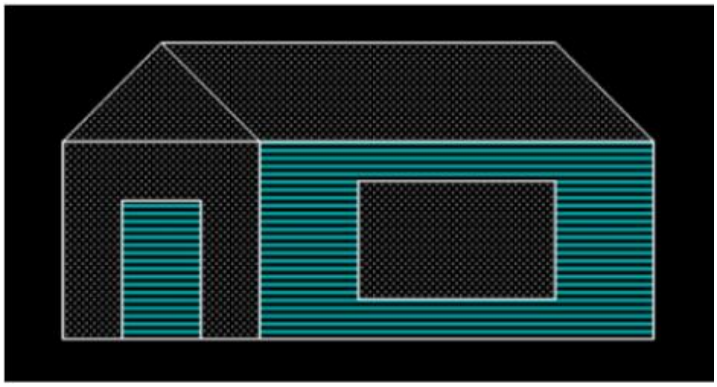
COMPUTER GRAPHICS AND ANIMATION

PRACTICAL QUESTIONS

1. Write a program to draw a co-ordinate axis at the center of the screen.
2. Write a program to divide your screen into four region, draw circle, rectangle, ellipse and line in each region with appropriate message.
3. Write a program to draw a simple hut on the screen.



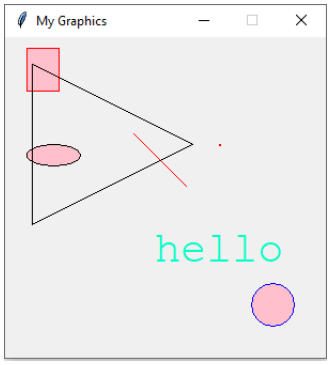
4. Write a program to draw a simple clock in the centre of the screen.

5. Write a program to Draw the following basic shapes with names in the centre of the screen :
    a. Circle
    b. Rectangle
    c. Square
    d. Concentric Circles
    e. Ellipse
    f. Line
    g. Polygon

```
from graphics import *
def main():
   win = GraphWin('My Graphics', 300, 300) # specifies graphics window size 250x250
```

```
    g = Point(200,100)# creates a Point object at x=125 y=125
    g.setOutline('red')
    g.draw(win) # draws to the graphics window
    cir = Circle(Point(250,250), 20)
    cir.draw(win)
    cir.setOutline('blue')
    cir.setFill('pink')

    txt= Text (Point (200, 200), "hello")
    txt.setTextColor (color_rgb(0, 255, 200))
    txt.setSize(30)
    txt. setFace('courier')
    txt.draw(win)

    line = Line(Point(100, 50), Point(150, 100))
    line.setOutline('red')
    line.move(20,40)
    line.draw(win)

    pt = Point(50, 50)
    rect = Rectangle(Point (20, 10), pt)
    rect.setOutline('red')
    rect.setFill('pink')
    rect.draw(win)

    oval = Oval(Point (20, 100), Point (70, 120))
    oval.setFill('pink')
    oval.draw(win)

    tri = Polygon(Point(25, 25), Point(175, 100), Point(25, 175))
    tri.draw(win)

    win.getMouse() # keep window up
    win.close()
main()
```

OUTPUT:

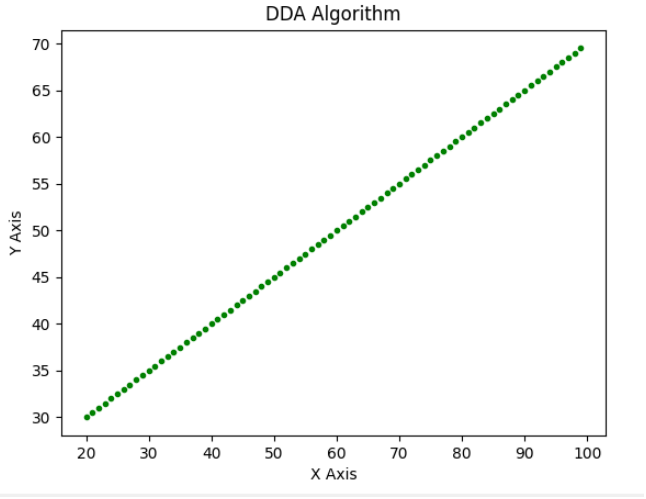


6. Write a program to Develop the program for DDA Line drawing algorithm.

```
import matplotlib.pyplot as plt

plt.title("DDA Algorithm")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")

def DDALine(x1, y1, x2, y2, color):
   dx = x2 - x1
   dy = y2 - y1

   # calculate steps required for generating pixels

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

   #calculate increment in x & y for each steps
   Xinc = float(dx / steps)
   Yinc = float(dy / steps)

   for i in range(0, int(steps)):
                  # Draw pixels
      plt.plot(float(x1), float(y1), color)
      x1 += Xinc
      y1 += Yinc
   plt.show()

def main():
   x = int(input("Enter X1: "))
   y = int(input("Enter Y1: "))
   xEnd = int(input("Enter X2: "))
   yEnd = int(input("Enter Y2: "))
   color = "g."
   DDALine(x, y, xEnd, yEnd, color)

if __name__ == '__main__':
   main()
```

OUTPUT:

```
Enter X1: 20
Enter Y1: 30
Enter X2: 100
Enter Y2: 70
```



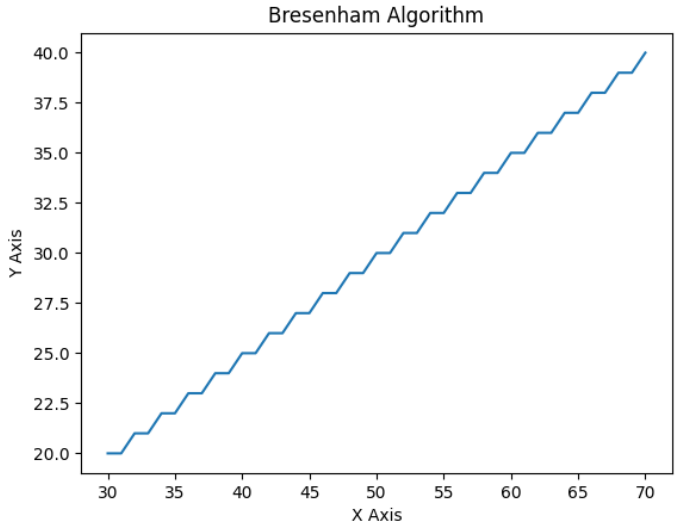
7. Write a program to Develop the program for Bresenham's Line drawing algorithm.

```
import matplotlib.pyplot as plt
plt.title("Bresenham Algorithm")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")

def bres(x1,y1,x2,y2):
  x,y = x1,y1
  dx = abs(x2 - x1)
  dy = abs(y2 -y1)
  gradient = dy/float(dx)

  if gradient > 1:
    dx, dy = dy, dx
    x, y = y, x
    x1, y1 = y1, x1
    x2, y2 = y2, x2

  p = 2*dy - dx
  print(f"x = {x}, y = {y}")
  # Initialize the plotting points
  xcoordinates = [x]
  ycoordinates = [y]

  for k in range(2, dx + 2):
    if p > 0:
      y = y + 1 if y < y2 else y - 1
      p = p + 2 * (dy - dx)
    else:
```

```
            p = p + 2 * dy

        x = x + 1 if x < x2 else x - 1

        print(f"x = {x}, y = {y}")
        xcoordinates.append(x)
        ycoordinates.append(y)

    plt.plot(xcoordinates, ycoordinates)
    plt.show()

def main():
    x1 = int(input("Enter the Starting point of x: "))
    y1 = int(input("Enter the Starting point of y: "))
    x2 = int(input("Enter the end point of x: "))
    y2 = int(input("Enter the end point of y: "))

    bres(x1, y1, x2, y2)

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

OUTPUT:

```
Enter the Starting point of x: 30
Enter the Starting point of y: 20
Enter the end point of x: 70
Enter the end point of y: 40
x = 30, y = 20
x = 31, y = 20
x = 32, y = 21
x = 33, y = 21
x = 34, y = 22
x = 35, y = 22
x = 36, y = 23
x = 37, y = 23
x = 38, y = 24
x = 39, y = 24
x = 40, y = 25
x = 41, y = 25
x = 42, y = 26
x = 43, y = 26
x = 44, y = 27
x = 45, y = 27
x = 46, y = 28
x = 47, y = 28
x = 48, y = 29
x = 49, y = 29
x = 50, y = 30
x = 51, y = 30
x = 52, y = 31
x = 53, y = 31
x = 54, y = 32
x = 55, y = 32
x = 56, y = 33
x = 57, y = 33
x = 58, y = 34
x = 59, y = 34
x = 60, y = 35
x = 61, y = 35
x = 62, y = 36
x = 63, y = 36
x = 64, y = 37
x = 65, y = 37
x = 66, y = 38
x = 67, y = 38
x = 68, y = 39
x = 69, y = 39
x = 70, y = 40
```



8. Write a program to Develop the program for the mid-point circle drawing algorithm.

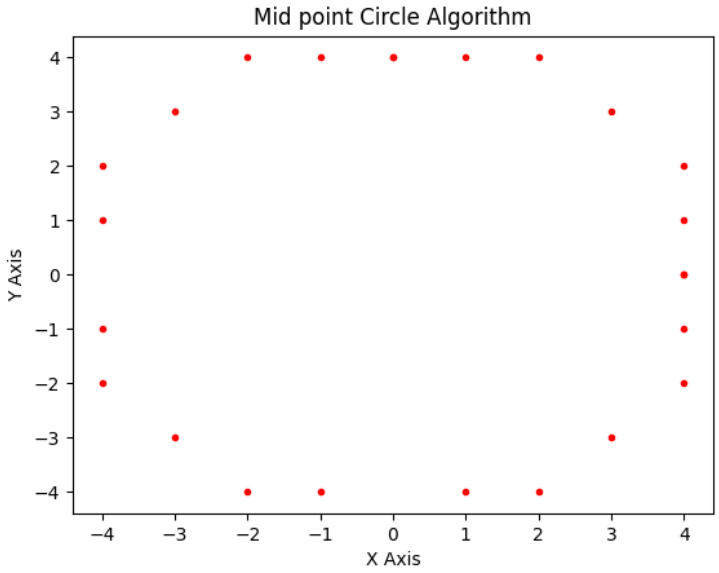
```
import matplotlib.pyplot as plt

plt.title("Mid point Circle Algorithm")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
color = "r."
def midPointCircleDraw(x_centre, y_centre, r):
        x = r
        y = 0

        # Printing the initial point the axes after translation
        print("(", x + x_centre, ", ", y + y_centre, ")")
        plt.plot(float(x + x_centre), float(y + y_centre), color)

        # When radius is zero only a single point be printed

        if (r > 0) :

                print("(", x + x_centre, ", ",-y + y_centre, ")")
                plt.plot(float(x + x_centre), float(-y + y_centre), color)
                print("(", y + x_centre, ", ", x + y_centre, ")")
                plt.plot(float(y + x_centre), float(x + y_centre), color)
                print("(", -y + x_centre, ", ", x + y_centre, ")")
                plt.plot(float(-y + x_centre), float(x + y_centre), color)

        # Initialising the value of P
        P = 1 - r
```

```
        while x > y:

            y += 1

            # Mid-point inside or on the perimeter
            if P <= 0:
                P = P + 2 * y + 1

            # Mid-point outside the perimeter
            else:
                x -= 1
                P = P + 2 * y - 2 * x + 1

            # All the perimeter points have
            # already been printed
            if (x < y):
                break

            # Printing the generated point its reflection
            # in the other octants after translation

            print("(", x + x_centre, ", ", y + y_centre,")")
            plt.plot(float(x + x_centre), float(y + y_centre), color)
            print("(", -x + x_centre, ", ", y + y_centre, ")")
            plt.plot(float(-x + x_centre), float(y + y_centre), color)
            print("(", x + x_centre, ", ", -y + y_centre,")")
            plt.plot(float(x + x_centre), float(-y + y_centre), color)
            print("(", -x + x_centre, ", ", -y + y_centre,")")
            plt.plot(float(-x + x_centre), float(-y + y_centre), color)

            # If the generated point on the line x = y then
            # the perimeter points have already been printed
            if x != y:

                print("(", y + x_centre, ", ", x + y_centre,")")
                plt.plot(float(y + x_centre), float(x + y_centre), color)
                print("(", -y + x_centre, ", ", x + y_centre,")")
                plt.plot(float(-y + x_centre), float(x + y_centre), color)
                print("(", y + x_centre, ", ", -x + y_centre,")")
                plt.plot(float(y + x_centre), float(-x + y_centre), color)
                print("(", -y + x_centre, ", ", -x + y_centre,")")
                plt.plot(float(-y + x_centre), float(-x + y_centre), color)
        plt.show()

# Driver Code
if __name__ == '__main__':
```

```
# To draw a circle of radius 3
# centered at (0, 0)
midPointCircleDraw(0, 0, 4)
```

OUTPUT:

```
( 4 ,  0 )
( 4 ,  0 )
( 0 ,  4 )
( 0 ,  4 )
( 4 ,  1 )
( -4 ,  1 )
( 4 ,  -1 )
( -4 ,  -1 )
( 1 ,  4 )
( -1 ,  4 )
( 1 ,  -4 )
( -1 ,  -4 )
( 4 ,  2 )
( -4 ,  2 )
( 4 ,  -2 )
( -4 ,  -2 )
( 2 ,  4 )
( -2 ,  4 )
( 2 ,  -4 )
( -2 ,  -4 )
( 3 ,  3 )
( -3 ,  3 )
( 3 ,  -3 )
( -3 ,  -3 )
```



9. Write a program to Develop the program for the mid-point ellipse drawing algorithm.

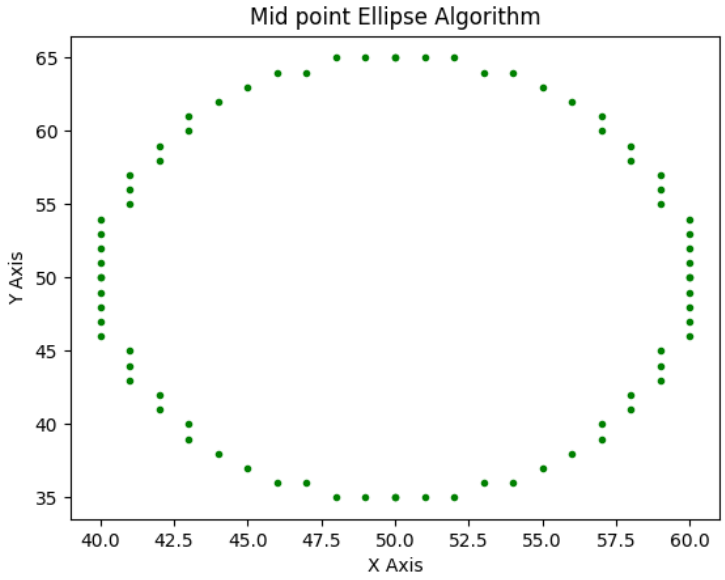
```
import matplotlib.pyplot as plt

plt.title("Mid point Ellipse Algorithm")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
color = "g."

def midptellipse(rx, ry, xc, yc):

    x = 0;
    y = ry;

    # Initial decision parameter of region 1
    d1 = ((ry * ry) - (rx * rx * ry) +(0.25 * rx * rx));
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
```

```
# For region 1
while (dx < dy):

        # Print points based on 4-way symmetry
        print("(", x + xc, ",", y + yc, ")");
        plt.plot(float(x + xc), float(y + yc), color)
        print("(",-x + xc,",", y + yc, ")");
        plt.plot(float(-x + xc), float(y + yc), color)
        print("(",x + xc,",", -y + yc ,")");
        plt.plot(float(x + xc), float(-y + yc), color)
        print("(",-x + xc, ",", -y + yc, ")");
        plt.plot(float(-x + xc), float(-y + yc), color)

        # Checking and updating value of
        # decision parameter based on algorithm
        if (d1 < 0):
                x += 1;
                dx = dx + (2 * ry * ry);
                d1 = d1 + dx + (ry * ry);
        else:
                x += 1;
                y -= 1;
                dx = dx + (2 * ry * ry);
                dy = dy - (2 * rx * rx);
                d1 = d1 + dx - dy + (ry * ry);

# Decision parameter of region 2
d2 = (((ry * ry) * ((x + 0.5) * (x + 0.5))) +
        ((rx * rx) * ((y - 1) * (y - 1))) -
        (rx * rx * ry * ry));

# Plotting points of region 2
while (y >= 0):

        # printing points based on 4-way symmetry
        print("(", x + xc, ",", y + yc, ")");
        plt.plot(float(x + xc), float(y + yc), color)
        print("(", -x + xc, ",", y + yc, ")");
        plt.plot(float(-x + xc), float(y + yc), color)
        print("(", x + xc, ",", -y + yc, ")");
        plt.plot(float(x + xc), float(-y + yc), color)
        print("(", -x + xc, ",", -y + yc, ")");
        plt.plot(float(-x + xc), float(-y + yc), color)

        # Checking and updating parameter
        # value based on algorithm
```

```
                if (d2 > 0):
                        y -= 1;
                        dy = dy - (2 * rx * rx);
                        d2 = d2 + (rx * rx) - dy;
                else:
                        y -= 1;
                        x += 1;
                        dx = dx + (2 * ry * ry);
                        dy = dy - (2 * rx * rx);
                        d2 = d2 + dx - dy + (rx * rx);
        plt.show()

# Driver code
midptellipse(10, 15, 50, 50);
```

OUTPUT:

```
( 50 , 65 )
( 50 , 65 )
( 50 , 35 )
( 50 , 35 )
( 51 , 65 )
( 49 , 65 )
( 51 , 35 )
( 49 , 35 )
( 52 , 65 )
( 48 , 65 )
( 52 , 35 )
( 48 , 35 )
( 53 , 64 )
( 47 , 64 )
( 53 , 36 )
( 47 , 36 )
( 54 , 64 )
( 46 , 64 )
( 54 , 36 )
( 46 , 36 )
( 55 , 63 )
( 45 , 63 )
( 55 , 37 )
( 45 , 37 )
( 59 , 56 )
( 41 , 56 )
( 59 , 44 )
( 41 , 44 )
( 59 , 55 )
( 41 , 55 )
( 59 , 45 )
( 41 , 45 )
( 60 , 54 )
( 40 , 54 )
( 60 , 46 )
( 40 , 46 )
( 60 , 53 )
( 40 , 53 )
( 60 , 47 )
( 40 , 47 )
( 60 , 52 )
( 40 , 52 )
( 60 , 48 )
( 40 , 48 )
( 60 , 51 )
( 40 , 51 )
( 60 , 49 )
( 40 , 49 )
( 60 , 50 )
( 40 , 50 )
( 60 , 50 )
( 40 , 50 )
```



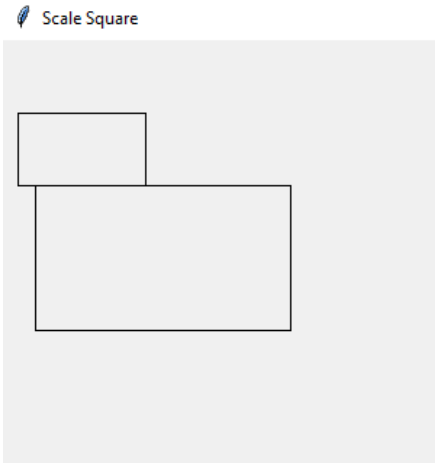
10. Write a program to implement 2D scaling.

```
from graphics import *
win = GraphWin("Scale Square", 600, 600)
print("Corner 1")
x1=int(input("Enter x"))
y1=int(input("Enter y"))
c1=Point(x1, y1)

print("Corner 2")
x2=int(input("Enter x"))
y2=int(input("Enter y"))
c2 = Point(x2, y2)

s = Rectangle(c1, c2)
s.draw(win)

sx=float(input("Scaling Factor sx"))
sy=float(input("Scaling Factor sy"))

x1*=sx
x2*=sx
y1*=sy
y2*=sy
c1=Point(x1, y1)
c2 = Point(x2, y2)

ss=Rectangle(c1, c2)
ss.draw(win)
```

```
win.getMouse()
win.close()
```

OUTPUT:

```
Corner 1
Enter x 12
Enter y100
Corner 2
Enter x100
Enter y50
Scaling Factor sx 2
Scaling Factor sy 2
```



11. Write a program to perform 2D translation.

12. Write a program to Develop a simple text screen saver using graphics functions.

```
from graphics import *
import random, time

win = GraphWin("Random Circles", 300, 300)

#creating 75 random circle objects
for i in range(100):

#randrange takes one integer argument which defines the intensity of red, blue, and green colors
   r = random.randrange(250)
   b = random.randrange(250)
   g = random.randrange(250)
   color = color_rgb(r, b, g)
   a = random.randrange(10)
   x = random.randrange(300)
   y = random.randrange(300)
   p = Point(x,y)
   #creating circle objects with different radius ranging from 3 to 40
   txt = Text(p,"hello")
   txt.setTextColor(color)
   txt.setSize(30)
   txt. setFace('courier')
   txt.draw(win)

   time.sleep(1.0)
```

```
txt.undraw()
```

OUTPUT:

13. Write a program to Perform smiling face animation using graphic functions.
14. Write a program to draw the moving car on the screen.
15. Write a program to Perform rolling ball animation using graphic functions.

16. Write a program to Develop a program for creating random circle objects.

```
from graphics import *
import random, time
win = GraphWin("Random Circles", 300, 300)

#creating 75 random circle objects
for circle in range(100):

#randrange takes one integer argument which defines the intensity of red, blue,
and green colors
   r = random.randrange(250)
   b = random.randrange(250)
   g = random.randrange(250)
   color = color_rgb(r, b, g)

   #creating circle objects with different radius ranging from 3 to 40
   radius = random.randrange(3, 10)
   x = random.randrange(5, 295)
   y = random.randrange(5, 295)
   circle = Circle(Point(x,y), radius)

   #painting circle objects with different colors
   circle.setFill(color)
   circle.draw(win)

#time taken for each circle object to appear
```

```
time.sleep(.05)
```

OUTPUT: