CSE 352 - Assignment 4

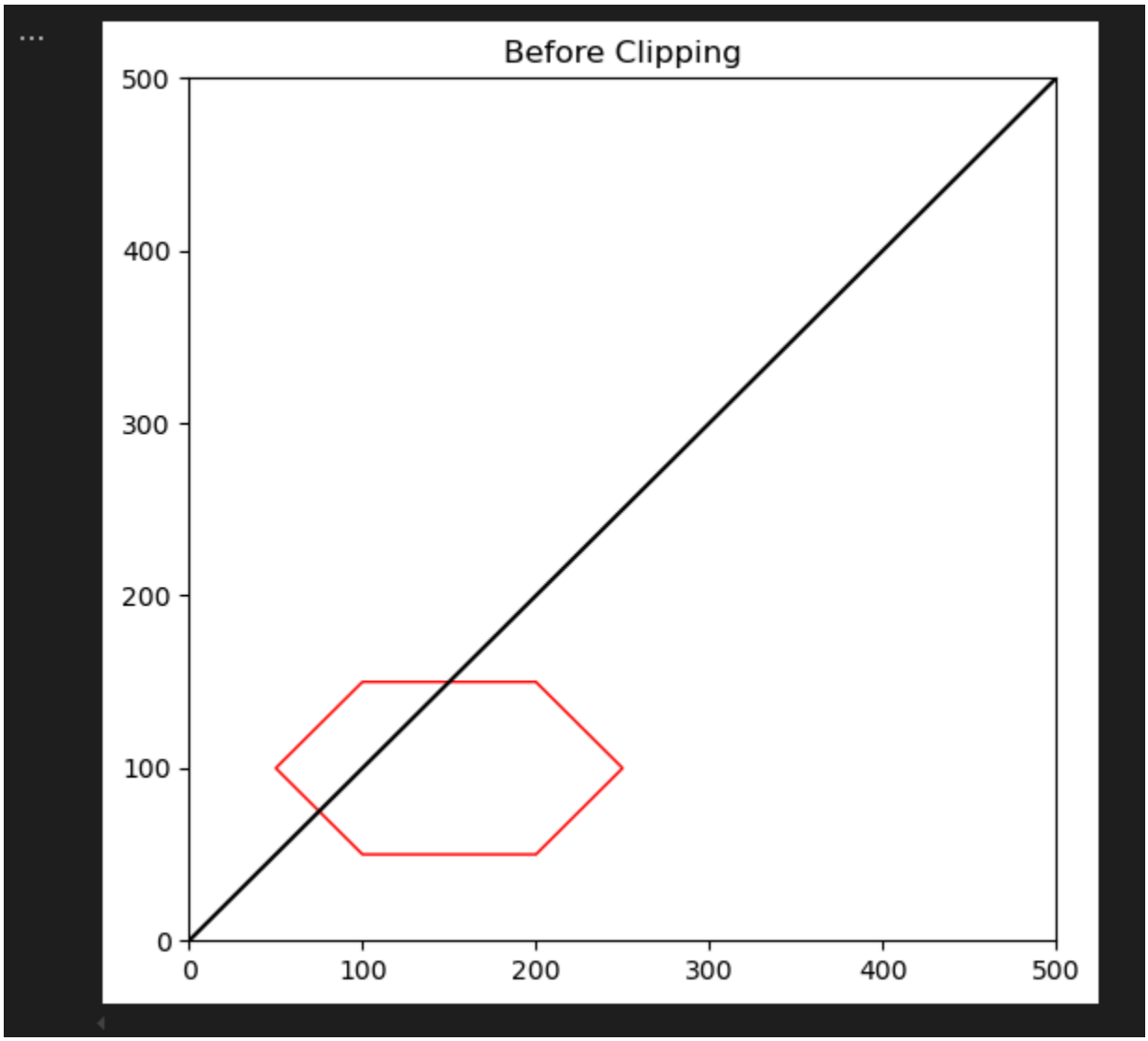
Arnav Jain - 220002018

Question 1

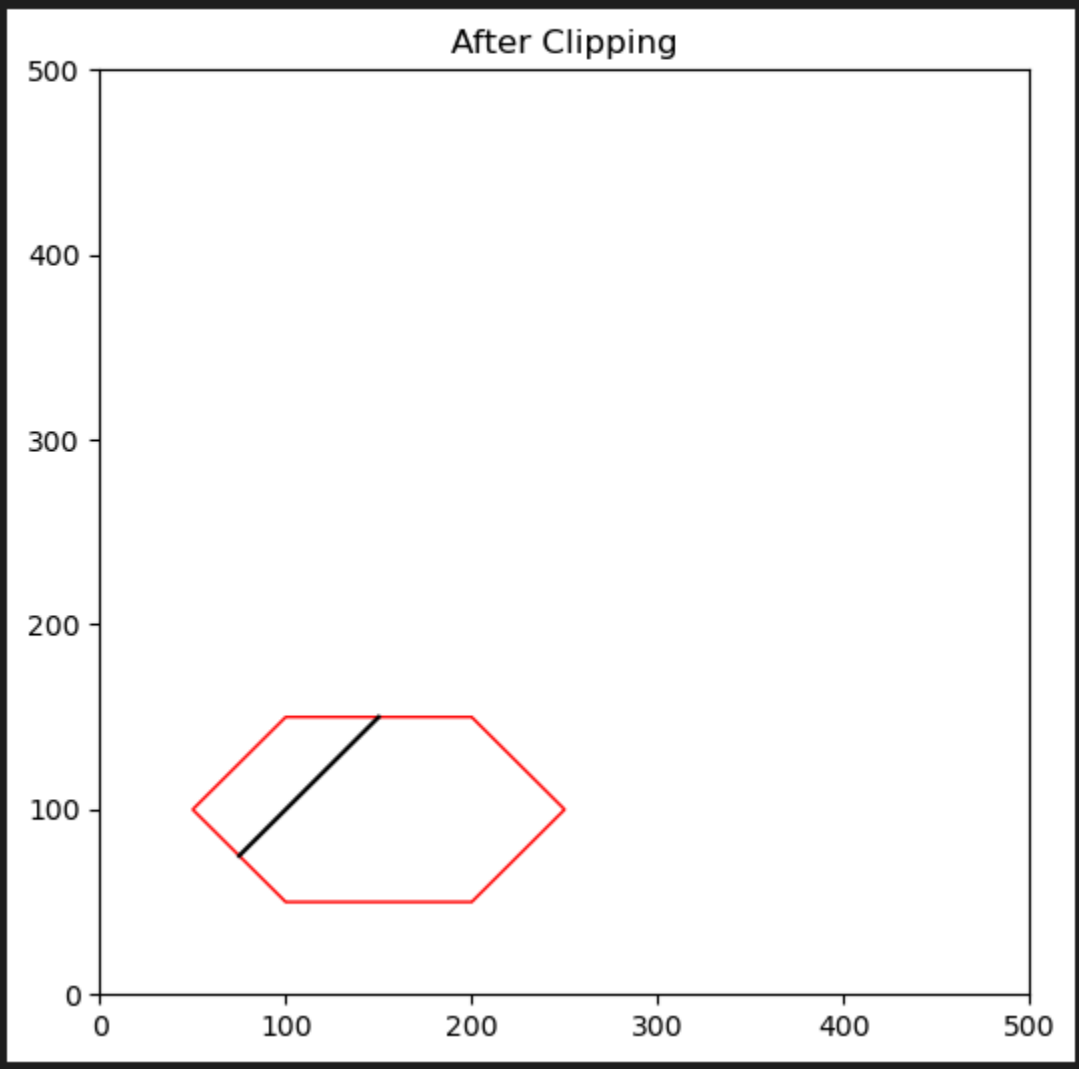
```python
import matplotlib.pyplot as plt
import numpy as np

def drawline(p0, p1):
    plt.plot([p0[0], p1[0]], [p0[1], p1[1]], 'k-')

def drawPolygon(vertices):
    vertices.append(vertices[0])
    xs, ys = zip(*vertices)
    plt.fill(xs, ys, edgecolor='r', fill=False)

def dot(p0, p1):
    return p0[0] * p1[0] + p0[1] * p1[1]

def max(t):
    return np.max(t)

def min(t):
    return np.min(t)

# Cyrus Beck function
def CyrusBeck(vertices, line):
    n = len(vertices)
    P1_P0 = (line[1][0] - line[0][0], line[1][1] - line[0][1])
    normal = [(vertices[i][1] - vertices[(i + 1) % n][1], vertices[(i + 1) % n][0] - vertices[i][0]) for i in range(n)]
    P0_PEi = [(vertices[i][0] - line[0][0], vertices[i][1] - line[0][1]) for i in range(n)]
    numerator = [dot(normal[i], P0_PEi[i]) for i in range(n)]
    denominator = [dot(normal[i], P1_P0) for i in range(n)]
    t = [numerator[i] / denominator[i] if denominator[i] != 0 else 0 for i in range(n)]
    tE = [t[i] for i in range(n) if denominator[i] > 0]
    tL = [t[i] for i in range(n) if denominator[i] < 0]
    tE.append(0)
    tL.append(1)
    print(f"Parametric Values entering the Polygon: {tE}")
    print(f"Parametric Values leaving the Polygon: {tL}")
    temp = [max(tE), min(tL)]
    if temp[0] > temp[1]:
        return None # Trivial reject

    # New Line Co ordinates
    newPair = [(line[0][0] + P1_P0[0] * temp[0], line[0][1] + P1_P0[1] * temp[0]),
               (line[0][0] + P1_P0[0] * temp[1], line[0][1] + P1_P0[1] * temp[1])]
    print(f"New Coordinates: {newPair}")
    return newPair
```

```
    def plotting(vertices , line):
        plt.figure(figsize=(6, 6))
        plt.title('Before Clipping')
        drawPolygon(vertices)
        drawline(line[0], line[1])
        plt.xlim(0, 500)
        plt.ylim(0, 500)
        plt.show()

        newPair = CyrusBeck(vertices, line)
        if newPair is not None:
            plt.figure(figsize=(6, 6))
            plt.title('After Clipping')
            drawPolygon(vertices)
            drawline(newPair[0], newPair[1])
            plt.xlim(0, 500)
            plt.ylim(0, 500)
            plt.show()
```

```
    vertices = [(200, 50), (250, 100), (200, 150), (100, 150), (50, 100), (100, 50)]
    line = [(0, 0), (500, 500)]
    plotting(vertices , line)
```


Before Clipping
500
400
300
200
100
0
0
100
200
300
400
500

```
Parametric Values entering the Polygon: [0.15, 0.1, 0]
Parametric Values leaving the Polygon: [0.35, 0.3, 1]
New Coordinates: [(75.0, 75.0), (150.0, 150.0)]
```

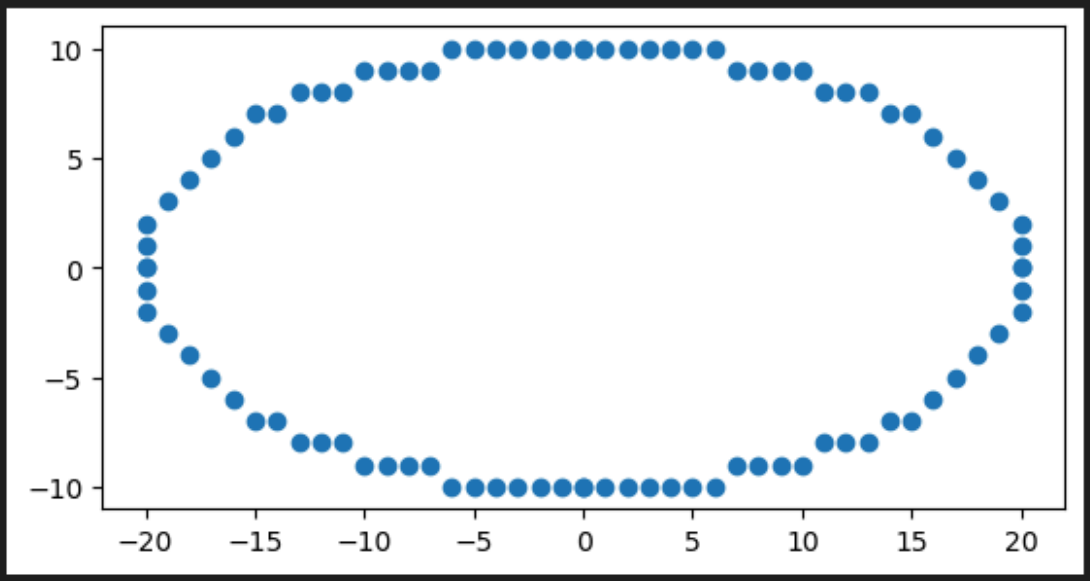


Question 2

```
[3]
import matplotlib.pyplot as plt

def midptellipse(rx, ry, xc, yc):
    x = 0
    y = ry
    d1 = ((ry * ry) - (rx * rx * ry) + (0.25 * rx * rx))
    dx = 2 * ry * ry * x
    dy = 2 * rx * rx * y

    x_points = []
    y_points = []

    while (dx < dy):
        x_points.extend([x + xc, -x + xc, x + xc, -x + xc])
        y_points.extend([y + yc, y + yc, -y + yc, -y + yc])

        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)

    d2 = (((ry * ry) * ((x + 0.5) * (x + 0.5))) +
          ((rx * rx) * ((y - 1) * (y - 1))) -
          (rx * rx * ry * ry))

    while (y >= 0):
        x_points.extend([x + xc, -x + xc, x + xc, -x + xc])
        y_points.extend([y + yc, y + yc, -y + yc, -y + yc])

        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.scatter(x_points, y_points)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.show()
```

```
midptellipse(20, 10, 0, 0)
```

[6]



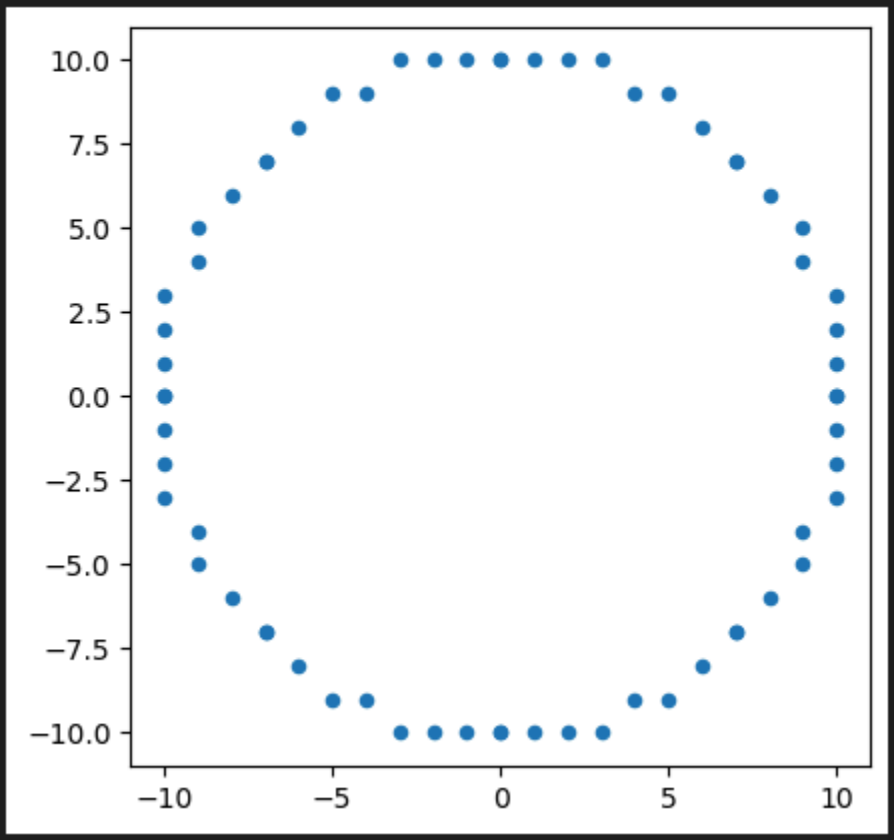
Question 3

```
import matplotlib.pyplot as plt
def draw_circle(xc, yc, r , print_points = True):
    x, y = 0, r
    p = 1 - r
    points = []

    def plot_circle_points(xc, yc, x, y):
        points.extend([(xc + x, yc + y), (xc - x, yc + y),
                       (xc + x, yc - y), (xc - x, yc - y),
                       (xc + y, yc + x), (xc - y, yc + x),
                       (xc + y, yc - x), (xc - y, yc - x)])

    plot_circle_points(xc, yc, x, y)

    while x < y:
        if print_points:
            print(points)
        x = x + 1
        if p < 0:
            p += 2*x + 1
        else:
            y -= 1
            p += 2*x - 2*y + 1

        plot_circle_points(xc, yc, x, y)

    if print_points:
        print(points)

    x_cordinate, y_cordinate = zip(*points)
    plt.scatter(x_cordinate, y_cordinate, s=20)
    plt.gca().set_aspect('equal')
    plt.show()
```

✓ 0.0s

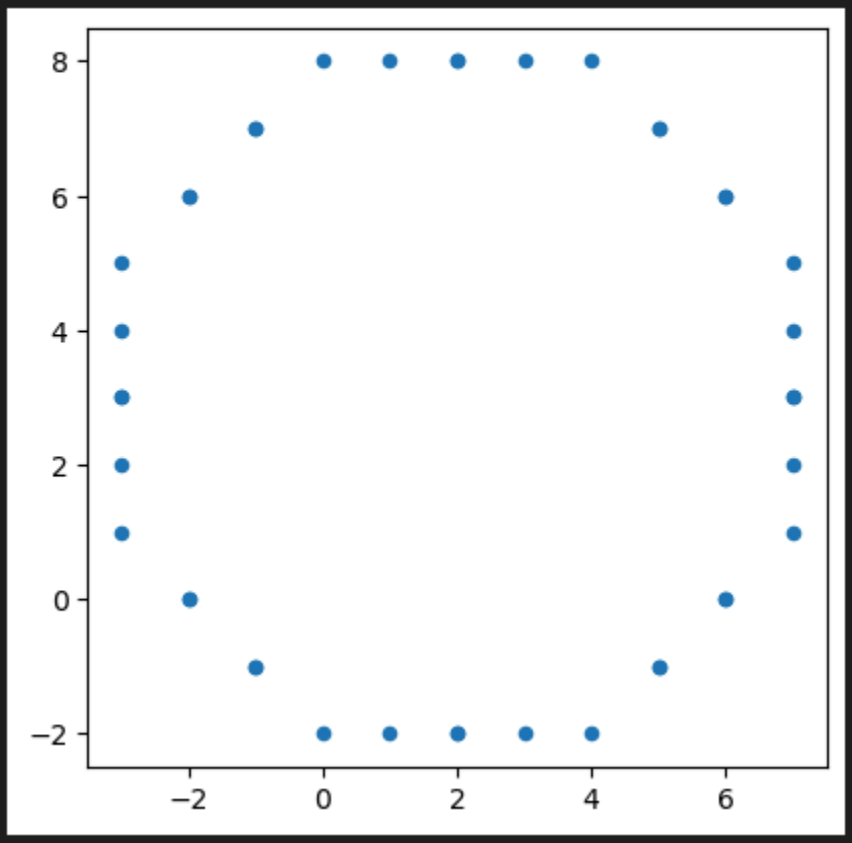
```
draw_circle(0, 0, 10 , False)
```

✓ 0.1s



```
draw_circle(2, 3, 5 , False)
```

✓ 0.1s



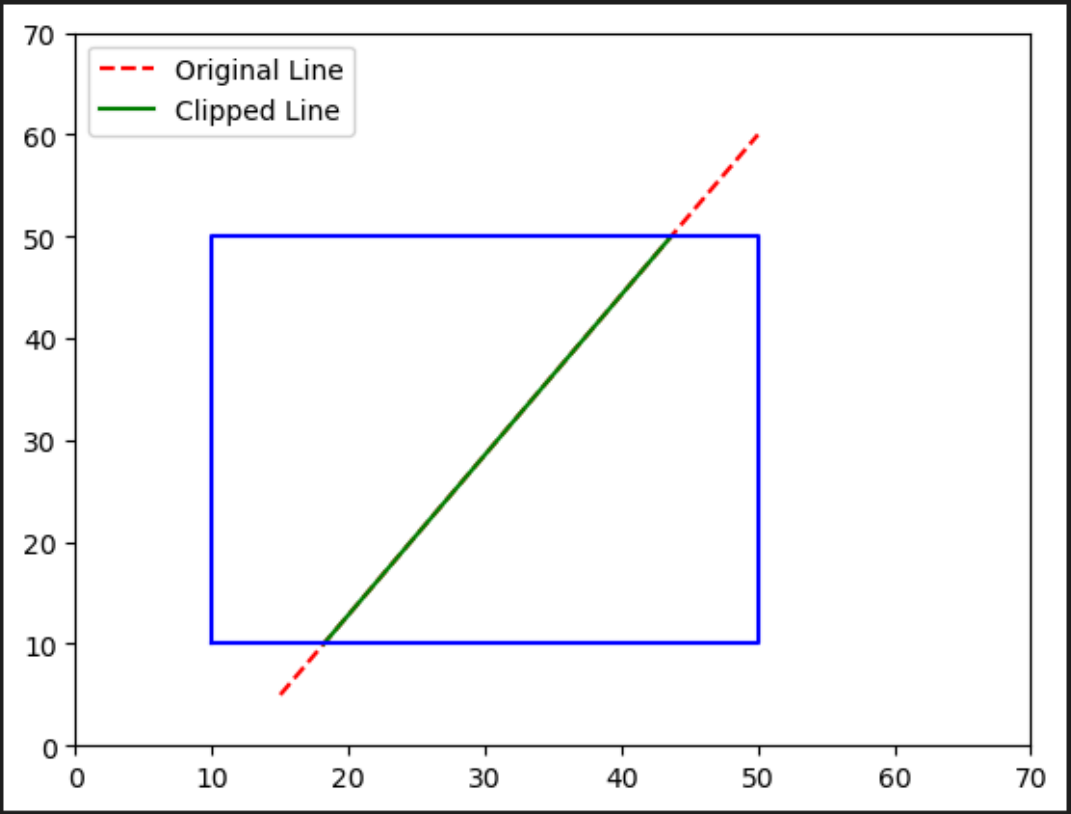
Question 4

```python
import matplotlib.pyplot as plt

INSIDE = 0  # 0000
LEFT = 1    # 0001
RIGHT = 2   # 0010
BOTTOM = 4  # 0100
TOP = 8     # 1000

def compute_code(x, y):
    code = INSIDE
    if x < xmin:
        code = code | LEFT
    if x > xmax:
        code = code | RIGHT
    if y < ymin:
        code = code | BOTTOM
    if y > ymax:
        code = code | TOP
    return code
```

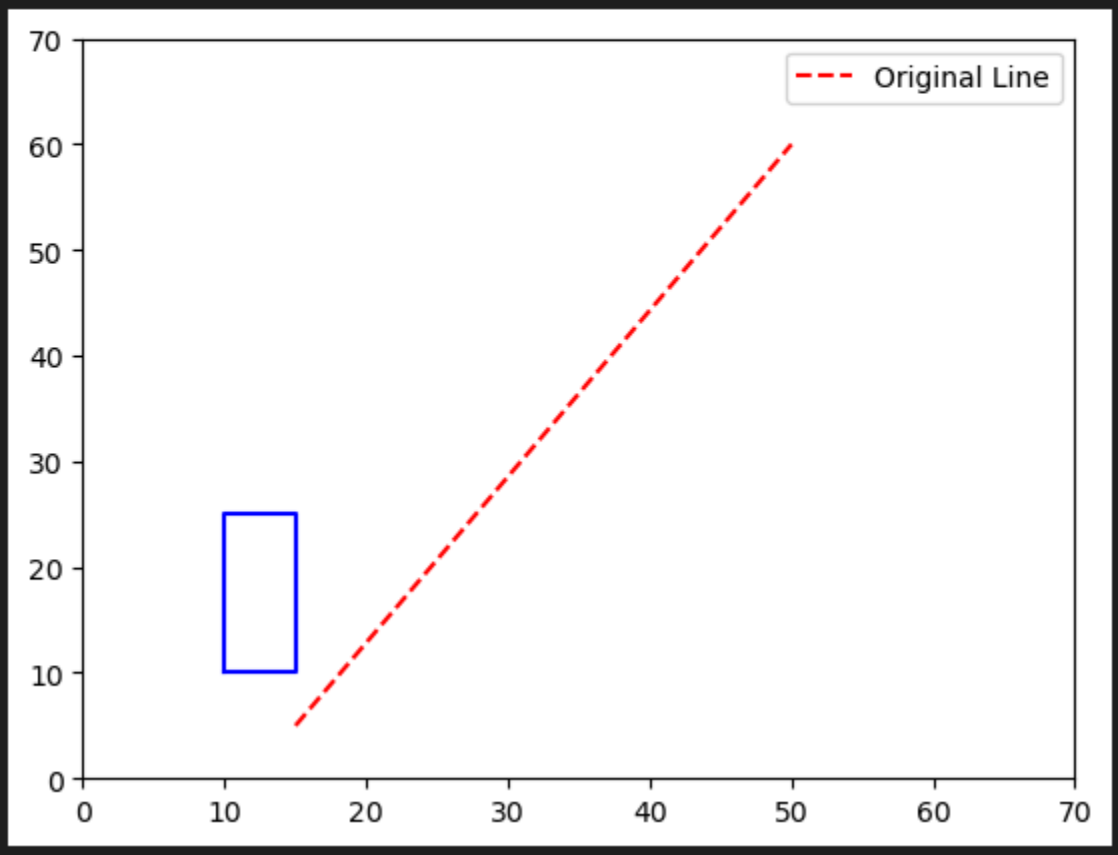
```
def cohen_sutherland_clip(x1, y1, x2, y2 , xmin, ymin, xmax, ymax ):
    code1 = compute_code(x1, y1)
    code2 = compute_code(x2, y2)
    accept = False

    while True:

        if code1 == 0 and code2 == 0:
            accept = True
            break

        # Trivially reject
        elif code1 & code2:
            break

        else:
            if code1:
                code_out = code1
            else:
                code_out = code2

            if code_out & TOP:
                x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1)
                y = ymax
            elif code_out & BOTTOM:
                x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1)
                y = ymin
            elif code_out & RIGHT:
                y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1)
                x = xmax
            elif code_out & LEFT:
                y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1)
                x = xmin

            if code_out == code1:
                x1, y1 = x, y
                code1 = compute_code(x1, y1)
            else:
                x2, y2 = x, y
                code2 = compute_code(x2, y2)

    if accept:
        return (x1, y1, x2, y2)
    else:
        return None
```

```python
def plot_line(x1 ,y1, x2, y2 , xmin, ymin, xmax, ymax ):
    clipped_line = cohen_sutherland_clip(x1, y1, x2, y2, xmin, ymin, xmax, ymax )
    fig, ax = plt.subplots()
    ax.set_xlim(0, 70)
    ax.set_ylim(0, 70)
    ax.plot([x1, x2], [y1, y2], 'r--', label="Original Line")

    if clipped_line:
        x1_clip, y1_clip, x2_clip, y2_clip = clipped_line
        ax.plot([x1_clip, x2_clip], [y1_clip, y2_clip], 'g-', label="Clipped Line")

    ax.plot([xmin, xmax, xmax, xmin, xmin], [ymin, ymin, ymax, ymax, ymin], 'b-')\

    ax.legend()
    plt.show()
```

```python
x1, y1, x2, y2 = 15, 5, 50, 60
xmin, ymin, xmax, ymax = 10, 10, 50, 50
plot_line(x1, y1, x2, y2 , xmin, ymin, xmax, ymax)
```



```
x1, y1, x2, y2 = 15, 5, 50, 60
xmin, ymin, xmax, ymax = 10, 10, 15, 25
plot_line(x1, y1, x2, y2 , xmin, ymin, xmax, ymax)
```



For code , refer GitHub

https://github.com/arnavjain2710/Computer-Graphics-Lab/tree/main/LAB%204