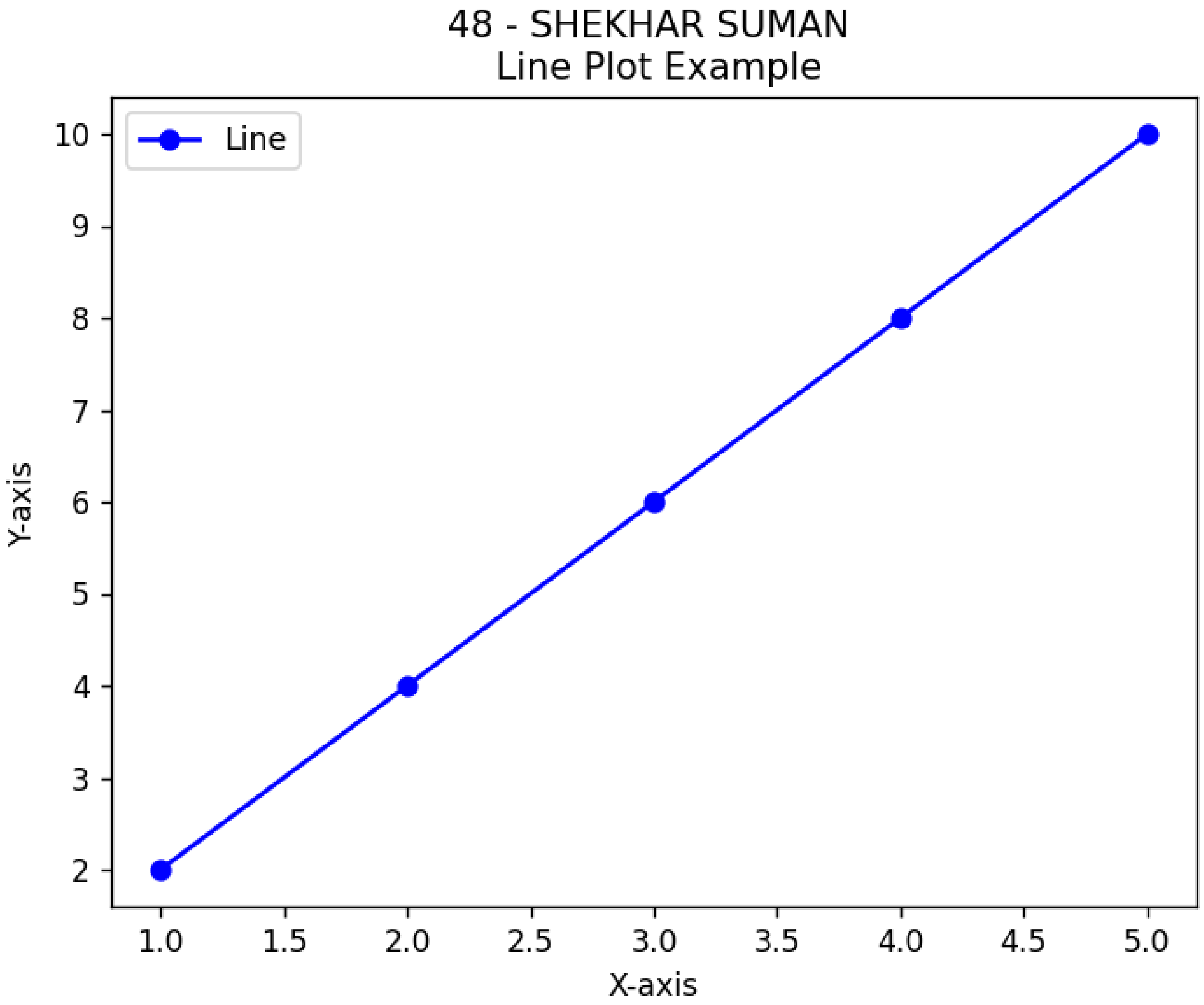
PRACTICAL: 1-A

AIM: Write a program to draw Line Plot

CODE

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y, marker='o', linestyle='-', color='b', label='Line')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('48 - SHEKHAR SUMAN \n Line Plot Example')
plt.legend()
plt.show()
```

OUTPUT



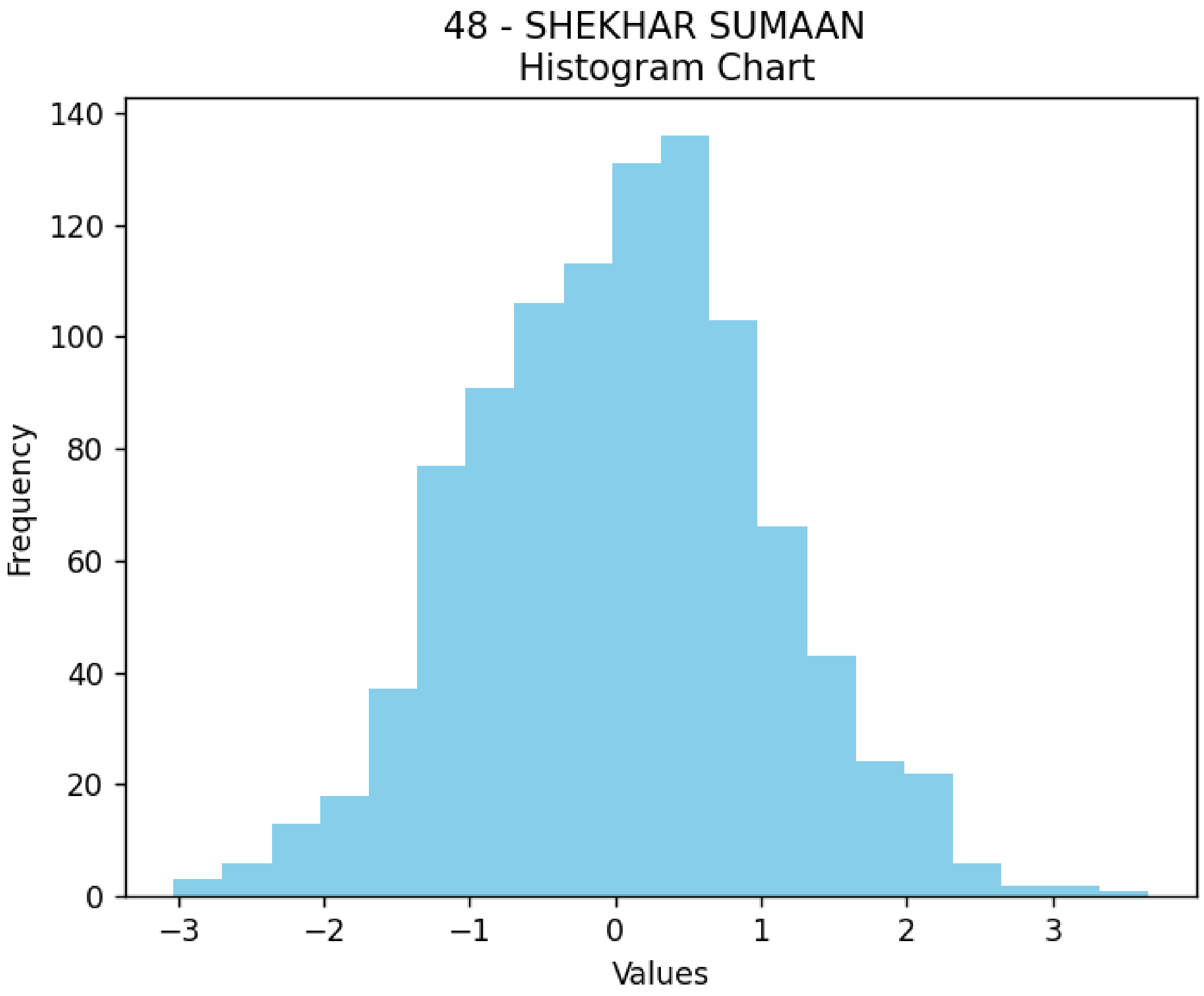
PRACTICAL: 1-B

AIM: Write a program to Developed a Histogram chart

CODE

```
import matplotlib.pyplot as plt
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=20, color='skyblue')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('48 - SHEKHAR SUMAAN \n Histogram Chart')
plt.show()
```

OUTPUT



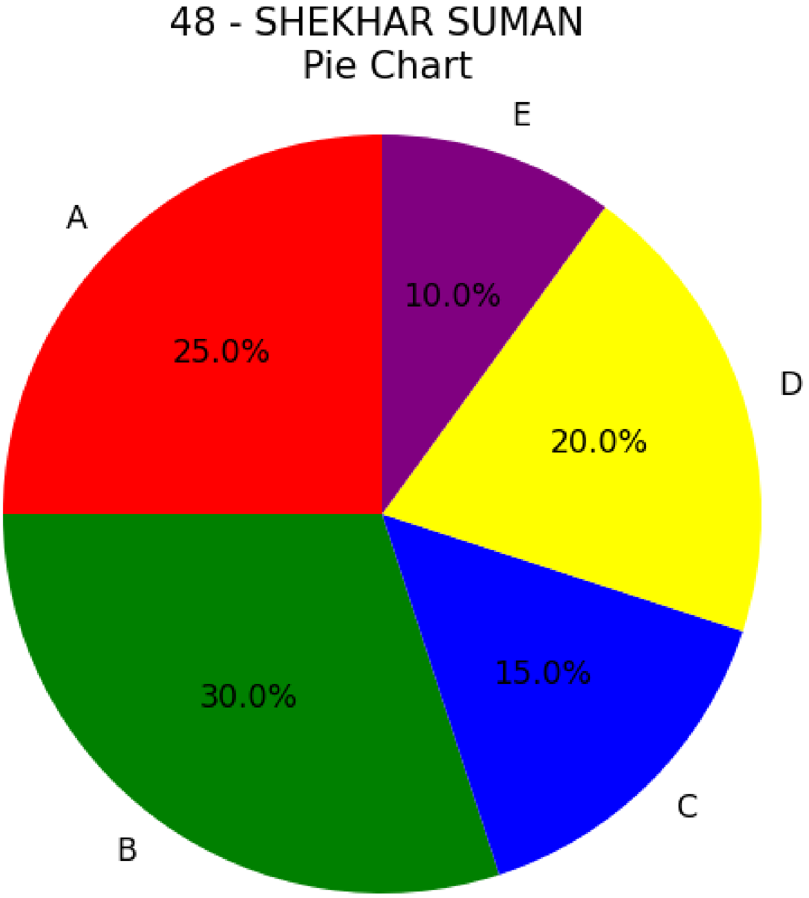
PRACTICAL: 1-C

AIM: Write a program to developed Pie Chart

CODE

```
import matplotlib.pyplot as plt

sizes = [25, 30, 15, 20, 10]
labels = ['A', 'B', 'C', 'D', 'E']

plt.pie(sizes, labels=lables, autopct='%1.1f%%', startangle=90, colors=['red', 'green', 'blue', 'yellow', 'purple'])

plt.axis('equal')

plt.title('48 - SHEKHAR SUMAN \n Pie Chart')
plt.show()
```

OUTPUT



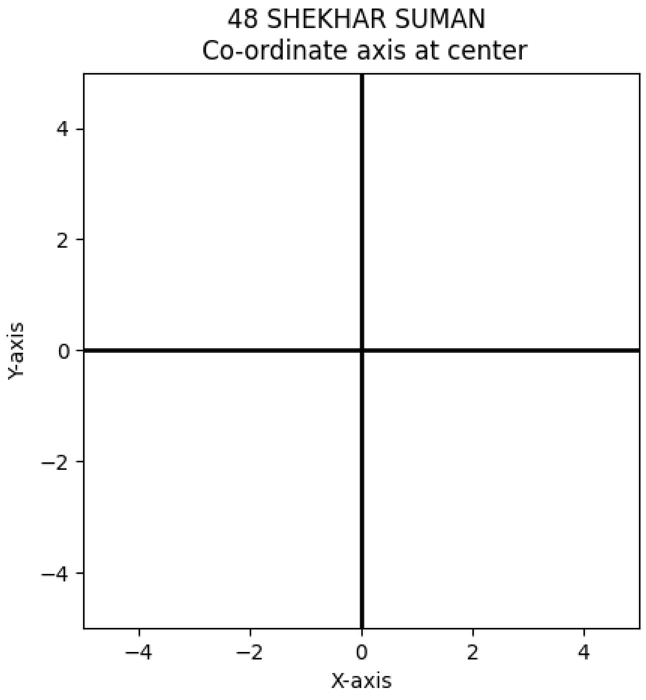
PRACTICAL: 2-A

AIM: Draw a co-ordinate axis at the center of the screen.

CODE

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.set_xlim(-5, 5)
ax.set_ylim(-5, 5)
ax.axhline(0, color='black', lw=2)
ax.axvline(0, color='black', lw=2)
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_aspect('equal', adjustable='box')
plt.title('48 SHEKHAR SUMAN \n Co-ordinate axis at center')
plt.show()
```

OUTPUT



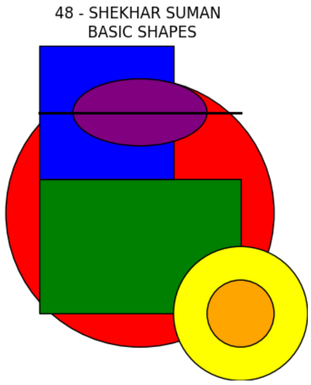
PRACTICAL: 3-A

AIM: Write a program to draw basic shape.

CODE

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
fig, ax = plt.subplots()

circle = plt.Circle((0.5, 0.5), 0.4, edgecolor='black', facecolor='red')
ax.add_patch(circle)

rectangle = plt.Rectangle((0.2, 0.2), 0.6, 0.4, edgecolor='black', facecolor='green')
ax.add_patch(rectangle)

square = plt.Rectangle((0.2, 0.6), 0.4, 0.4, edgecolor='black', facecolor='blue')
ax.add_patch(square)

circle_outer = plt.Circle((0.8, 0.2), 0.2, edgecolor='black', facecolor='yellow')
circle_inner = plt.Circle((0.8, 0.2), 0.1, edgecolor='black', facecolor='orange')
ax.add_patch(circle_outer)
ax.add_patch(circle_inner)

ellipse = patches.Ellipse((0.5, 0.8), 0.4, 0.2, edgecolor='black', facecolor='purple')
ax.add_patch(ellipse)

line = plt.Line2D([0.2, 0.8], [0.8, 0.8], color='black', linewidth=2)
ax.add_line(line)

ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_aspect('equal', adjustable='box')
ax.axis('off')
plt.title('48 - SHEKHAR SUMAN \n BASIC SHAPES')
plt.show()
```

OUTPUT



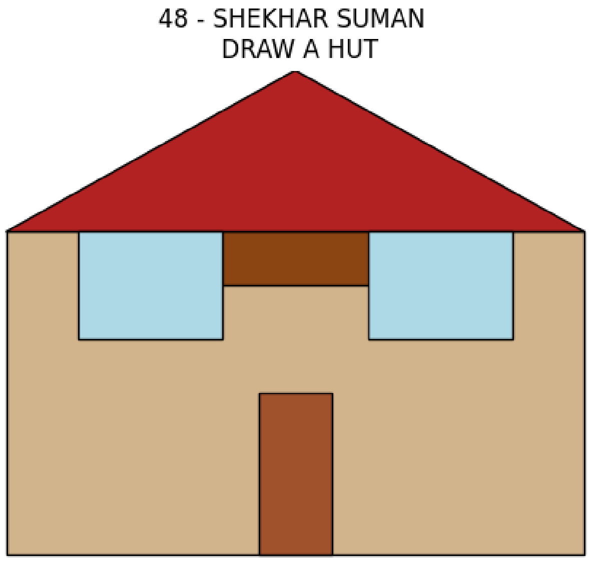
PRACTICAL: 3-B

AIM: Write a program to draw a hut.

CODE

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

fig, ax = plt.subplots()

ax.add_patch(patches.Rectangle((0.1, 0.1), 0.8, 0.6, edgecolor='black', facecolor='tan'))
ax.add_patch(patches.Rectangle((0.35, 0.6), 0.3, 0.1, edgecolor='black',
facecolor='saddlebrown'))

roof = plt.Polygon([(0.1, 0.7), (0.5, 1.0), (0.9, 0.7)], edgecolor='black', facecolor='firebrick')
ax.add_patch(roof)

ax.add_patch(patches.Rectangle((0.45, 0.1), 0.1, 0.3, edgecolor='black', facecolor='sienna'))

ax.add_patch(patches.Rectangle((0.2, 0.5), 0.2, 0.2, edgecolor='black', facecolor='lightblue'))
ax.add_patch(patches.Rectangle((0.6, 0.5), 0.2, 0.2, edgecolor='black', facecolor='lightblue'))

ax.set_xlim(0, 1)
ax.set_ylim(0, 1)

ax.axis('off')

plt.title('48 - SHEKHAR SUMAN \n DRAW A HUT')
plt.show()
```

OUTPUT



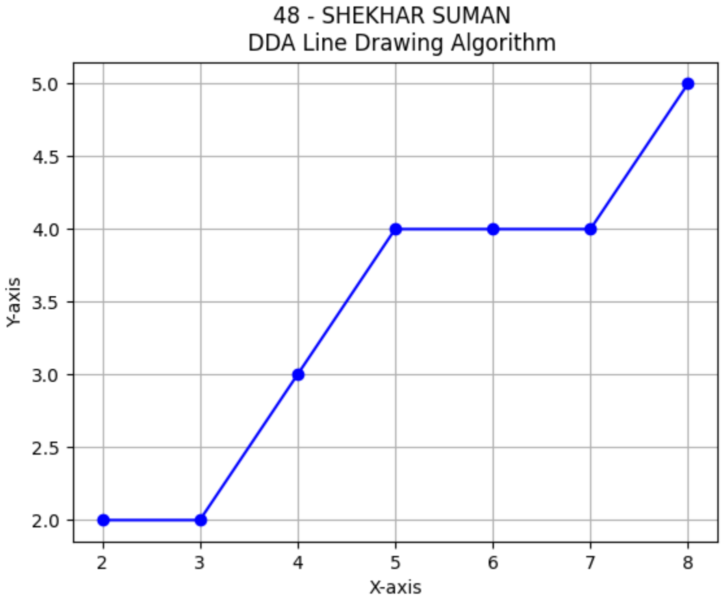
PRACTICAL: 4-A

AIM: Write a program to developed **DDA Line drawing algorithm**

CODE

```
import matplotlib.pyplot as plt
def dda_line(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    steps = max(abs(dx), abs(dy))
    x_increment = dx / steps
    y_increment = dy / steps
    x, y = x1, y1
    x_points = [x]
    y_points = [y]
    for _ in range(steps):
        x += x_increment
        y += y_increment
        x_rounded = round(x)
        y_rounded = round(y)
        x_points.append(x_rounded)
        y_points.append(y_rounded)
    return x_points, y_points
x1, y1 = 2, 2
x2, y2 = 8, 5
x_points, y_points = dda_line(x1, y1, x2, y2)
plt.plot(x_points, y_points, marker='o', linestyle='-', color='b')
plt.title('48 - SHEKHAR SUMAN \n  DDA Line Drawing Algorithm')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
```

OUTPUT



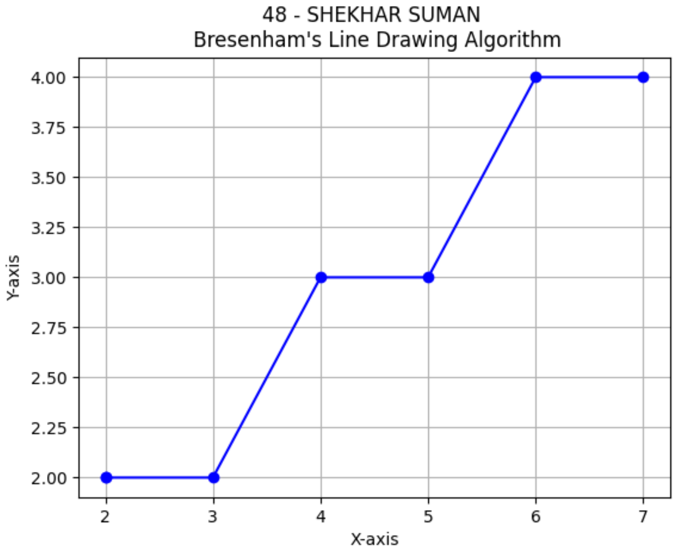
PRACTICAL: 4-B

AIM: Write a program to developed **Bresenham's Line Drawing Algorithm**.

CODE

```
import matplotlib.pyplot as plt
def bresenham_line(x1, y1, x2, y2):
    dx = abs(x2 - x1)
    dy = abs(y2 - y1)
    sx = 1 if x1 < x2 else -1
    sy = 1 if y1 < y2 else -1
    x, y = x1, y1
    error = dx - dy
    x_points = [x]
    y_points = [y]
    while x != x2 or y != y2:
        x_points.append(x)
        y_points.append(y)
        error2 = 2 * error
        if error2 > -dy:
            error -= dy
            x += sx
        if error2 < dx:
            error += dx
            y += sy
    return x_points, y_points
x1, y1 = 2, 2
x2, y2 = 8, 5
x_points, y_points = bresenham_line(x1, y1, x2, y2)
plt.plot(x_points, y_points, marker='o', linestyle='-', color='b')
plt.title("48 - SHEKHAR SUMAN \n Bresenham's Line Drawing Algorithm")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
```

OUTPUT



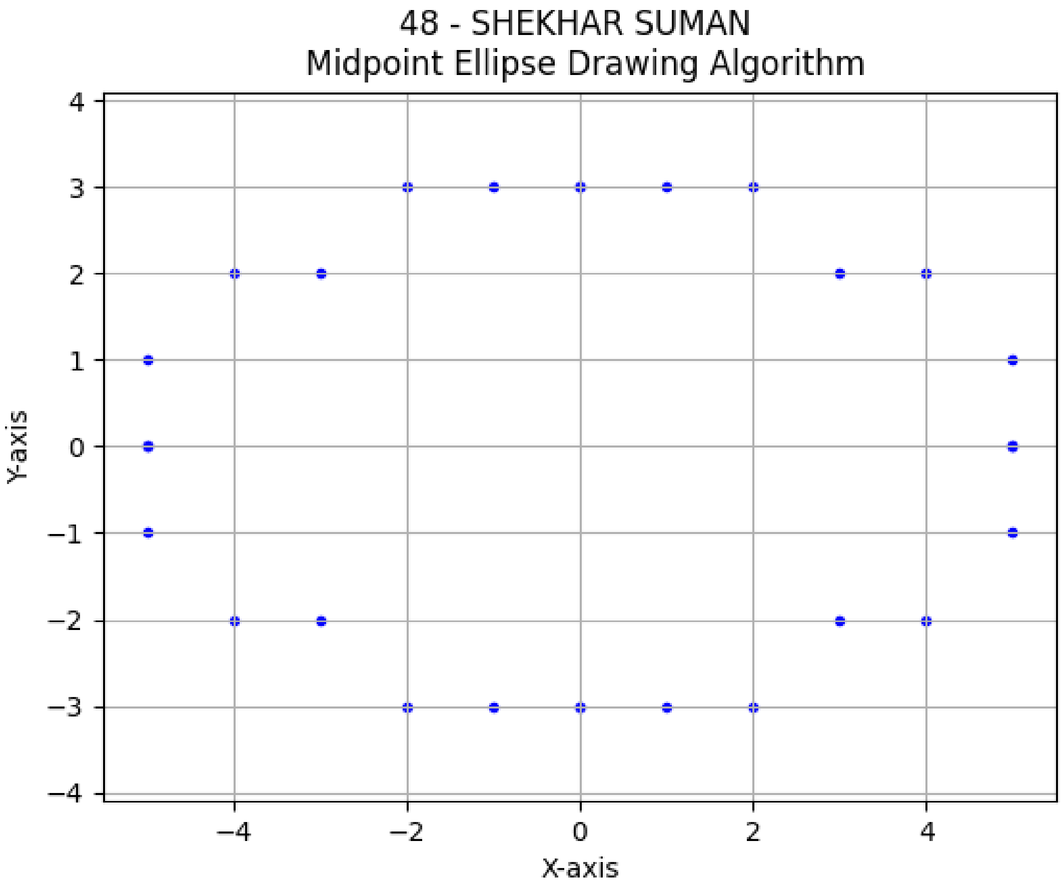
PRACTICAL: 5-A

AIM: Write a program to developed **Midpoint Ellipse Drawing Algorithm**.

CODE

```
import matplotlib.pyplot as plt
def midpoint_ellipse(a, b):
    x, y = 0, b
    a_sqr = a**2
    b_sqr = b**2
    d1 = (b_sqr - (a_sqr * b) + 0.25 * a_sqr)
    dx = 2 * b_sqr * x
    dy = 2 * a_sqr * y
    x_points = []
    y_points = []
    plot_ellipse_points(x, y, x_points, y_points)
    while dx < dy:
        x += 1
        dx += 2 * b_sqr
        if d1 < 0:
            d1 += dx + b_sqr
        else:
            y -= 1
            dy -= 2 * a_sqr
            d1 += dx - dy + b_sqr
        plot_ellipse_points(x, y, x_points, y_points)
    d2 = b_sqr * (x + 0.5)**2 + a_sqr * (y - 1)**2 - a_sqr * b_sqr
    while y > 0:
        y -= 1
        dy -= 2 * a_sqr
        if d2 > 0:
            d2 += a_sqr - dy
        else:
            x += 1
            dx += 2 * b_sqr
            d2 += a_sqr + dx - dy
        plot_ellipse_points(x, y, x_points, y_points)
    return x_points, y_points
def plot_ellipse_points(x, y, x_points, y_points):
    x_points.extend([x, -x, x, -x])
    y_points.extend([y, y, -y, -y])
a, b = 5, 3
x_points, y_points = midpoint_ellipse(a, b)
plt.scatter(x_points, y_points, marker='.', color='b')
plt.title("48 - SHEKHAR SUMAN \n Midpoint Ellipse Drawing Algorithm")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.axis('equal')
plt.grid(True)
plt.show()
```

OUTPUT


48 - SHEKHAR SUMAN
Midpoint Ellipse Drawing Algorithm
4
3
2
1
0
−1
−2
−3
−4
Y-axis
−4
−2
0
2
4
X-axis

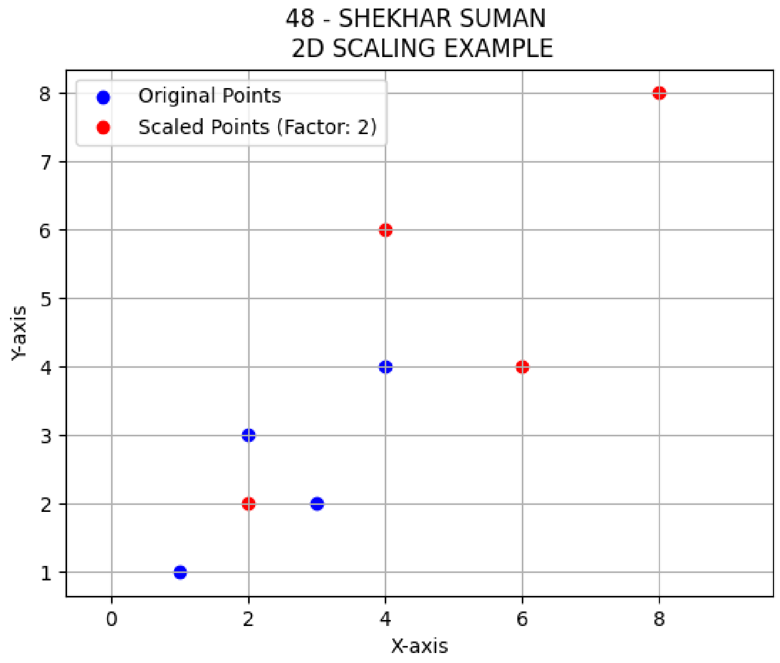
PRACTICAL: 6-A

AIM: Write a program to developed **2D SCALING EXAMPLE**.

CODE

```
import matplotlib.pyplot as plt
def scale_2d(points, scale_factor):
    scaled_points = []
    for point in points:
        x, y = point
        scaled_x = x * scale_factor
        scaled_y = y * scale_factor
        scaled_points.append((scaled_x, scaled_y))
    return scaled_points
def plot_points(points, color='b', label=None):
    x, y = zip(*points)
    plt.scatter(x, y, color=color, label=label)
original_points = [(1, 1), (2, 3), (3, 2), (4, 4)]
scale_factor = 2
scaled_points = scale_2d(original_points, scale_factor)
plot_points(original_points, color='b', label='Original Points')
plot_points(scaled_points, color='r', label=f'Scaled Points (Factor: {scale_factor})')
plt.axis('equal')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('48 - SHEKHAR SUMAN \n 2D SCALING EXAMPLE')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT



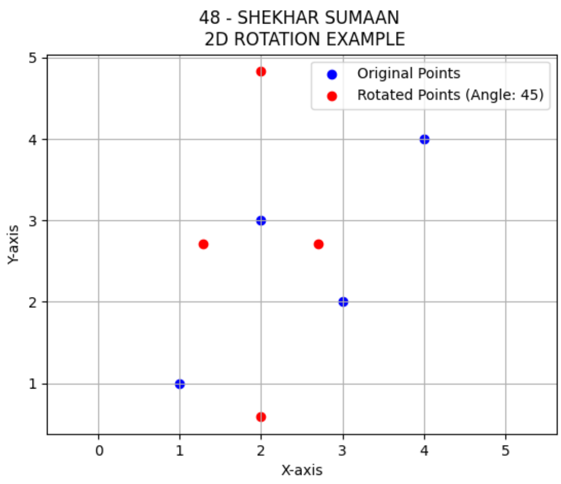
PRACTICAL: 7-A

AIM: Write a program to developed **2D ROTATION EXAMPLE**.

CODE

```
import matplotlib.pyplot as plt
import numpy as np
def rotate_2d(points, angle_degrees, rotation_center=(0, 0)):
    angle_radians = np.radians(angle_degrees)
    cos_theta = np.cos(angle_radians)
    sin_theta = np.sin(angle_radians)
    rotated_points = []
    for point in points:
        x, y = point
        x_centered = x - rotation_center[0]
        y_centered = y - rotation_center[1]
        rotated_x = x_centered * cos_theta - y_centered * sin_theta
        rotated_y = x_centered * sin_theta + y_centered * cos_theta
        rotated_x += rotation_center[0]
        rotated_y += rotation_center[1]
        rotated_points.append((rotated_x, rotated_y))
    return rotated_points
def plot_points(points, color='b', label=None):
    x, y = zip(*points)
    plt.scatter(x, y, color=color, label=label)
original_points = [(1, 1), (2, 3), (3, 2), (4, 4)]
rotation_angle = 45
rotation_center = (2, 2)
rotated_points = rotate_2d(original_points, rotation_angle, rotation_center)
plot_points(original_points, color='b', label='Original Points')
plot_points(rotated_points, color='r', label=f'Rotated Points (Angle: {rotation_angle})')
plt.axis('equal')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('48 - SHEKHAR SUMAAN \n 2D ROTATION EXAMPLE')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT


48 - SHEKHAR SUMAAN
2D ROTATION EXAMPLE
● Original Points
● Rotated Points (Angle: 45)
Y-axis
X-axis