**EXPERIMENT 3: *LINE CLIPPING ALGORITHMS(Cohen Sutherland)***

**Name: Samyak Jain**

**Roll: LIT2018056**

**CODE:-**

from turtle import \*

INSIDE = 0

LEFT = 1

RIGHT = 2

BOTTOM = 4

TOP = 8

print('Input x\_min, y\_min, x\_max, y\_max in each line')

x\_min = int(input())

y\_min = int(input())

x\_max = int(input())

y\_max = int(input())

#printing rectangle

pencolor("black")

penup()

goto(x\_min,y\_min)

pendown()

goto(x\_min,y\_max)

goto(x\_max,y\_max)

penup()

goto(x\_min,y\_min)

pendown()

goto(x\_max,y\_min)

goto(x\_max,y\_max)

penup()

def get\_bits\_value(x, y):

    temp = INSIDE

    if int(x) < int(x\_min):  # left

        temp  |= LEFT

    elif int(x) > int(x\_max): #  right

        temp |= RIGHT

    if int(y) < int(y\_min):  # below

        temp |= BOTTOM

    elif int(y) > int(y\_max): # above

        temp |= TOP

    #print(temp)

    return temp

print('Now print the no. of lines')

#input no. of lines

n = int(input())

for number in range(n):

    print('Now print value of  x1,y1,x2,y2 in each line')

    #input coordinates

    x1 = int(input())

    y1 = int(input())

    x2 = int(input())

    y2 = int(input())

    x1\_temp = x1

    y1\_temp = y1

    x2\_temp = x2

    y2\_temp = y2

    #to find the position of point

    temp1 = get\_bits\_value(x1, y1)

    temp2 = get\_bits\_value(x2, y2)

    indicate = False

    #print(temp1,temp2)

    while True:

        #inside totally

        if temp1 == 0 and temp2 == 0:

            indicate = True

            #print('inside totally')

            break

        #outside totally

        elif ((temp1 & temp2) != 0):

            #print('outside totally')

            break

        #partial case

        else:

            x = 1.0

            y = 1.0

            #choosing one of the point

            if temp1 != 0:

                code\_out = temp1

            else:

                code\_out = temp2

            #now check the position & find respective coordinates

            if code\_out & TOP:

                x = x1 + (x2 - x1) \*(y\_max - y1) / (y2 - y1)

                y = y\_max

            elif code\_out & BOTTOM:

                x = x1 + (x2 - x1) \*(y\_min - y1) / (y2 - y1)

                y = y\_min

            elif code\_out & RIGHT:

                y = y1 + (y2 - y1) \*(x\_max - x1) / (x2 - x1)

                x = x\_max

            elif code\_out & LEFT:

                y = y1 + (y2 - y1) \*(x\_min - x1) / (x2 - x1)

                x = x\_min

            if code\_out == temp1:

                x1 = x

                y1 = y

                temp1 = get\_bits\_value(x1, y1)

            else:

                x2 = x

                y2 = y

                temp2 = get\_bits\_value(x2, y2)

    if indicate:

        print ("It lies between ", x1, " , " , y1 ," to " ,x2 ," , " ,y2)

        pencolor('red')

        penup()

        goto(x1\_temp,y1\_temp)

        pendown()

        goto(x1,y1)

        pencolor('green')

        goto(x2,y2)

        pencolor('red')

        goto(x2\_temp,y2\_temp)

    else:

        print("It is outside of this rectangle completely")

        pencolor('red')

        penup()

        goto(x1\_temp,y1\_temp)

        pendown()

        goto(x2\_temp,y2\_temp)

exitonclick()

***Input:-***

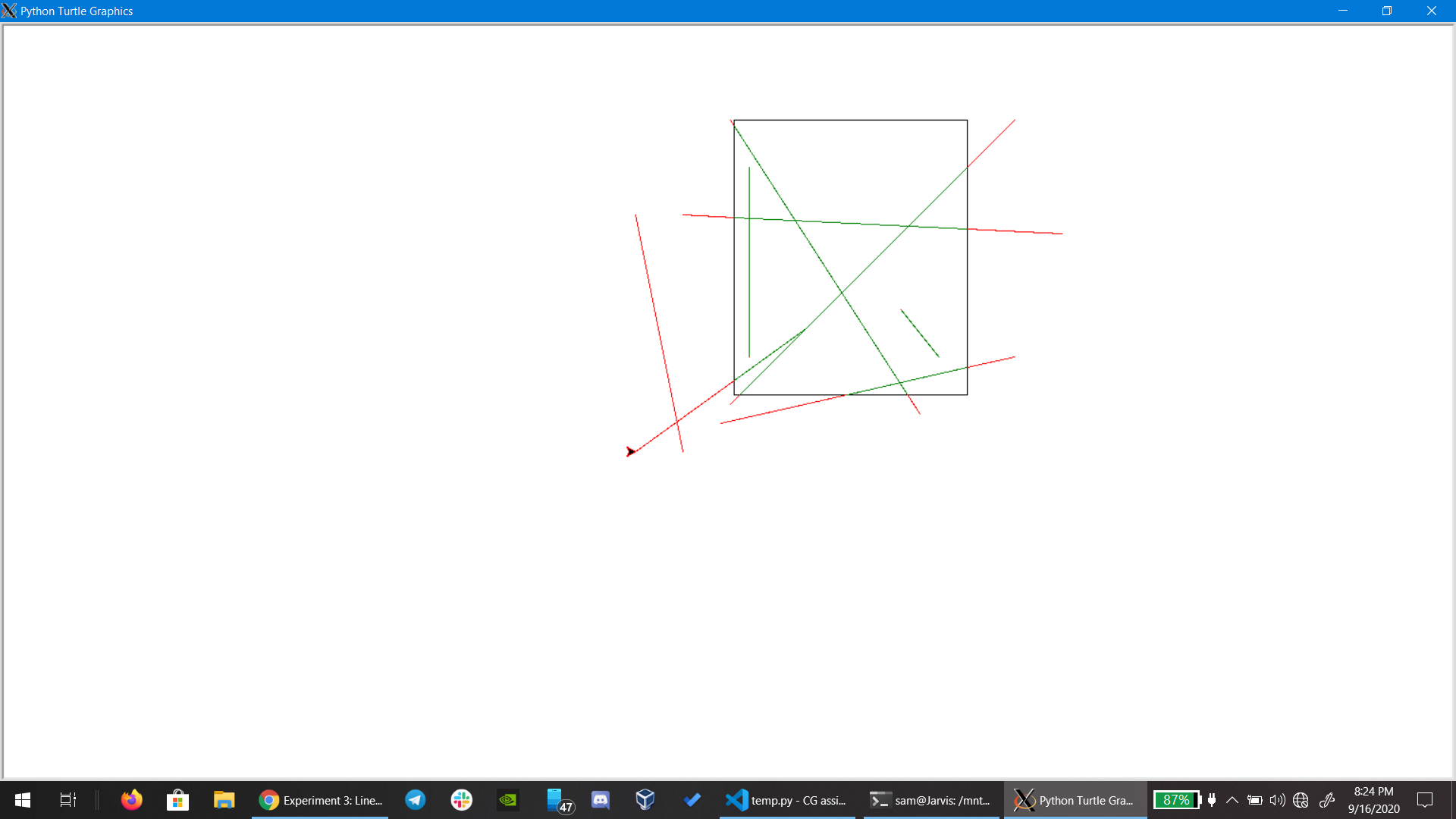
Dimension of Rectangle (Left and Bottom most point = 4,10 & Right and Topmost point = 250,300)

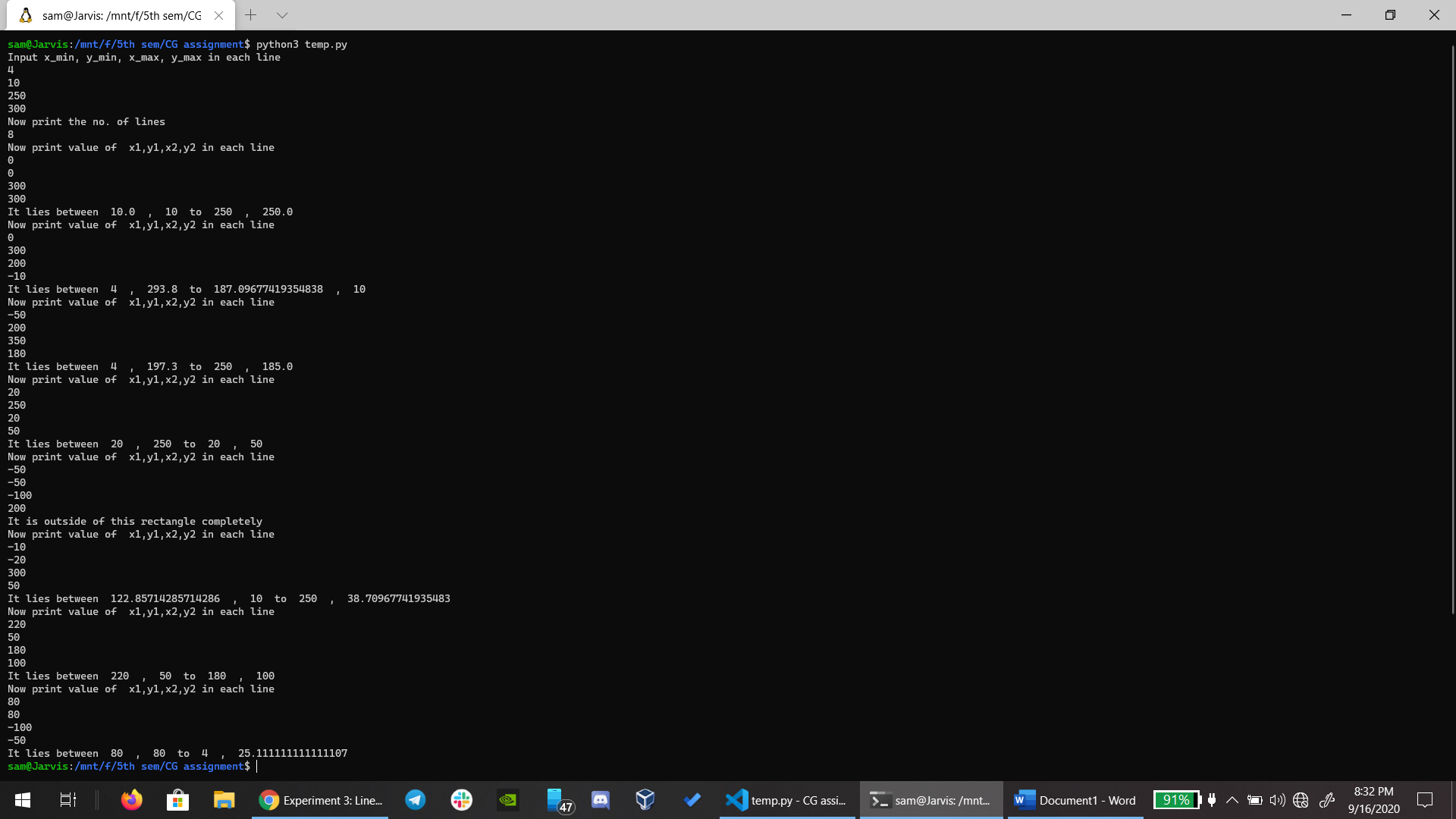
8 lines were drawn:

Their end points are:-

1. (0,0),(300,30)
2. (0,300),(200,-10)
3. (-50,200),(350,180)
4. (20,250),(20,50)
5. (-50,-50),(-100,200)
6. (-10,-20),(300,50)
7. (220,50),(180,100)
8. (80,80),(-100,-50)

**Output:-**





**Note:- 1.It also shows the starting and end point in rectangle for a line.**

**2.Also I took black color for rectangle, Green for part of line inside the rectangle and Red for part of line outside the rectangle for better visualization.**