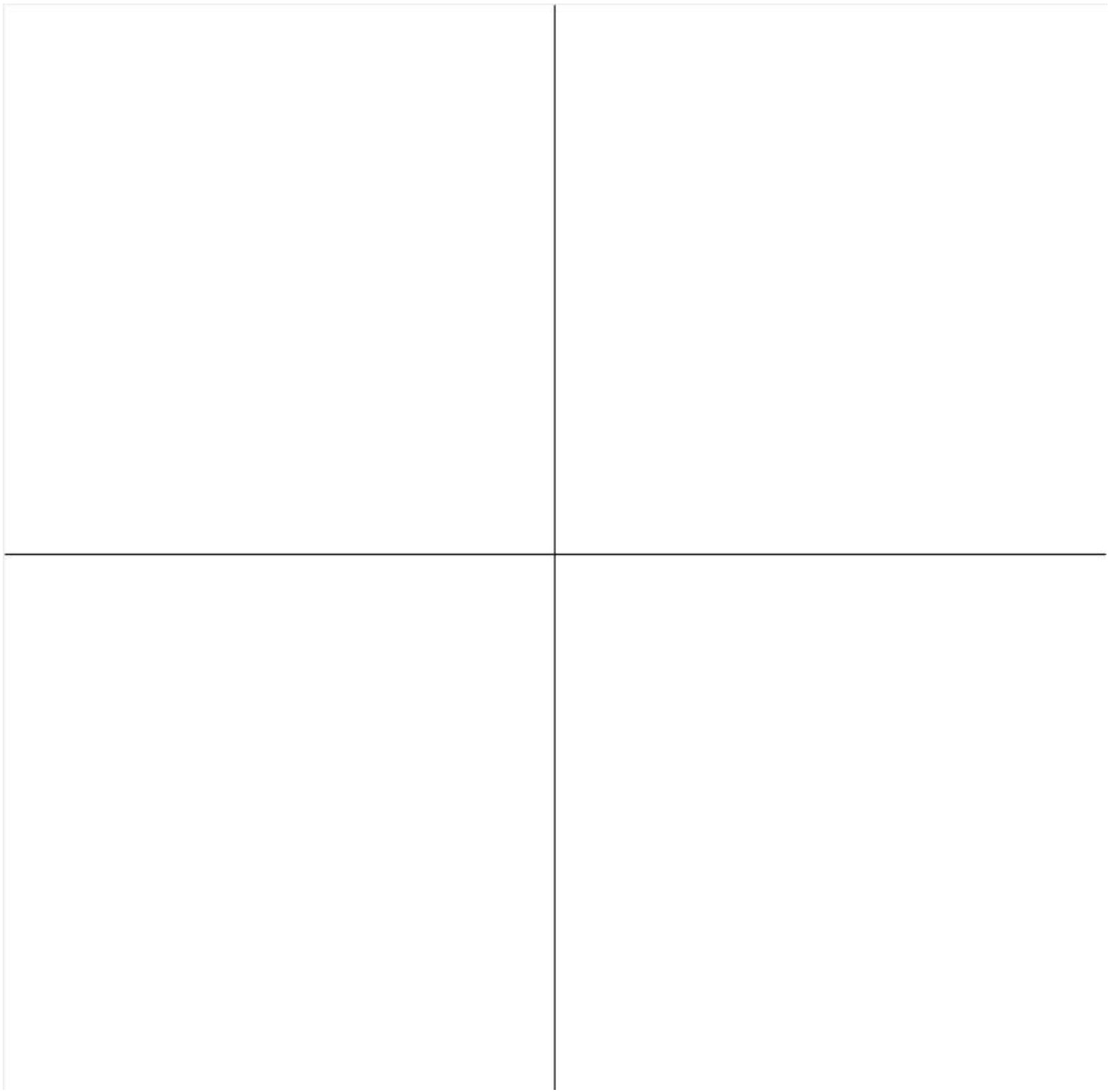
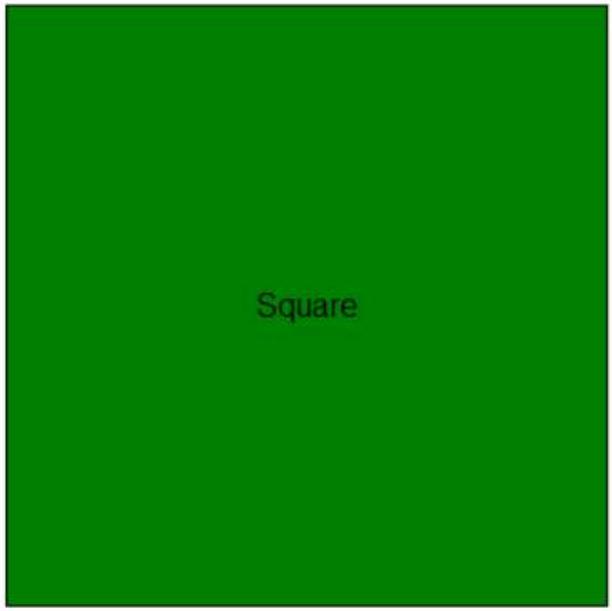
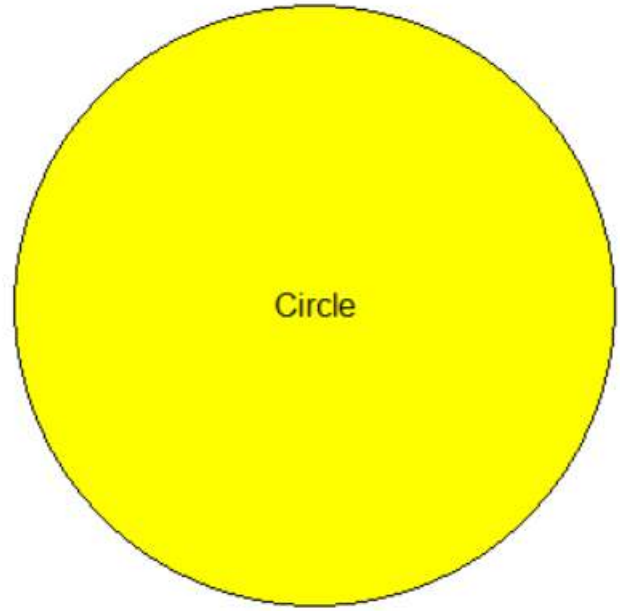
```
#PRATICAL 1A: Theory question not given till now. Leave for now.
```

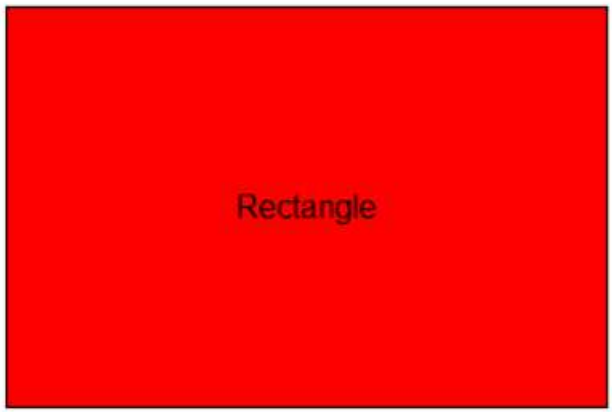
```python
#Practical 1B: WAP to draw Co-ordinate axes.

import tkinter
root = tkinter.Tk()
root.geometry("700x700")
C = tkinter.Canvas(root, height=800, width=800, bg="white")
C.create_line(0,400,800,400, fill="black")
C.create_line(400,0,400,800, fill="black")
C.pack()
root.mainloop()
```

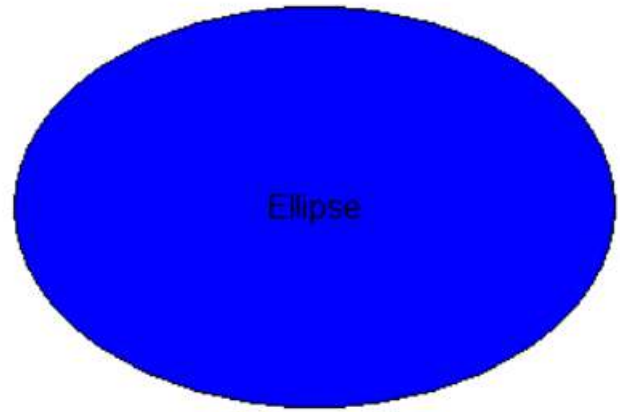


```python
#Practical 2A: Divide Screen into 4 regions, draw circle, rectangle, square, and ellipse in each region with appropriate title.

import tkinter
root = tkinter.Tk()
root.geometry("700x700")
C = tkinter.Canvas(root, height=800, width=800, bg="white")
C.create_line(0,400,800,400, fill="black")
C.create_line(400,0,400,800, fill="black")
C.create_rectangle(50,50,350,350,fill="green")
C.create_rectangle(50,500,350,700,fill="red")
C.create_oval(450,50,750,350,fill="yellow")
C.create_oval(450,500,750,700,fill="blue")
C.create_text(200,200,text="Square",font=(15))
C.create_text(600,200,text="Circle",font=(15))
C.create_text(200,600,text="Rectangle",font=(15))
C.create_text(600,600,text="Ellipse",font=(15))
C.pack()
root.mainloop()
```

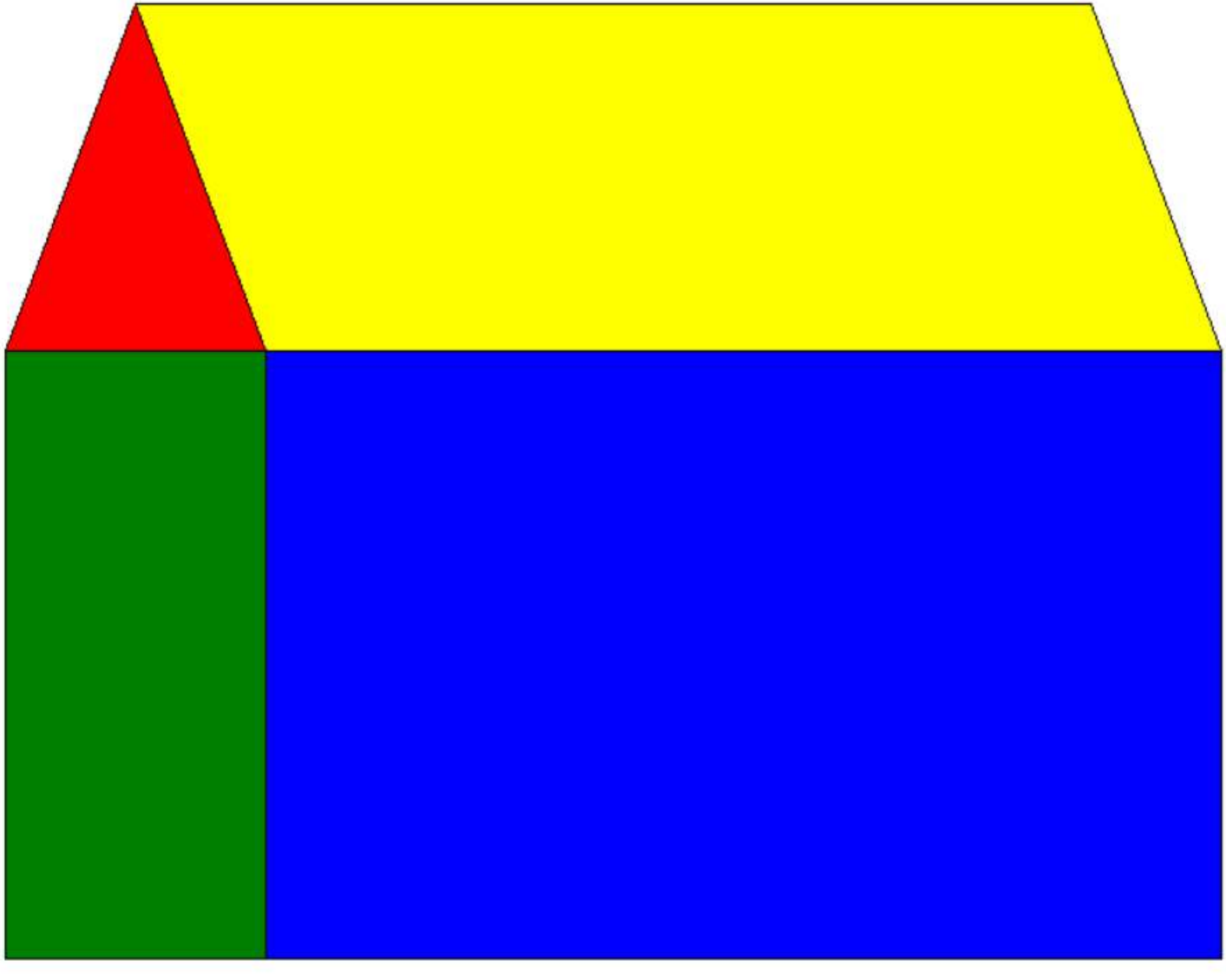

Square


Circle

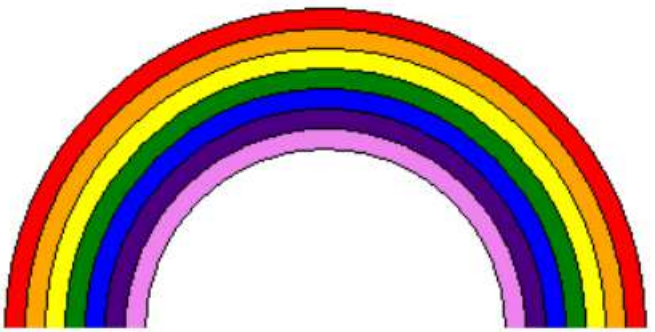

Rectangle


Ellipse

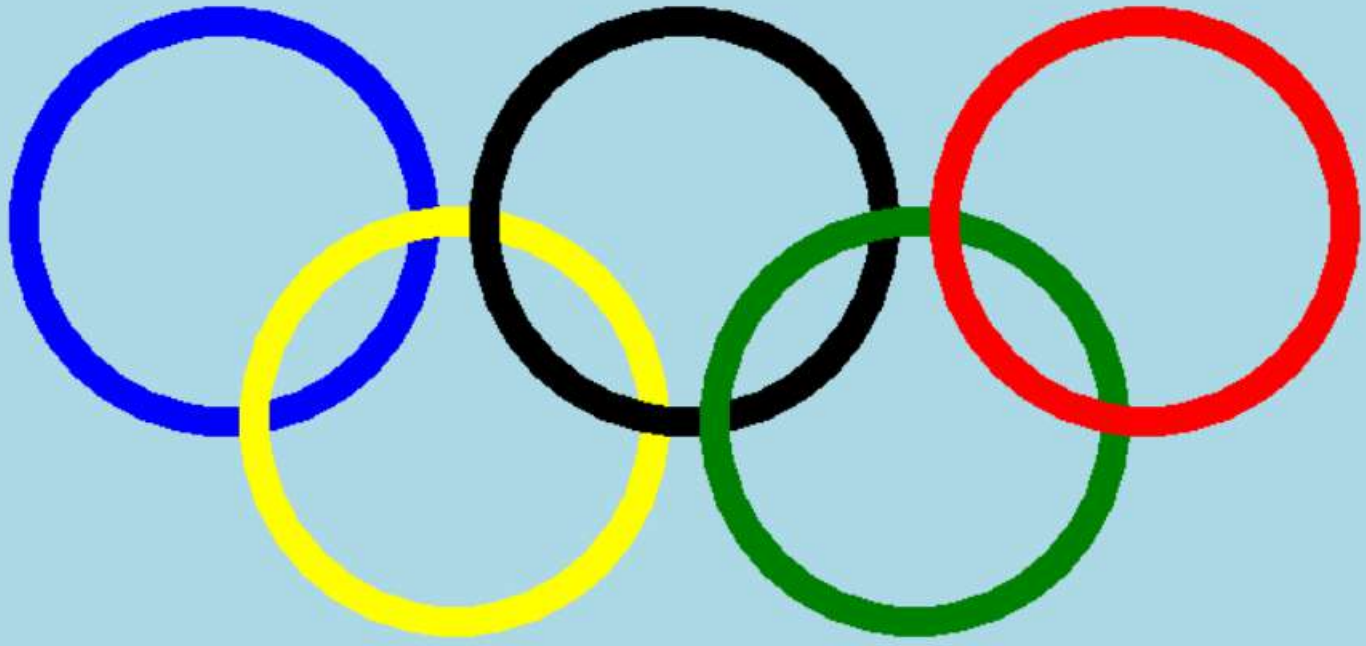
```python
#Practical 2B: WAP to Draw A Simple Hut

import tkinter
root = tkinter.Tk()
root.geometry("700x700")
C = tkinter.Canvas(root, height=800, width=800, bg="white")
C.create_polygon(50,400,125,200,200,400,fill="red",outline="black")
C.create_polygon(125,200,675,200,750,400,200,400,fill="yellow",outline="black")
C.create_rectangle(50,400,200,750,fill="green",outline="black")
C.create_rectangle(200,400,750,750,fill="blue",outline="black")
C.pack()
root.mainloop()
```



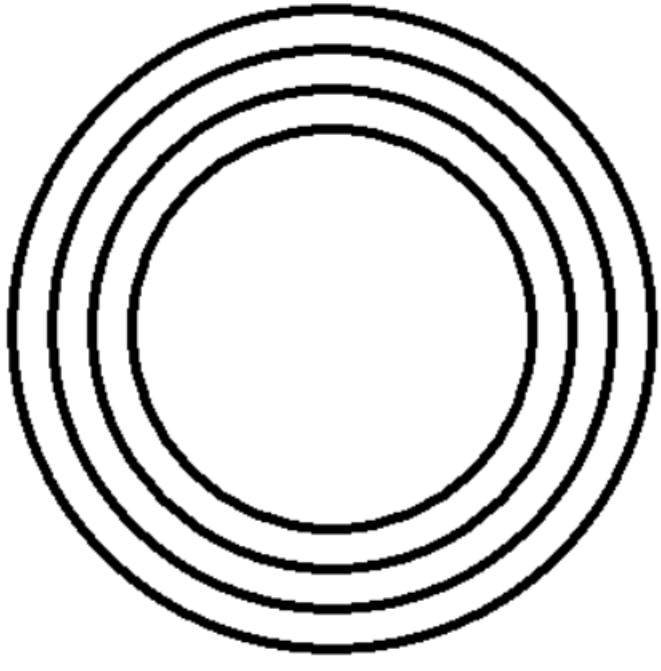
```python
#Practical 3a: WAP to Draw a Rainbow

import tkinter
root = tkinter.Tk()
root.geometry("1000x1000")
C = tkinter.Canvas(root,bg="white",height=800, width=800)
C.create_oval(240,240,560,560,fill="red")
C.create_oval(250,250,550,550,fill="orange")
C.create_oval(260,260,540,540,fill="yellow")
C.create_oval(270,270,530,530,fill="green")
C.create_oval(280,280,520,520,fill="blue")
C.create_oval(290,290,510,510,fill="indigo")
C.create_oval(300,300,500,500,fill="violet")
C.create_oval(310,310,490,490,fill="white")
C.create_rectangle(0,400,800,800,outline="white",fill="white")
C.pack()
root.mainloop()
```



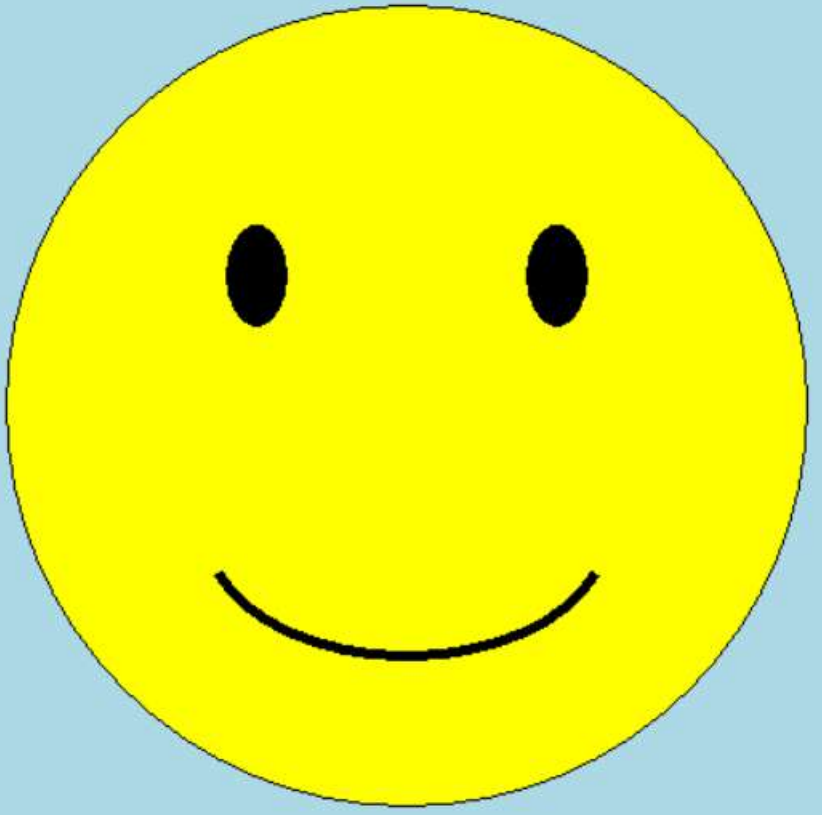
```python
#Practical 3B: WAP to draw Olympic Ring

import tkinter
root = tkinter.Tk()
root.geometry("700x700")
C = tkinter.Canvas(root, height=800, width=845, bg="lightblue")
C.create_oval(100,200,300,400,width="15",outline="blue")
C.create_oval(215,300,415,500,width="15",outline="yellow")
C.create_oval(330,200,530,400,width="15",outline="black")
C.create_oval(445,300,645,500,width="15",outline="green")
C.create_oval(560,200,760,400,width="15",outline="red")
C.pack()
root.mainloop()
```



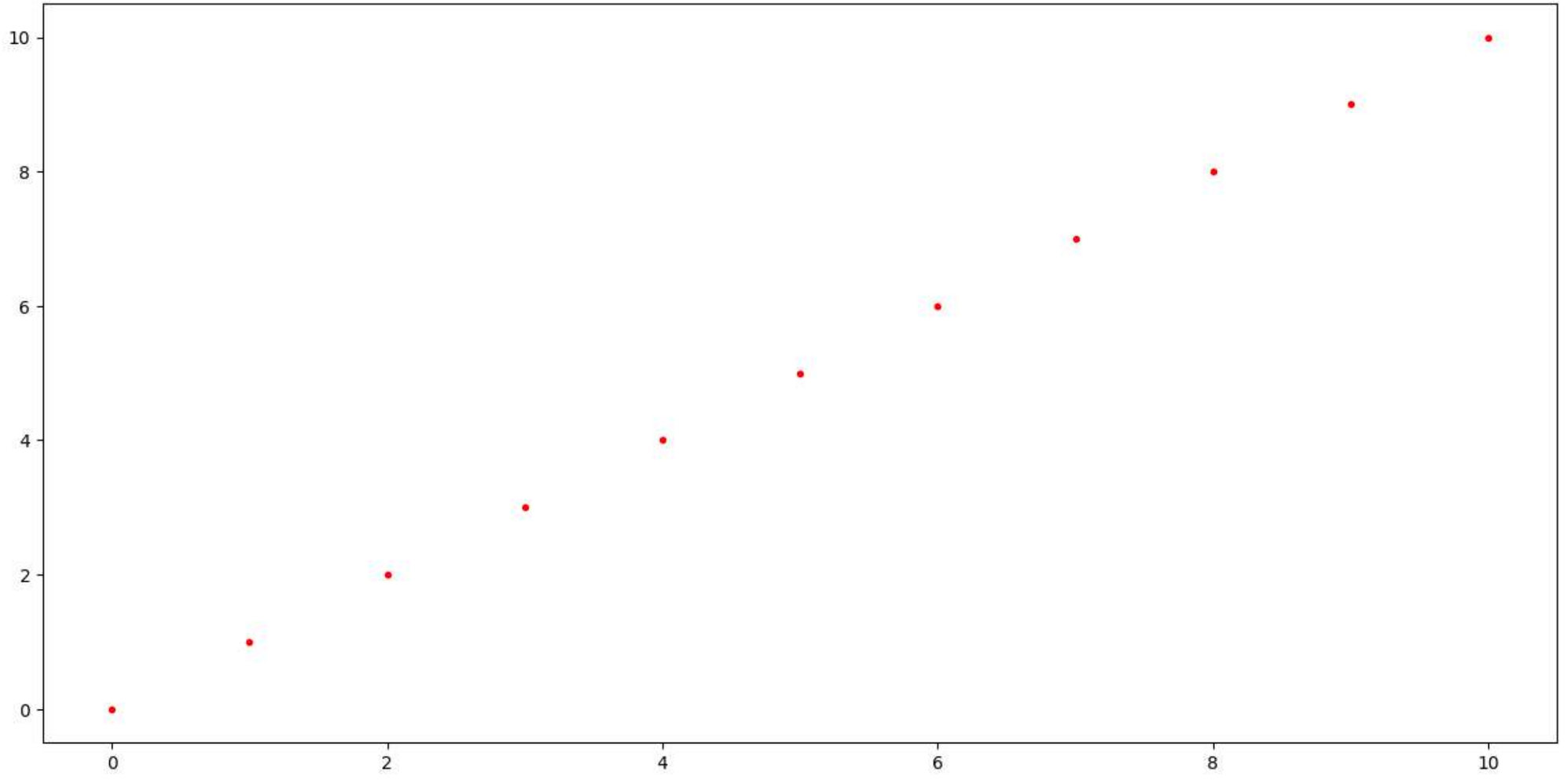
```python
#Practical 3C: WAP to Draw a cocentric circle

import tkinter
root = tkinter.Tk()
root.geometry("1000x1000")
C = tkinter.Canvas(root,bg="white",height=800, width=800)
C.create_oval(240,240,560,560,width="5")
C.create_oval(260,260,540,540,width="5")
C.create_oval(280,280,520,520,width="5")
C.create_oval(300,300,500,500,width="5")
C.pack()
root.mainloop()
```



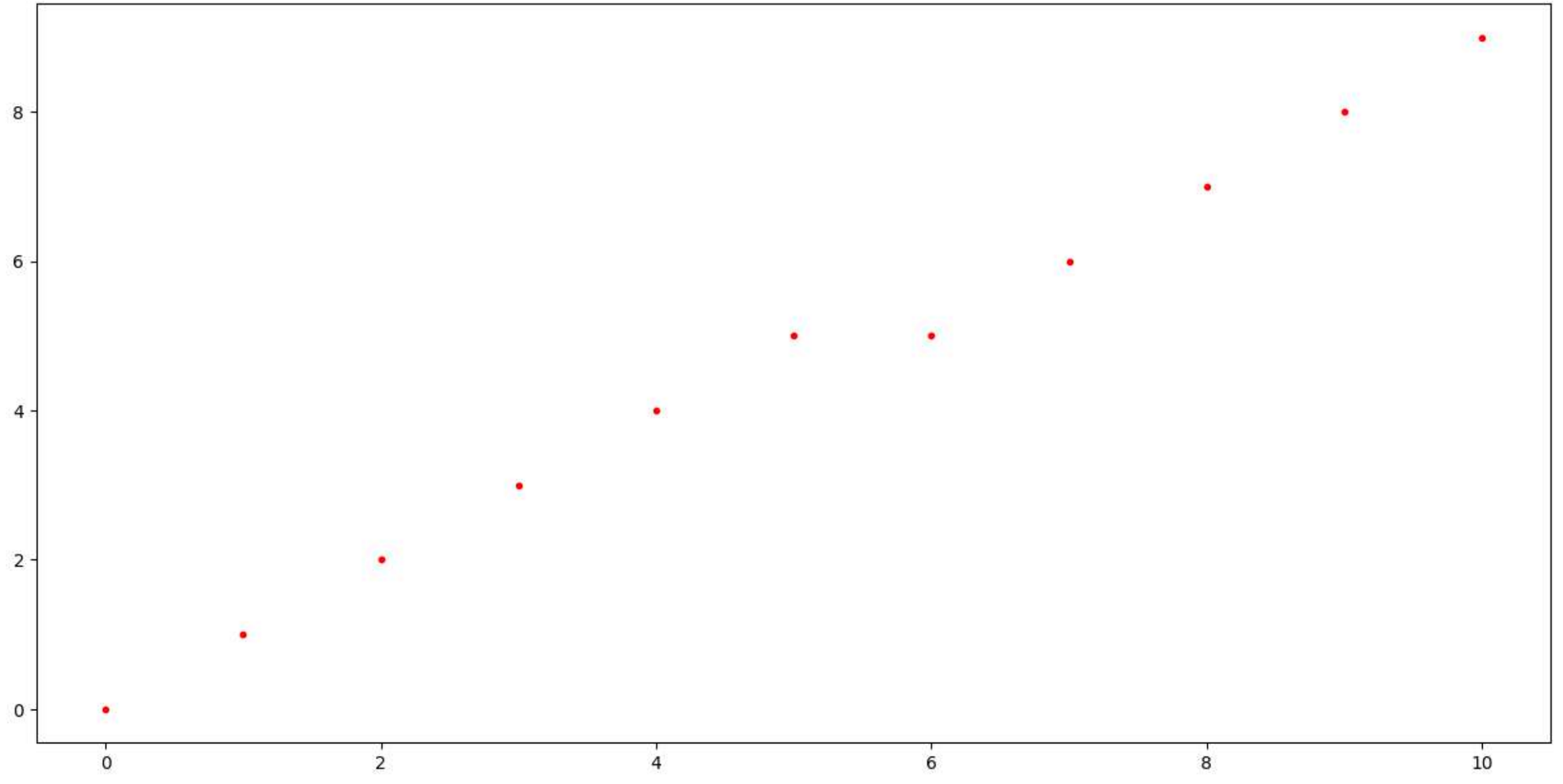
```python
#Practical 3D: Write a Program to draw Smiley

import tkinter
root = tkinter.Tk()
root.geometry("1000x1000")
C = tkinter.Canvas(root, height=800, width=845, bg="lightblue")
C.create_oval(200,200,600,600,fill="yellow")
C.create_oval(310,310,340,360,fill="black")
C.create_oval(460,310,490,360,fill="black")
C.create_arc(300,400,500,525,start=-20,extent=-140,style="arc",width="5")
C.pack()
root.mainloop()
```



```python
# Pratical 4A: WAP to implement DDA Line Drawing Algorithm

import matplotlib.pyplot as nigger

x0 = int(input("Enter x0: "))
y0 = int(input("Enter y0: "))
xn = int(input("Enter xn: "))
yn = int(input("Enter yn: "))

dx = xn - x0
dy = yn - y0

if abs(dx) > abs(dy):
    steps = abs(dx)
else:
    steps = abs(dy)

xincrement = dx // steps
yincrement = dy // steps

i = 0
x = x0
y = y0

while i <= steps:
    nigger.plot(x,y,"r.")
    x = x + xincrement
    y = y + yincrement
    i = i + 1
nigger.show()
```


10
8
6
4
2
0
0
2
4
6
8
10

```python
# PRACTICAL 4B: WAP to implement Bresenham's Line Drawing Algorithm

# This works only when slope is less than 1, i.e when the difference between xn and x0 is more than yn and y0

import matplotlib.pyplot as nigger

x0 = int(input("Enter x0: "))
y0 = int(input("Enter y0: "))
xn = int(input("Enter xn: "))
yn = int(input("Enter yn: "))

dx = xn - x0
dy = yn - y0

pk = (2 * dy) - dx

x = x0
y = y0

i = 0

while i <= dx:
    nigger.plot(x,y,"r.")
    if pk < 0:
        pknext = pk + ( 2 * dy)
    else:
        pknext = pk + (2 * (dy - dx))
        y = y + 1
    x = x + 1
    pk = pknext
    i = i + 1
nigger.show()
```

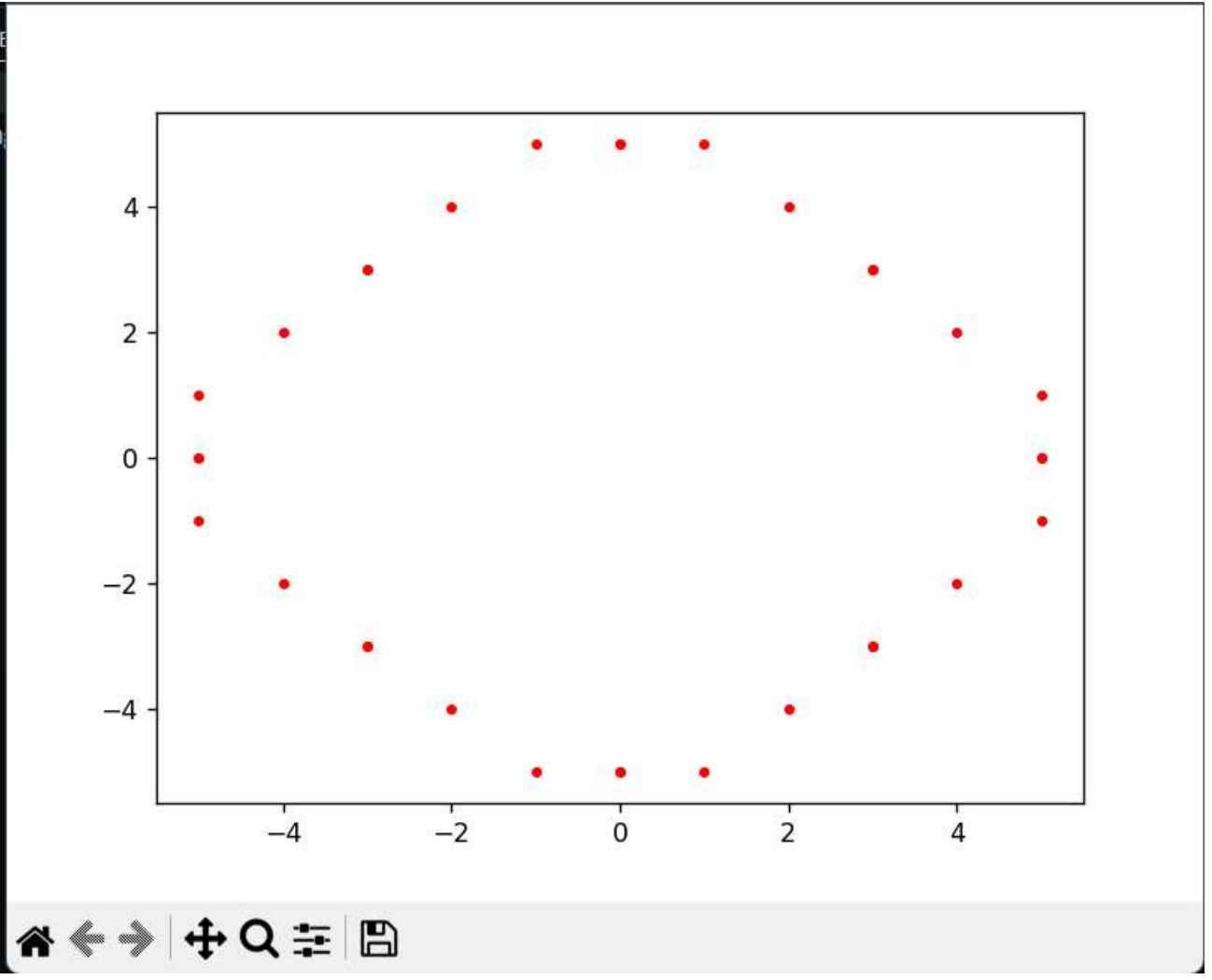

0
2
4
6
8
0
2
4
6
8
10

```python
# PRACTICAL 5A: WAP to implement Bresenham's Circle Drawing Algorithm

import matplotlib.pyplot as plt

def octants(x_sign, y_sign, flip, oct1_x_coords, oct1_y_coords, x_coords, y_coords, x_cen_coord, y_cen_coord):
    if flip:
        for i in oct1_x_coords:
            y_coords.append(y_sign * i + y_cen_coord)
        for i in oct1_y_coords:
            x_coords.append(x_sign * i + x_cen_coord)
    else:
        for i in oct1_x_coords:
            x_coords.append(x_sign * i + x_cen_coord)
        for i in oct1_y_coords:
            y_coords.append(y_sign * i + y_cen_coord)

x0 = int(input("Enter x0: "))
y0 = int(input("Enter y0: "))
rad = int(input("Enter Radius: "))

x = 0
y = rad

x_cord1 = []
y_cord1 = []

Pk = 3 - (2 * rad)

while x <= y:
    x_cord1.append(x)
    y_cord1.append(y)

    if Pk < 0:
        x += 1
        Pk += 4 * x + 6
    else:
        x += 1
        y -= 1
        Pk += 4 * (x - y) + 10

x_cord = []
y_cord = []

octants(1,  1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(1,  1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(1, -1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(1, -1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1,-1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1,-1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1, 1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1, 1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)

for i, j in zip(x_cord, y_cord):
    plt.plot(i, j, "r.")
plt.show()
```

PROBLEMS OUTPUT TE

TERMINAL

```
PS C:\Programmin
Enter x0: 0
Enter y0: 0
Enter Radius: 5
```

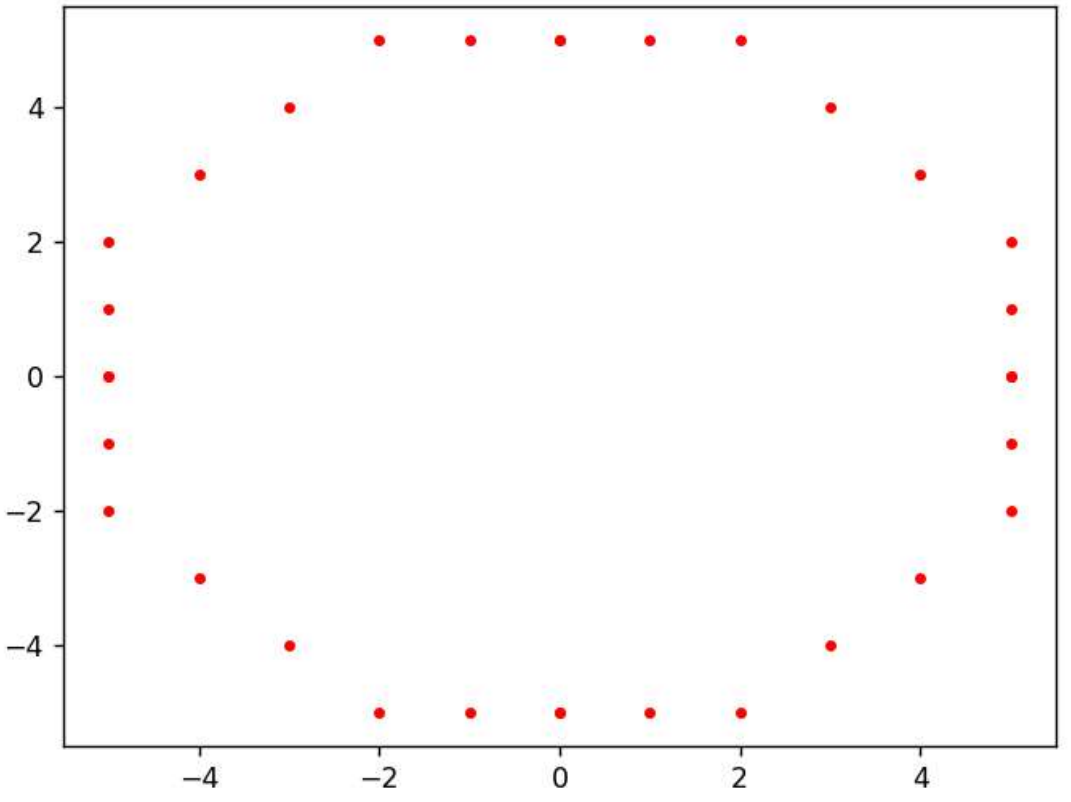


```python
# PRACTICAL 5B: WAP to implement Mid-Point Circle Drawing Algorithm

import matplotlib.pyplot as plt

def octants(x_sign, y_sign, flip, oct1_x_coords, oct1_y_coords, x_coords, y_coords, x_cen_coord, y_cen_coord):
    if flip:
        for i in oct1_x_coords:
            y_coords.append(y_sign * i + y_cen_coord)
        for i in oct1_y_coords:
            x_coords.append(x_sign * i + x_cen_coord)
    else:
        for i in oct1_x_coords:
            x_coords.append(x_sign * i + x_cen_coord)
        for i in oct1_y_coords:
            y_coords.append(y_sign * i + y_cen_coord)

x0 = int(input("Enter x0: "))
y0 = int(input("Enter y0: "))
rad = int(input("Enter Radius: "))

x = 0
y = rad

x_cord1 = []
y_cord1 = []

Pk = 1 - rad

while x <= y:
    x_cord1.append(x)
    y_cord1.append(y)

    if Pk < 0:
        x += 1
        Pk = Pk + (2 * x) + 1
    else:
        x += 1
        y -= 1
        Pk = Pk - 2 * (y - x) + 1

x_cord = []
y_cord = []

octants(1,  1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(1,  1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(1, -1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(1, -1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1,-1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1,-1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1, 1, True,  x_cord1, y_cord1, x_cord, y_cord, x0, y0)
octants(-1, 1, False, x_cord1, y_cord1, x_cord, y_cord, x0, y0)

for i, j in zip(x_cord, y_cord):
    plt.plot(i, j, "r.")
plt.show()
```

PROBLEMS OUTPUT TE

TERMINAL

```
PS C:\Programming
Enter x0: 0
Enter y0: 0
Enter Radius: 5
```

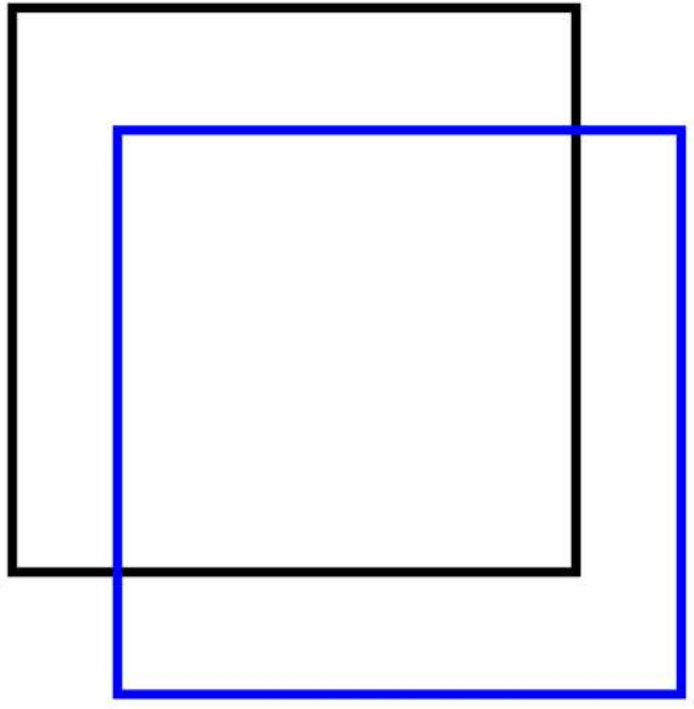


```python
#Practical 6: WAP to accept co-ordinates for the given object from the user & perform 2D translation

import tkinter

def matrixAddition(X: list, Y:list) -> list:
    Z = [0] * len(X) # result matrix
    for i in range(len(X)):
        Z[i] = X[i] + Y[i]
    return Z

# (x1,y1) are co-ordinates vertex of rectangle/square and (x2,y2) are other end vertex (suppose they
# are end points of diagonal of rectangle/square)
x1 = int(input("Enter x1 :"))
y1 = int(input("Enter y1 :"))
x2 = int(input("Enter x2 :"))
y2 = int(input("Enter y2 :"))
tx = int(input("Enter Translating Factor for x :"))
ty = int(input("Enter Translating Factor for y :"))

A = [x1,y1]
B = [x2,y2]
T = [tx,ty]

Anot = matrixAddition(A,T) #Anot will have co-ordinates of new rectangle and same for Cnot
Bnot = matrixAddition(B,T)

root = tkinter.Tk()
root.geometry("1000x1000")
C = tkinter.Canvas(root,bg="white",height=800, width=800)
C.create_rectangle(A + B, outline="black",width=5) #black colored = old figure
C.create_rectangle(Anot + Bnot, outline="blue",width=5) #blue colored = new figure (after translation)
C.pack()
root.mainloop()
```

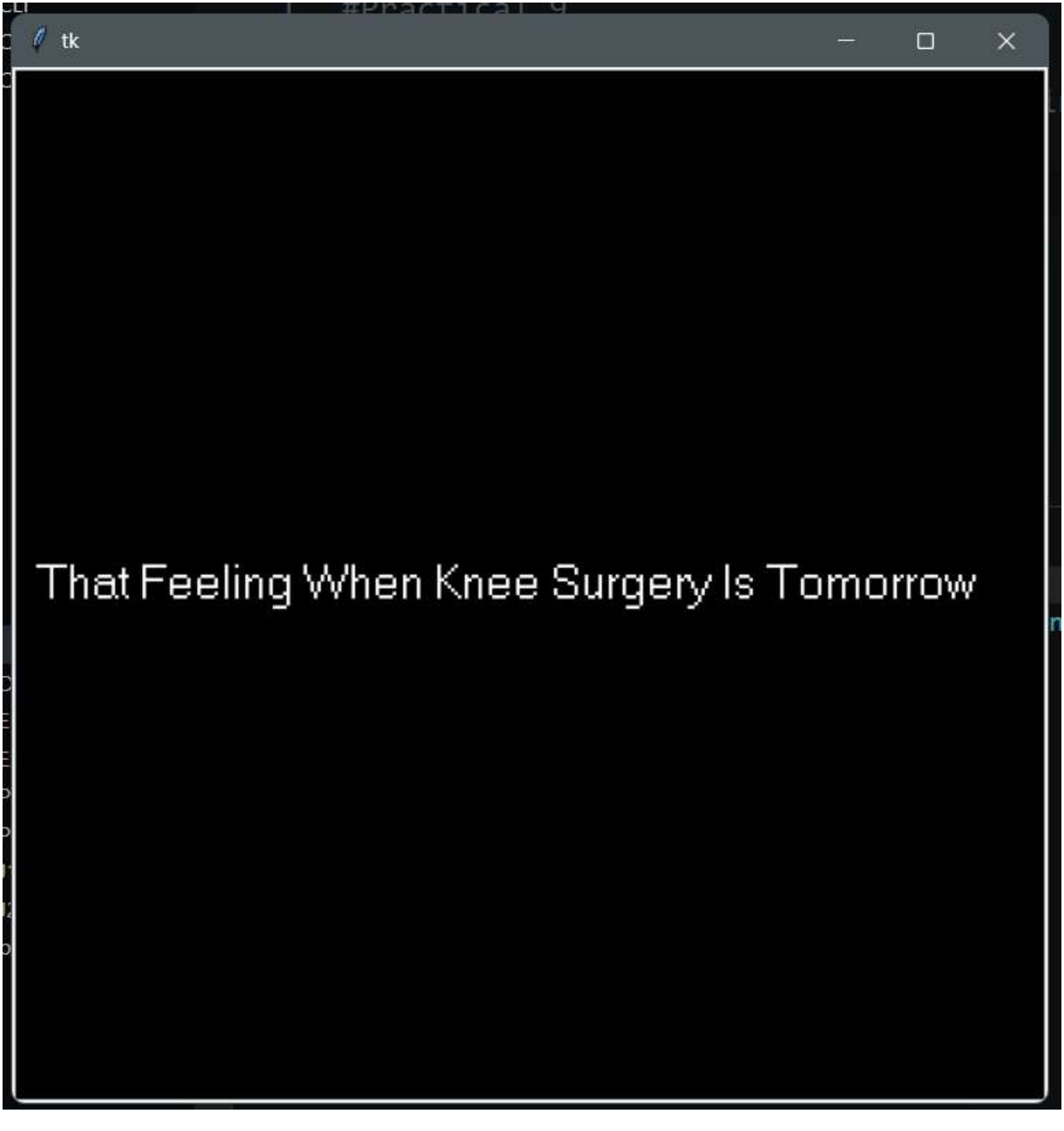
PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
PS C:\Programming\Python> python -u "c:\
Enter x1 :69
Enter y1 :69
Enter x2 :369
Enter y2 :369
Enter Translating Factor for x :56
Enter Translating Factor for y :65
```



```python
#Practical 9
#Text Screen Saver
#Design a simple text screen saver using graphic functions

from tkinter import *
root=Tk()

def scroll_text():
    c.move(text,-2,0)
    x1,y1,x2,y2=(c.bbox(text))
    if x2<0:
        c.coords(text,600,300)

    c.after(10,scroll_text)
c=Canvas(root,height=600,width=600,bg="black")

message="That Feeling When Knee Surgery Is Tomorrow"
text=c.create_text(600,300,text=message,fill="white",font=("ms sans serif" ,20))
scroll_text()
c.pack()

root.mainloop()
```


tk
That Feeling When Knee Surgery Is Tomorrow

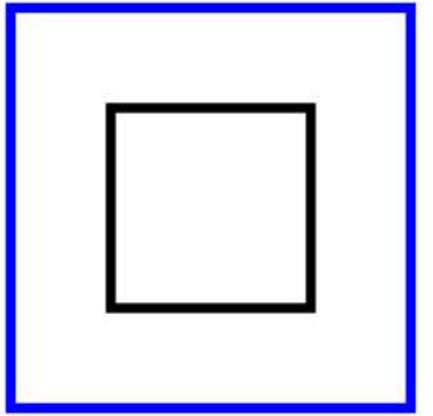
```python
#Practical 7 - Scaling.


#Documentation by ChatGPT.

import tkinter

def matrixMultiplication(X: list, Y: list) -> list:
    """
    Performs matrix multiplication of a 2D point with a 2x2 transformation matrix.

    Parameters:
    X (list): A 2-element list representing a point [x, y].
    Y (list): A 2x2 transformation matrix.

    Returns:
    list: The transformed point after multiplication.
    """
    Z = [0, 0]
    for i in range(2):
        for j in range(2):
            Z[i] += X[j] * Y[i][j]
    return Z

# Taking user input for the original rectangle coordinates
x1 = int(input("Enter top-left x coordinate: "))
y1 = int(input("Enter top-left y coordinate: "))
x2 = int(input("Enter bottom-right x coordinate: "))
y2 = int(input("Enter bottom-right y coordinate: "))

# Taking user input for scaling factors
sx = int(input("Enter scaling factor for x: "))
sy = int(input("Enter scaling factor for y: "))

# Calculate the center of the rectangle
cx = (x1 + x2) // 2
cy = (y1 + y2) // 2

# Convert points to local coordinates relative to the center
A = [x1 - cx, y1 - cy]
B = [x2 - cx, y2 - cy]

# Scaling transformation matrix
S = [[sx, 0], [0, sy]]

# Apply scaling transformation
Anot = matrixMultiplication(A, S)
Bnot = matrixMultiplication(B, S)

# Translate points back to original position
Anot = [Anot[0] + cx, Anot[1] + cy]
Bnot = [Bnot[0] + cx, Bnot[1] + cy]

# Initialize Tkinter window
root = tkinter.Tk()
root.geometry("1000x1000")  # Set window size

# Create a canvas for drawing
C = tkinter.Canvas(root, bg="white", height=800, width=800)

# Offset to center the drawing in Tkinter's coordinate system
xoffset = 400
yoffset = 400

# Draw original rectangle (black)
C.create_rectangle(
    x1 + xoffset, y1 + yoffset, x2 + xoffset, y2 + yoffset,
    outline="black", width=5
)

# Draw scaled rectangle (blue)
C.create_rectangle(
    Anot[0] + xoffset, Anot[1] + yoffset,
    Bnot[0] + xoffset, Bnot[1] + yoffset,
    outline="blue", width=5
)

C.pack()
root.mainloop()
```

```
import tkinter

def matrixMultiplication(X
    """
    Performs matrix multip

```

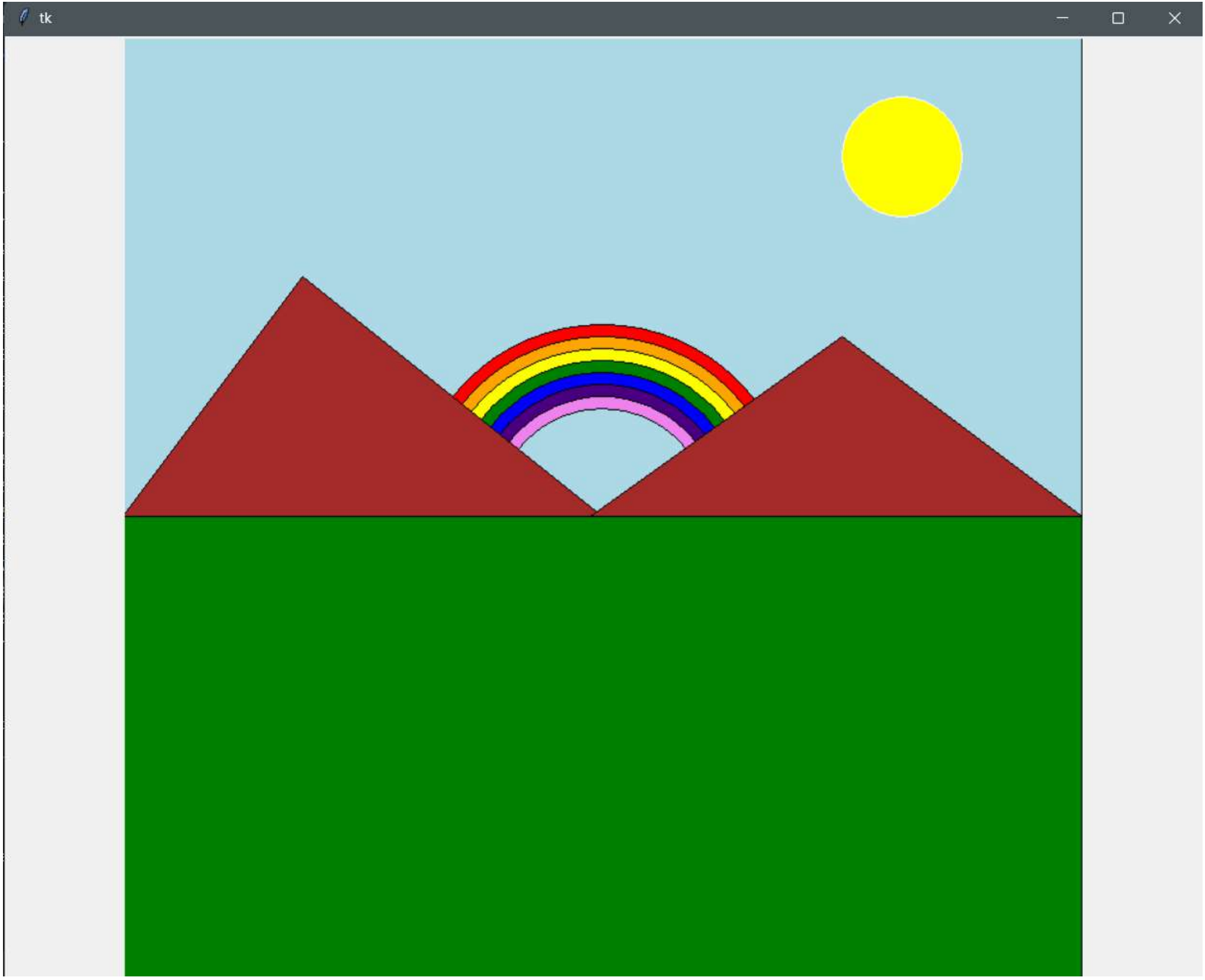
PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

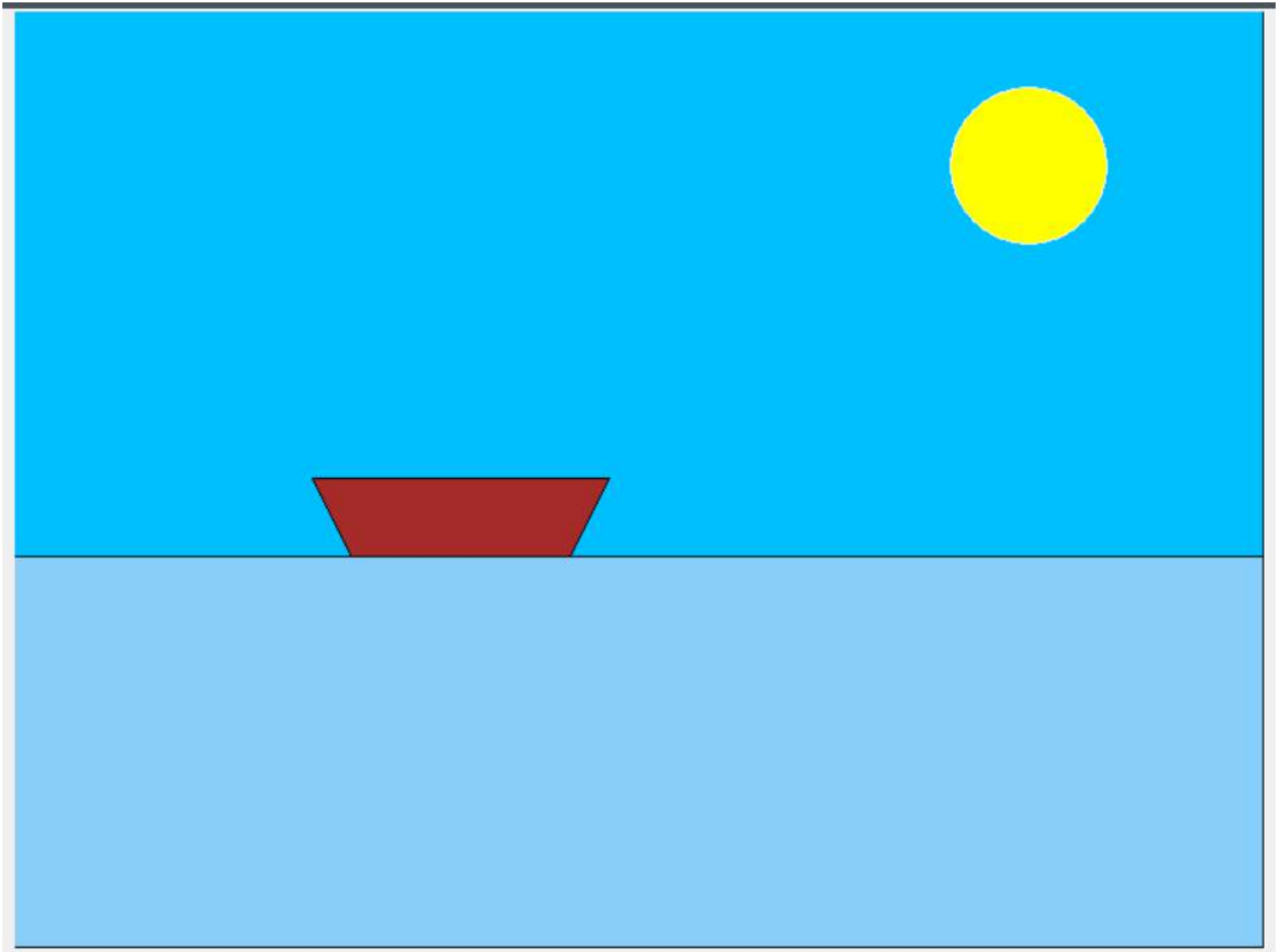
```
PS C:\Programming\Python> python -u "c:\P
Enter top-left x coordinate: 0
Enter top-left y coordinate: 0
Enter bottom-right x coordinate: 100
Enter bottom-right y coordinate: 100
Enter scaling factor for x: 2
Enter scaling factor for y: 2
```



```python
#Practical 8 - Chinary (Scenery)


import tkinter


root = tkinter.Tk()
root.geometry("1000x1000")
C = tkinter.Canvas(root,bg="white",height=800, width=800)
C.create_rectangle(0,0,800,400,outline="black",fill="lightblue")
C.create_oval(240,240,560,560,fill="red")
C.create_oval(250,250,550,550,fill="orange")
C.create_oval(260,260,540,540,fill="yellow")
C.create_oval(270,270,530,530,fill="green")
C.create_oval(280,280,520,520,fill="blue")
C.create_oval(290,290,510,510,fill="indigo")
C.create_oval(300,300,500,500,fill="violet")
C.create_oval(310,310,490,490,fill="lightblue")
C.create_polygon(0,400,400,400,150,200,outline="black", fill="brown")
C.create_polygon(390,400,800,400,600,250,outline="black", fill="brown")
C.create_oval(600,50,700,150,outline="white",fill="yellow")
C.create_rectangle(0,400,800,800,outline="black",fill="green")
C.pack()
root.mainloop()
```


tk

```python
#Practical 10b - Animation
#Moving Boat

def move_boat():
    C.move(boat,1,0)
    x1,y1,x2,y2 = C.bbox(boat)
    if x1 >= 800:
        C.coords(boat,10,300,200,300,175,350,35,350)
    root.after(10,move_boat)

import tkinter

root = tkinter.Tk()
root.geometry("1000x1000")

# Create a canvas for drawing
C = tkinter.Canvas(root, bg="white", height=600, width=800)
C.create_rectangle(0,0,800,350,outline="black",fill="deepskyblue")
C.create_rectangle(0,350,800,600,outline="black",fill="lightskyblue")
C.create_oval(600,50,700,150,outline="white",fill="yellow")
C.pack()
boat = C.create_polygon(10,300,200,300,175,350,35,350,outline="black",fill="brown")
root.after(10,move_boat)
root.mainloop()
```



```python
#Practical 10a
#WAP to animate a smiley

import tkinter

def smile():
    global start_angle,end_angle
    if end_angle > -140:
        end_angle -= 5
    else:
        end_angle = 5
    C.itemconfig(mouth, start = start_angle, extent = end_angle,style="arc",width="5")
    root.after(50,smile)

root = tkinter.Tk()
root.geometry("1000x1000")
C = tkinter.Canvas(root, height=800, width=845, bg="lightblue")
C.create_oval(200,200,600,600,fill="yellow")
C.create_oval(310,310,340,360,fill="black")
C.create_oval(460,310,490,360,fill="black")
C.pack()
start_angle = -20
end_angle = -5
mouth = C.create_arc(300,400,500,525,start=start_angle,extent=end_angle,style="arc",width="5")
smile()
root.mainloop()
```