX:200, centerY:200, radius:100, g); // Example center and radi
    }

    private void drawCircle(int centerX, int centerY, int radius, Graphics g) {
        int x = 0;
        int y = radius;
        int d = 3 - 2 * radius; // Initial decision parameter

        while (y >= x) {
            // Draw the eight octants of the circle
            drawCirclePoints(centerX, centerY, x, y, g);
            x++;

            // Update decision parameter
            if (d > 0) {
                y--;
                d = d + 4 * (x - y) + 10; // Move to the next point in y
            } else {
                d = d + 4 * x + 6; // Move to the next point in x
            }
        }
    }

    private void drawCirclePoints(int centerX, int centerY, int x, int y, Graphics g) {
        g.fillRect(centerX + x, centerY + y, 1, 1);
        g.fillRect(centerX - x, centerY + y, 1, 1);
        g.fillRect(centerX + x, centerY - y, 1, 1);
        g.fillRect(centerX - x, centerY - y, 1, 1);
        g.fillRect(centerX + y, centerY + x, 1, 1);
        g.fillRect(centerX - y, centerY + x, 1, 1);
        g.fillRect(centerX + y, centerY - x, 1, 1);
        g.fillRect(centerX - y, centerY - x, 1, 1);
    }    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.add(new BresenhamCircle());
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



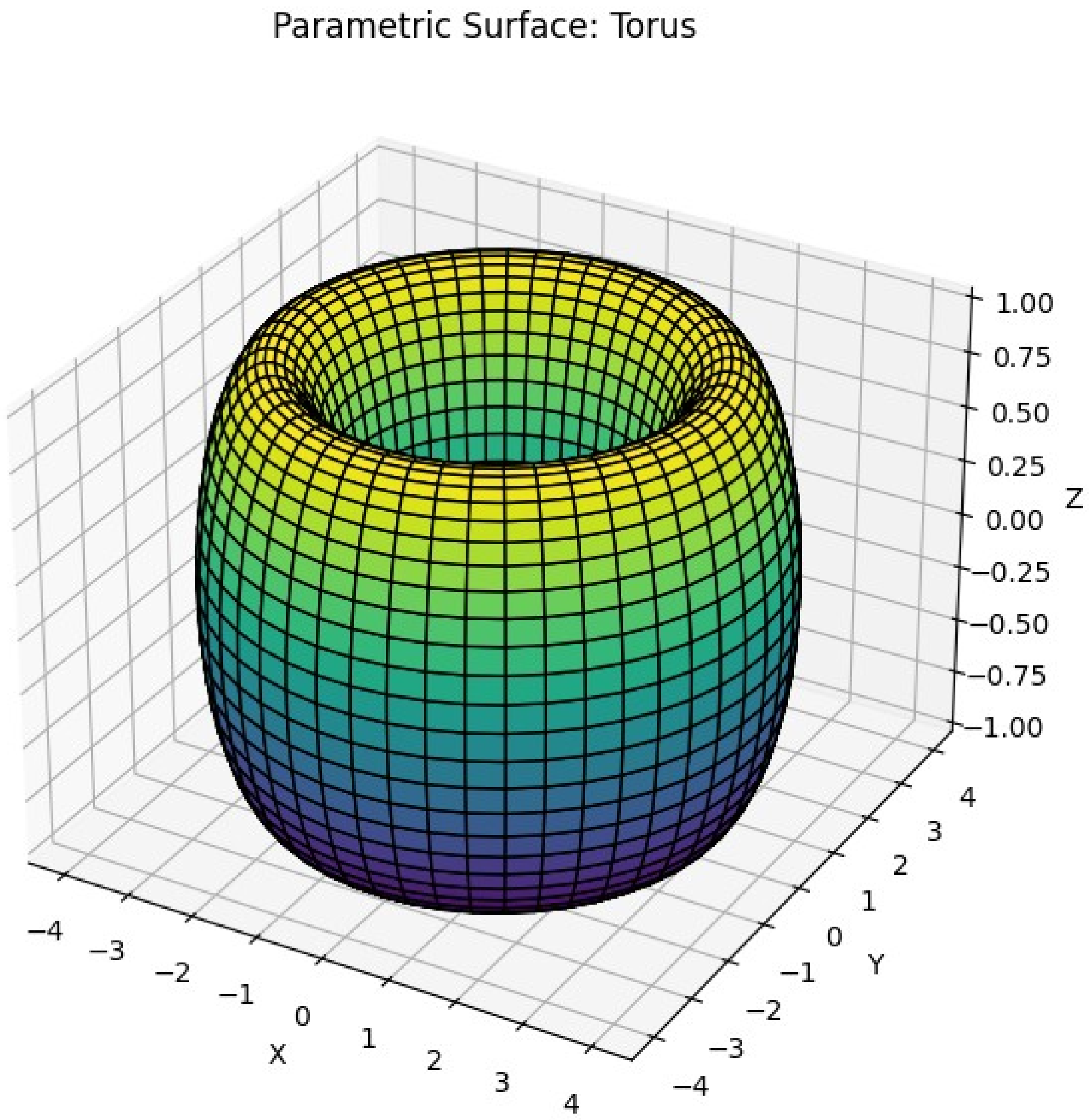
QUESTION 2:

Here is python codes that generate 3D plot of a torus surface defined by parametric equations, visualized with color map and labeled axes.



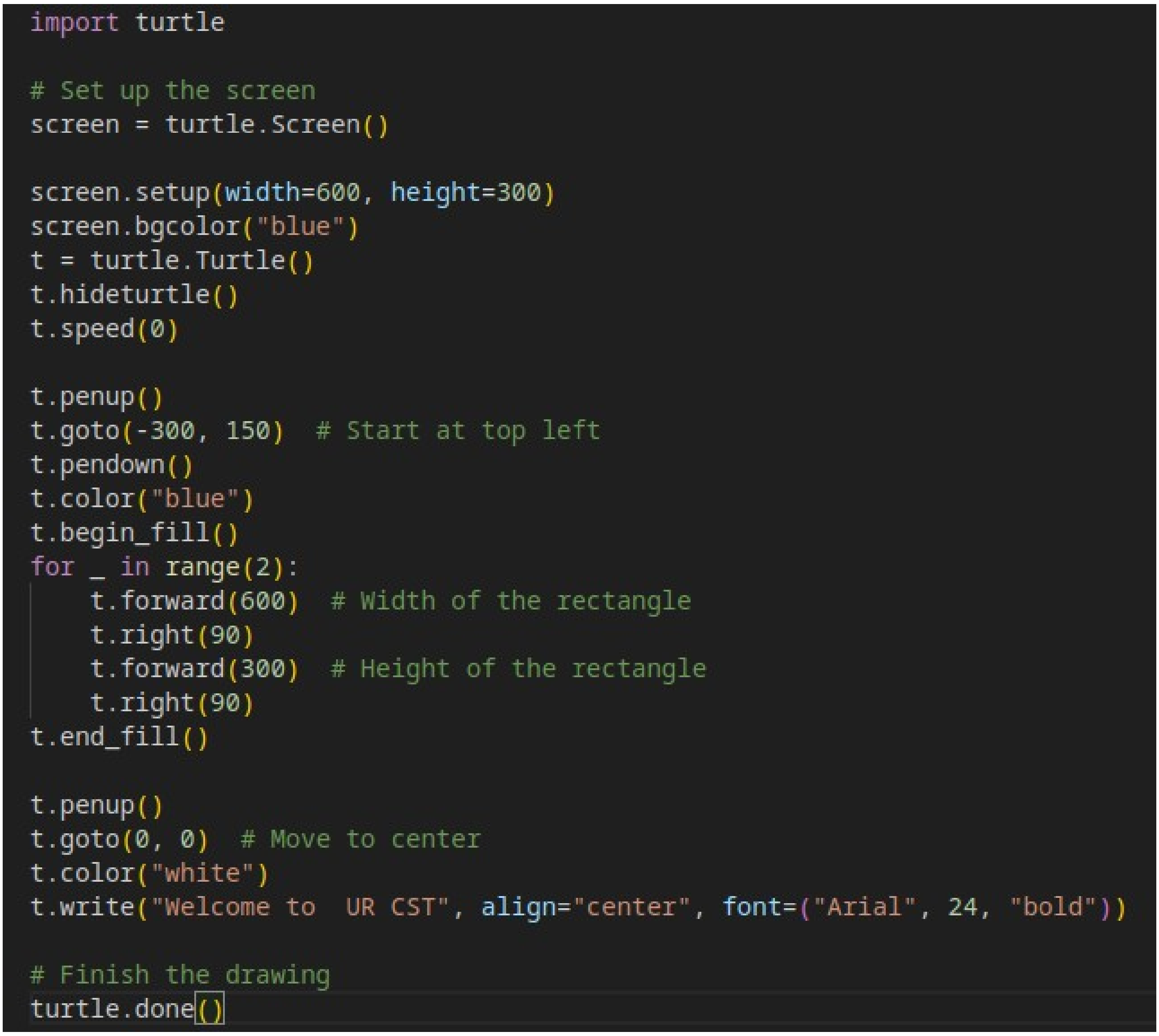
```
import matplotlib
matplotlib.use('TkAgg')  # Use an interactive backend, if available

import numpy as np
import matplotlib.pyplot as plt

# Define the parametric equations for the surface (e.g., a torus)
def parametric_torus(R, r, u, v):
    x = (R + r * np.cos(v)) * np.cos(u)
    y = (R + r * np.cos(v)) * np.sin(u)
    z = r * np.sin(v)
    return x, y, z

# Parameters
R, r = 3, 1
u = np.linspace(0, 2 * np.pi, 50)
v = np.linspace(0, 2 * np.pi, 50)
u, v = np.meshgrid(u, v)
x, y, z = parametric_torus(R, r, u, v)

# Plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z, cmap='viridis', edgecolor='k')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.title("Parametric Surface: Torus")

# Save the plot or display it
try:
    plt.show()  # Try displaying the plot
except:
    plt.savefig("parametric_surface.png")  # Save as an image if display fails
    print("Interactive display not available. Plot saved as parametric_surface.png")
```

And here is the output.



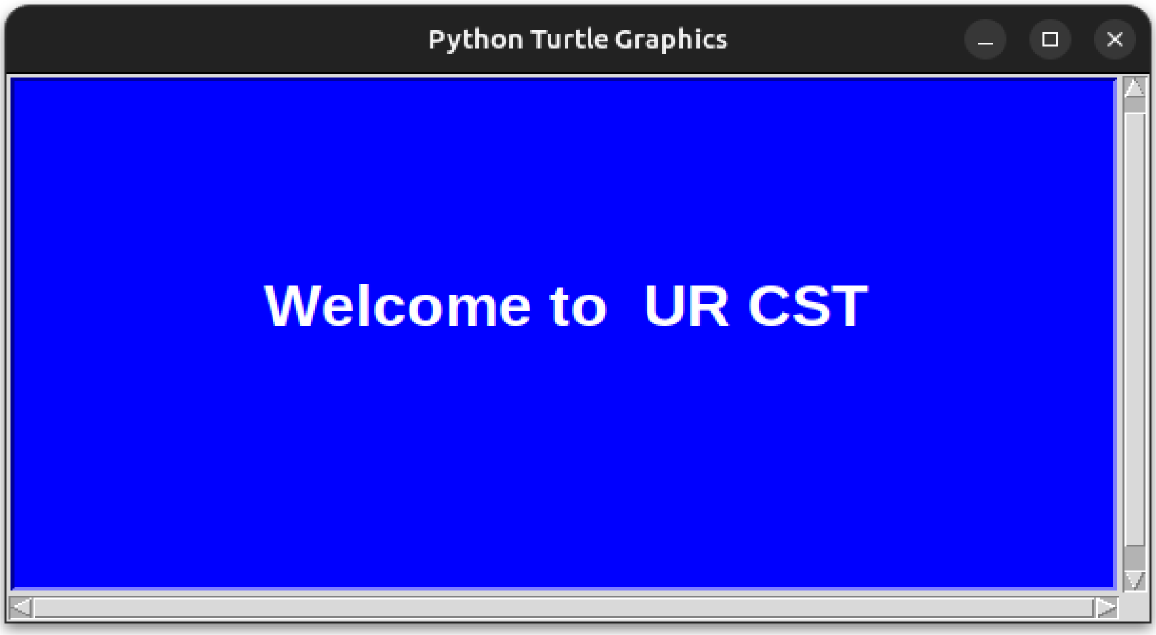
QUESTION 3:

On this question all codes are written in python



```python
import turtle

# Set up the screen
screen = turtle.Screen()

screen.setup(width=600, height=300)
screen.bgcolor("blue")
t = turtle.Turtle()
t.hideturtle()
t.speed(0)

t.penup()
t.goto(-300, 150)  # Start at top left
t.pendown()
t.color("blue")
t.begin_fill()
for _ in range(2):
    t.forward(600)  # Width of the rectangle
    t.right(90)
    t.forward(300)  # Height of the rectangle
    t.right(90)
t.end_fill()

t.penup()
t.goto(0, 0)  # Move to center
t.color("white")
t.write("Welcome to  UR CST", align="center", font=("Arial", 24, "bold"))

# Finish the drawing
turtle.done()
```

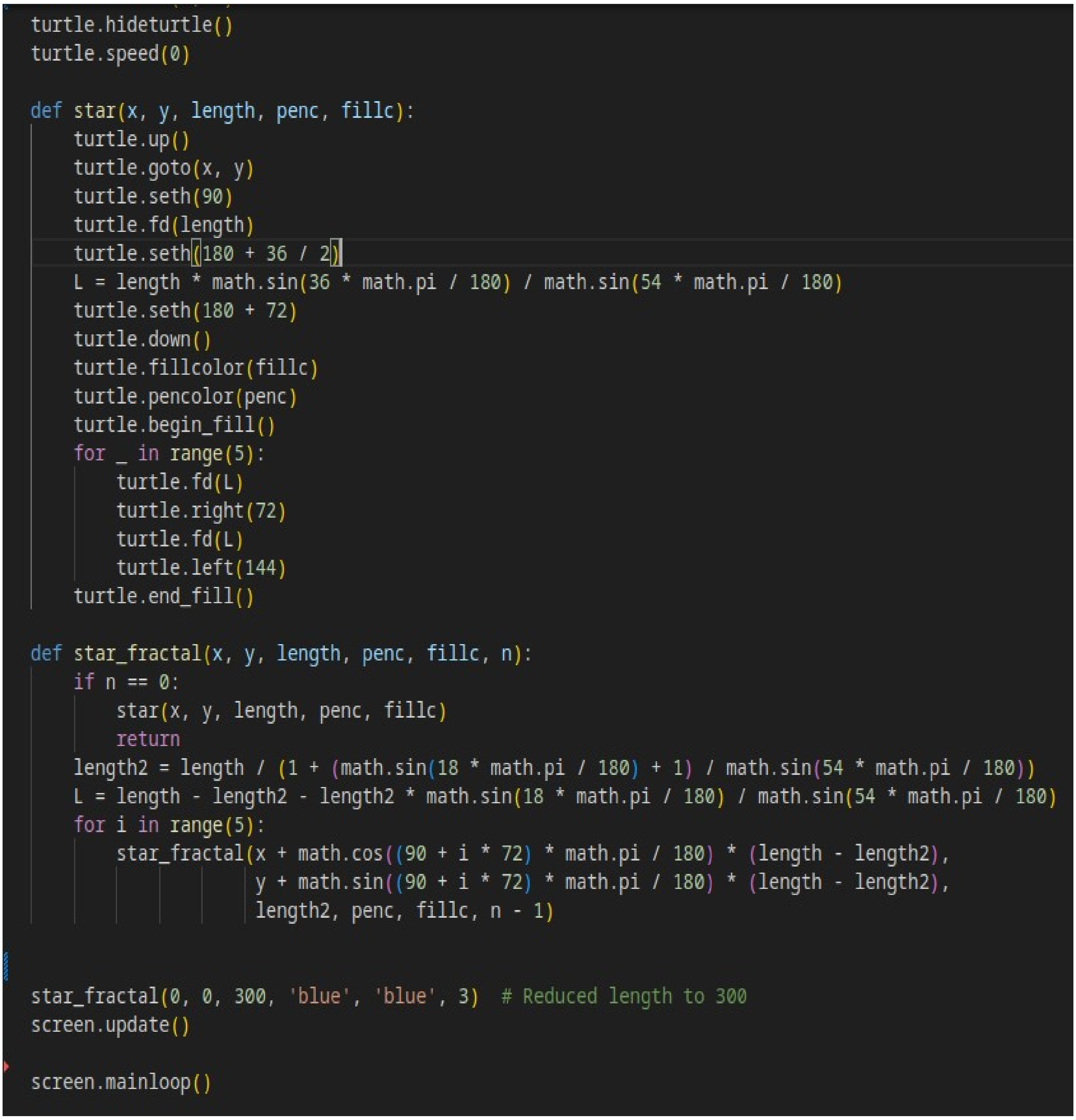
Output



```
Welcome to  UR CST
```

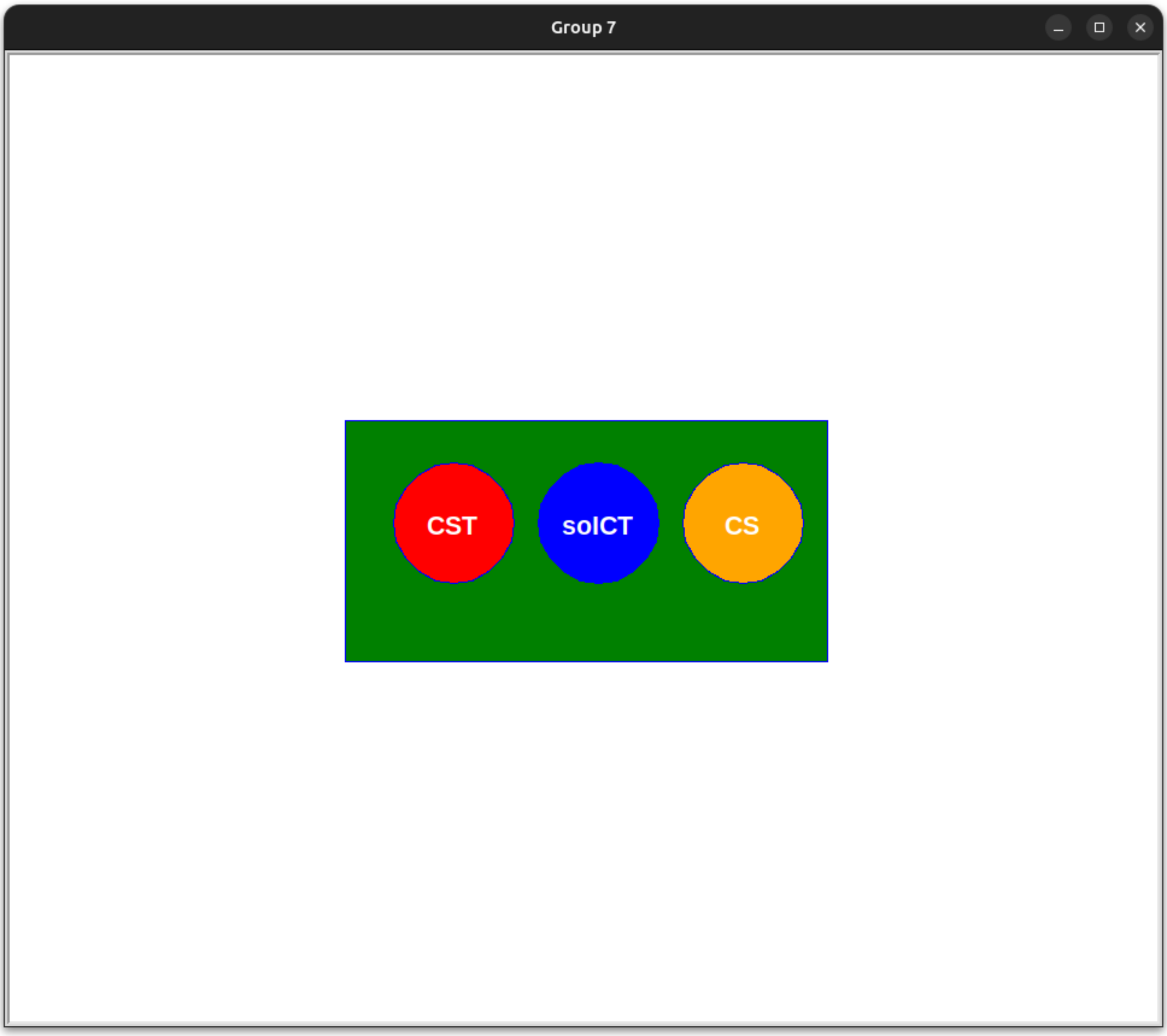
B.

Codes:



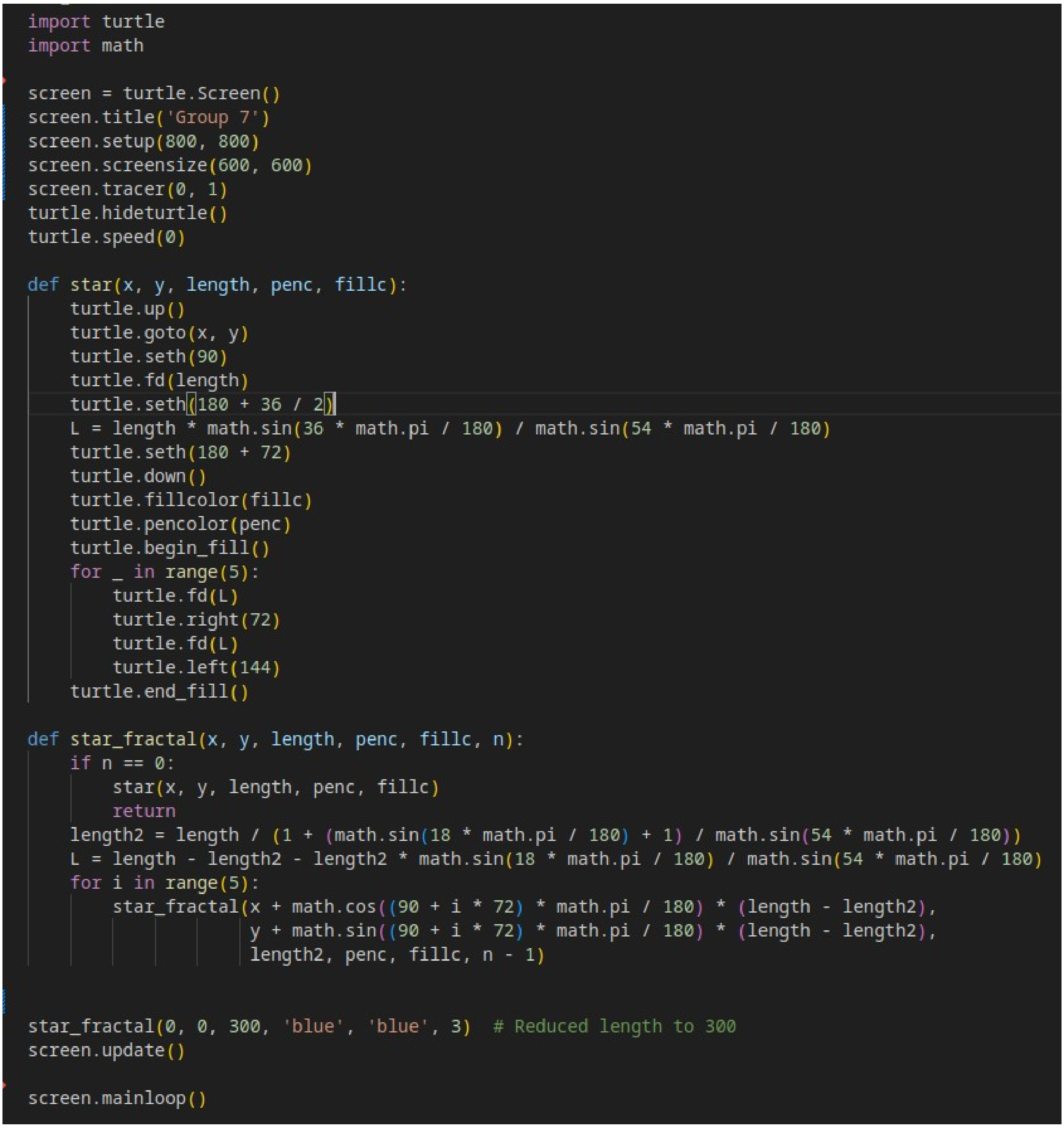
```python
turtle.hideturtle()
turtle.speed(0)

def star(x, y, length, penc, fillc):
    turtle.up()
    turtle.goto(x, y)
    turtle.seth(90)
    turtle.fd(length)
    turtle.seth(180 + 36 / 2)
    L = length * math.sin(36 * math.pi / 180) / math.sin(54 * math.pi / 180)
    turtle.seth(180 + 72)
    turtle.down()
    turtle.fillcolor(fillc)
    turtle.pencolor(penc)
    turtle.begin_fill()
    for _ in range(5):
        turtle.fd(L)
        turtle.right(72)
        turtle.fd(L)
        turtle.left(144)
    turtle.end_fill()

def star_fractal(x, y, length, penc, fillc, n):
    if n == 0:
        star(x, y, length, penc, fillc)
        return
    length2 = length / (1 + (math.sin(18 * math.pi / 180) + 1) / math.sin(54 * math.pi / 180))
    L = length - length2 - length2 * math.sin(18 * math.pi / 180) / math.sin(54 * math.pi / 180)
    for i in range(5):
        star_fractal(x + math.cos((90 + i * 72) * math.pi / 180) * (length - length2),
                     y + math.sin((90 + i * 72) * math.pi / 180) * (length - length2),
                     length2, penc, fillc, n - 1)


star_fractal(0, 0, 300, 'blue', 'blue', 3)  # Reduced length to 300
screen.update()

screen.mainloop()
```

Output:



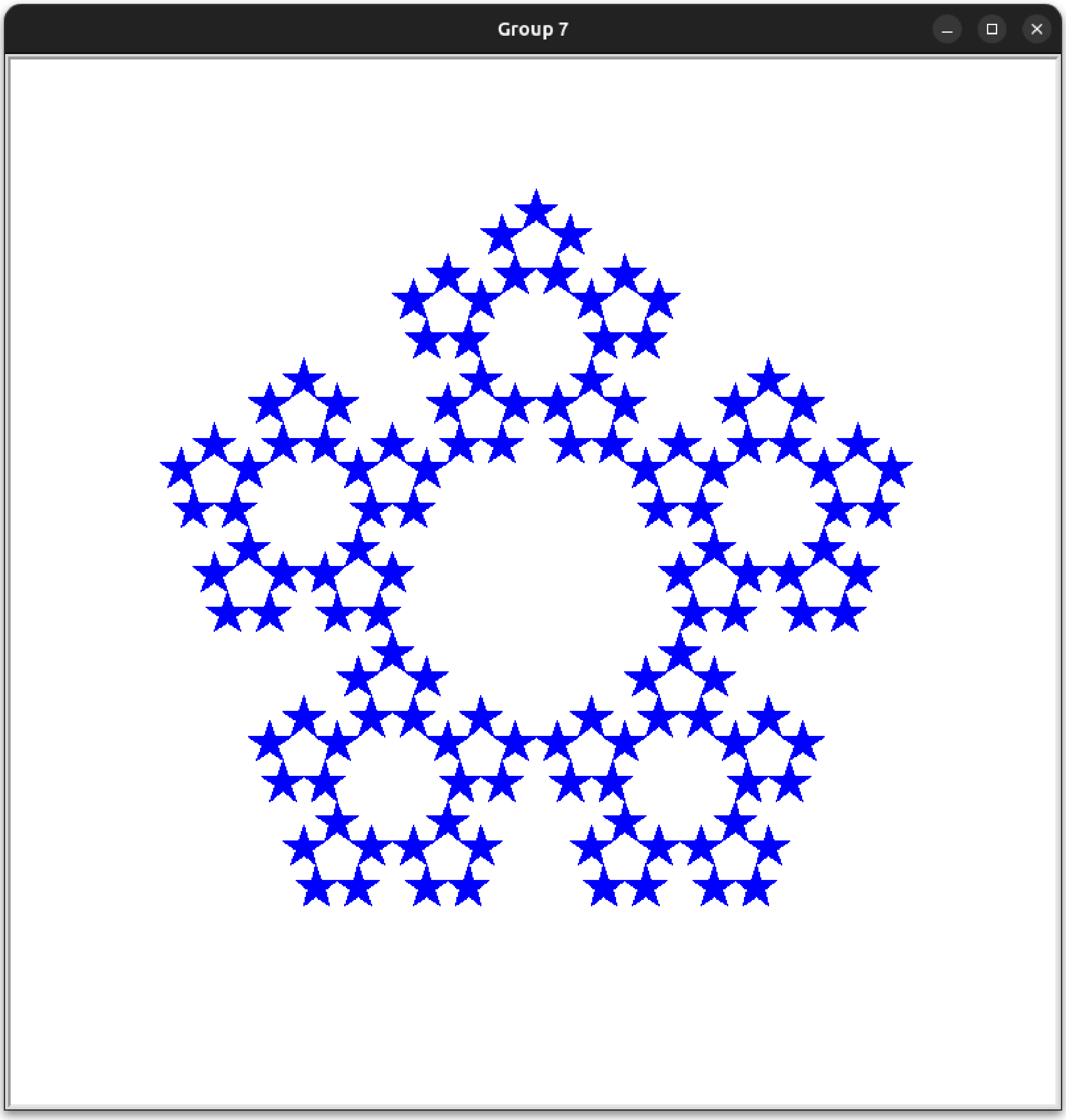
C.

Codes



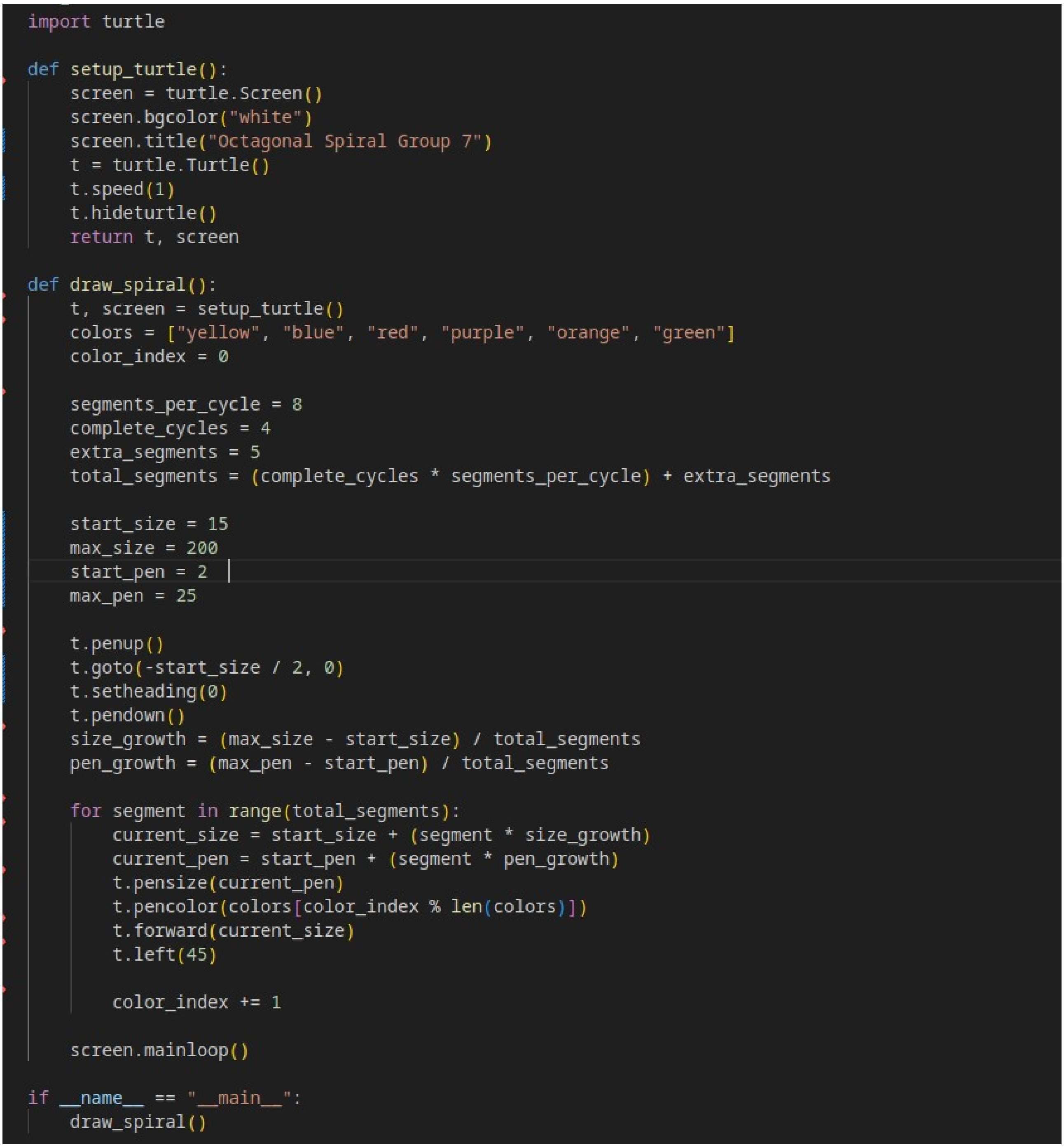
```python
import turtle
import math

screen = turtle.Screen()
screen.title('Group 7')
screen.setup(800, 800)
screen.screensize(600, 600)
screen.tracer(0, 1)
turtle.hideturtle()
turtle.speed(0)

def star(x, y, length, penc, fillc):
    turtle.up()
    turtle.goto(x, y)
    turtle.seth(90)
    turtle.fd(length)
    turtle.seth(180 + 36 / 2)
    L = length * math.sin(36 * math.pi / 180) / math.sin(54 * math.pi / 180)
    turtle.seth(180 + 72)
    turtle.down()
    turtle.fillcolor(fillc)
    turtle.pencolor(penc)
    turtle.begin_fill()
    for _ in range(5):
        turtle.fd(L)
        turtle.right(72)
        turtle.fd(L)
        turtle.left(144)
    turtle.end_fill()

def star_fractal(x, y, length, penc, fillc, n):
    if n == 0:
        star(x, y, length, penc, fillc)
        return
    length2 = length / (1 + (math.sin(18 * math.pi / 180) + 1) / math.sin(54 * math.pi / 180))
    L = length - length2 - length2 * math.sin(18 * math.pi / 180) / math.sin(54 * math.pi / 180)
    for i in range(5):
        star_fractal(x + math.cos((90 + i * 72) * math.pi / 180) * (length - length2),
                     y + math.sin((90 + i * 72) * math.pi / 180) * (length - length2),
                     length2, penc, fillc, n - 1)


star_fractal(0, 0, 300, 'blue', 'blue', 3)  # Reduced length to 300
screen.update()

screen.mainloop()
```

Output:



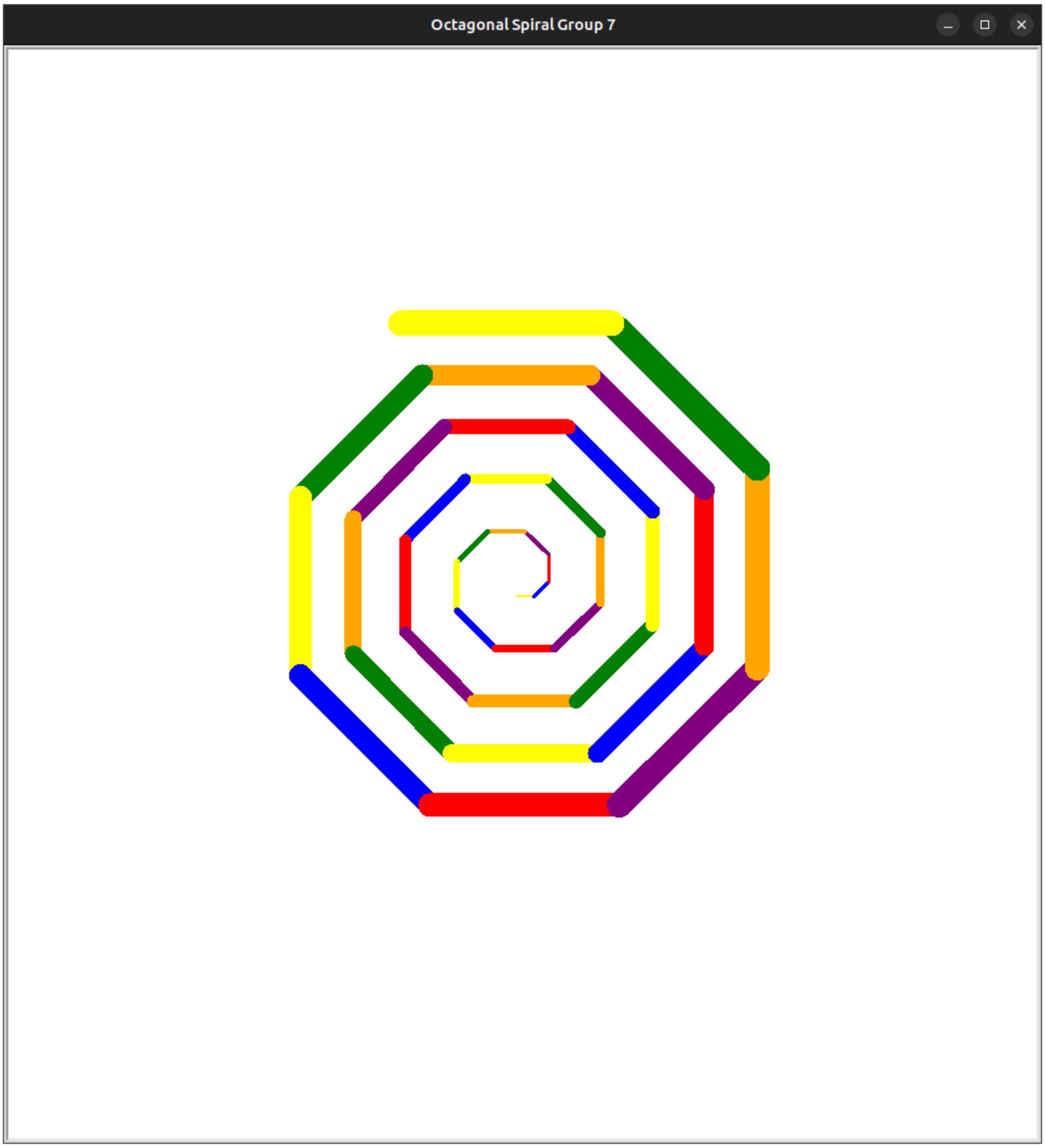
D.

Codes:



```python
import turtle

def setup_turtle():
    screen = turtle.Screen()
    screen.bgcolor("white")
    screen.title("Octagonal Spiral Group 7")
    t = turtle.Turtle()
    t.speed(1)
    t.hideturtle()
    return t, screen

def draw_spiral():
    t, screen = setup_turtle()
    colors = ["yellow", "blue", "red", "purple", "orange", "green"]
    color_index = 0

    segments_per_cycle = 8
    complete_cycles = 4
    extra_segments = 5
    total_segments = (complete_cycles * segments_per_cycle) + extra_segments

    start_size = 15
    max_size = 200
    start_pen = 2
    max_pen = 25

    t.penup()
    t.goto(-start_size / 2, 0)
    t.setheading(0)
    t.pendown()
    size_growth = (max_size - start_size) / total_segments
    pen_growth = (max_pen - start_pen) / total_segments

    for segment in range(total_segments):
        current_size = start_size + (segment * size_growth)
        current_pen = start_pen + (segment * pen_growth)
        t.pensize(current_pen)
        t.pencolor(colors[color_index % len(colors)])
        t.forward(current_size)
        t.left(45)

        color_index += 1

    screen.mainloop()

if __name__ == "__main__":
    draw_spiral()
```

Output:



Octagonal Spiral Group 7