Netaji Subhas University of Technology

Practical File
INITC08 Computer Graphics

Name: Jashanpreet Singh Rangi
Roll No. : 2021UIN3332
Branch: ITNS(Information Technology Network Security)

Practical-1

Aim: Write a program to draw a line using DDA Algorithm

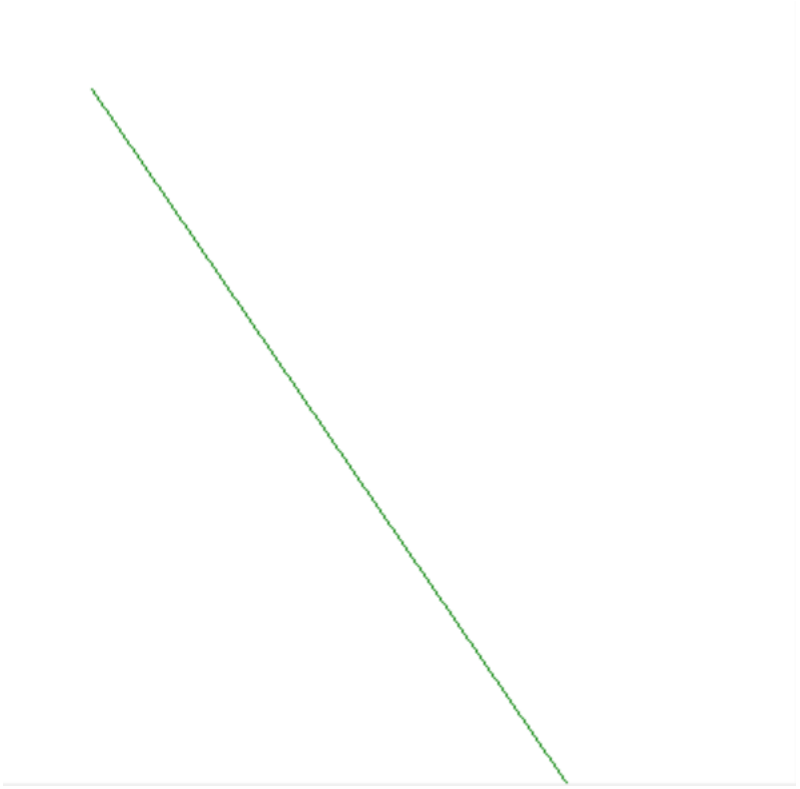
Software Used: Pycharm
Language Used: Python

Code:

```
from tkinter import *

root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=400,background="white")
canvas.place(x=100,y=100)

x0=int(input())
y0=int(input())
x1=int(input())
y1=int(input())

def putpixel(x,y):
    canvas.create_line(x,y,x+1,y+1,fill="green")

dx=abs(x1-x0)
dy=abs(y1-y0)
m=dx if dx>dy else dy

xadd=dx/m
yadd=dy/m
X=x0
Y=y0
for i in range (0,m):
    putpixel(X,Y)
    X=X+xadd
    Y=Y+yadd

root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
>>>
>>>
*** Remote Interpreter Reinitialized ***
50
55
310
435
#Jashan Rangi#
```



Practical-2

Aim: Write a program to draw a line using bresenham's algorithm

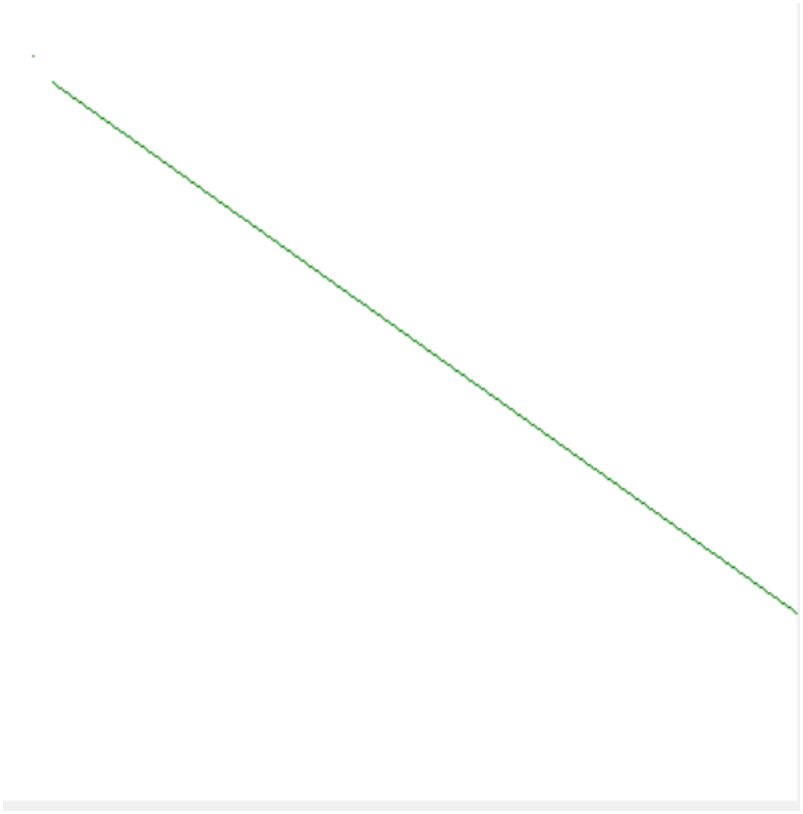
Software Used: Pycharm
Language Used: Python

Code:

```
from tkinter import *
root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=400,background="white")
canvas.place(x=100,y=100)

x0=int(input())
y0=int(input())
x1=int(input())
y1=int(input())

Y=y0
X=x0
dx=x1-x0
dy=y1-y0
p=2*dx-dy

def putpixel(x,y):
    canvas.create_line(x,y,x+1,y+1,fill="green")

while X<x1:
    putpixel(X,Y)
    X=X+1
    if p<0:
        p=p+2*dy
    else:
        p=p+2*dy-2*dx
        Y=Y+1

putpixel(20,30)

root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
>>>
*** Remote Interpreter Reinitialized ***
30
43
450
342
#Jashan Rangi#
```



Practical-3

Aim: Write a program to draw a circle using the Bresenhem's algorithm

Software Used: Pycharm
Language Used: Python

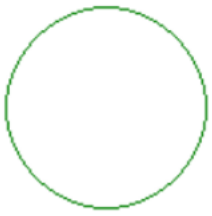
Code:

```
from ctypes import PYFUNCTYPE
from tkinter import *

root=Tk(4
root.geometry("600x600")
canvas=Canvas(root,width=400,height=400,background="white")
canvas.place(x=100,y=100)

r=int(input())
xc=int(input())
yc=int(input())

def makecircle(x0,y0,r):
    x=0
    y=r
    putpixel(x0+x,y0+y)
    putpixel(x0+x,y0-y)
    putpixel(x0-x,y0+y)
    putpixel(x0-x,y0-y)
    putpixel(x0+y,y0+x)
    putpixel(x0+y,y0-x)
    putpixel(x0-y,y0+x)
    putpixel(x0-y,y0-x)
    d=3-(2*r)
    while x<=y:
        if d<0:
            d=d+(4*x)+6
        else:
            d=d+(4*(x-y))+10
            y=y-1
        putpixel(x0+x,y0+y)
        putpixel(x0+x,y0-y)
```

```
        putpixel(x0-x,y0+y)
        putpixel(x0-x,y0-y)
        putpixel(x0+y,y0+x)
        putpixel(x0+y,y0-x)
        putpixel(x0-y,y0+x)
        putpixel(x0-y,y0-x)
        x=x+1

def putpixel(x,y):
    canvas.create_line(x,y,x+1,y,fill="green")

makecircle(xc,yc,r)

root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
50
120
230
>>>
#Jashan Rangi#
```



Practical-4

Aim: Write a program to draw an ellipse using midpoint algorithm

Software Used: Pycharm
Language Used: Python

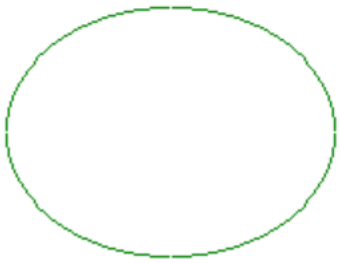
Code:

```
from ctypes import PYFUNCTYPE
from tkinter import *

root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=400,background="white")
canvas.place(x=100,y=100)

rx=int(input())
ry=int(input())
xc=int(input())
yc=int(input())

def putcoords(x0,y0,x,y):
    putpixel(x0+x,y0+y)
    putpixel(x0+x,y0-y)
    putpixel(x0-x,y0+y)
    putpixel(x0-x,y0-y)

def putpixel(x,y):
    canvas.create_line(x,y,x+1,y,fill="green")



def makeellipse(x0,y0,rx,ry):
    x=0
    y=ry
    d=(ry*ry)-((rx*rx)*ry)+((1/4)*(rx*rx))
    star=(ry*ry*x)
    end=(rx*rx*y)
```

```
    while star<end:
        x=x+1
        if d<0:
            d=d+(2*ry*ry*x)+(ry*ry)
        else:
            y=y-1
            d=d+(2*ry*ry*x)+(ry*ry)-(2*rx*rx*y)
        putcoords(x0,y0,x,y)
        star=ry*ry*x
        end=rx*rx*y

    while y>0:
        putcoords(xc,yc,x,y)
        y=y-1
        if d>0:
            d=d-(2*rx*rx*y)+(rx*rx)
        else:
            x=x+1
            d=d+(2*ry*ry*x)+(rx*rx)-(2*rx*rx*y)
            putcoords(xc,yc,x,y)

makeellipse(xc,yc,rx,ry)

root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
>>>
*** Remote Interpreter Reinitialized ***
83
62
104
92
#Jashan Rangi#
```



Practical-5

Aim: Write a program to implement Cohen Sutherland line clipping algorithm

Software Used: Pycharm
Language Used: Python

Code:

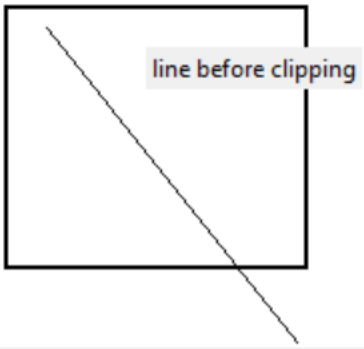
```
from tkinter import *

inside=0
left=1
right=2
bottom=4
top=8

x1=50
y1=40
x2=176
y2=198

x_min=30
y_min=30
x_max=180
y_max=160

root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=200,background="white")
canvas.place(x=100,y=100)
canvas1=Canvas(root,width=400,height=200,background="white")
canvas1.place(x=100,y=300)
canvas.create_rectangle(x_min,y_min,x_max,y_max,width=2,outlin
e="black")
label1=Label(canvas,text="line before clipping").place(x=100,y=50)
canvas.create_line(x1,y1,x2,y2,fill="black")
canvas1.create_rectangle(x_min,y_min,x_max,y_max,width=2,outlin
e="black")
label2=Label(canvas1,text="line after clipping").place(x=100,y=50)
```

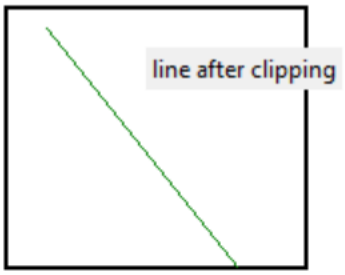
```
def checkboundary(x,y):
    check=inside
    if x<x_min:
        check=check|left
    elif x>x_max:
        check=check|right

    if y<y_min:
        check=check|bottom
    elif y>y_max:
        check=check|top

    return check

def cohen(x1,y1,x2,y2):
    check1=checkboundary(x1,y1)
    check2=checkboundary(x2,y2)

    flag=False
    while True:
        if check1==0 and check2==0:
            flag=True
            break
        elif (check1 & check2)!=0:
            break

        else:
            x=1.0
            y=1.0
            if check1!=0:
                check=check1
            else:
                check=check2

            if check & top:
                x = x1 + (x2 - x1) *(y_max - y1) / (y2 - y1)
                y = y_max

            elif check & bottom:
                x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1)
```

```
            y = y_min

        elif check & right:
            y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1)
            x = x_max

        elif check & left:
            y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1)
            x = x_min

        if check==check1:
            x1=x
            y1=y
            check1=checkboundary(x1,y1)
        else:
            x2=x
            y2=y
            check2=checkboundary(x2,y2)

    if flag:
        print("coordinates after clipping are:",x1,y1,x2,y2)
        canvas1.create_line(x1,y1,x2,y2,fill="green")
    else:
        print("line lies outside the clipping boundary")

cohen(x1, y1, x2, y2)

root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
*** Remote Interpreter Reinitialized ***
coordinates after clipping are: 50 40 145.69620253164555 160
>>>
#Jashan Rangi#
```





Practical-6

Aim: Write a program to implement Liang Barsky line clipping algorithm

Software Used: Pycharm
Language Used: Python

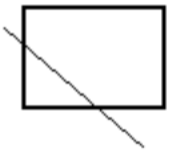
Code:

```
from tkinter import *

root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=200,background="white")
canvas.place(x=100,y=100)
canvas1=Canvas(root,width=400,height=200,background="white")
canvas1.place(x=100,y=300)

x1=10
y1=30
x2=80
y2=90

x_min=20
y_min=20
x_max=90
y_max=70

canvas.create_rectangle(x_min,y_min,x_max,y_max,width=2,outlin
e="black")
label1=Label(canvas,text="line before clipping").place(x=100,y=50)
canvas.create_line(x1,y1,x2,y2,fill="black")
canvas1.create_rectangle(x_min,y_min,x_max,y_max,width=2,outlin
e="black")
label2=Label(canvas1,text="line after clipping").place(x=100,y=50)

dx=x2-x1
dy=y2-y1

p=[-dx,dx,-dy,dy]
q=[x1-x_min,x_max-x1,y1-y_min,y_max-y1]
```

```
print(p)
print(q)

t1=0
t2=1
for i in range(0,4):
    if p[i]==0:
        print("line is parallel to one of the clipping boundaries")
    if q[i]>=0:
        if i<2:
            y1=max(y1,y_min)
            y2=min(y2,y_max)

        if i>1:
            x1=max(x1,x_min)
            x2=min(x2,x_max)

for i in range(0,4):
    r=q[i]/p[i]
    if p[i]<0:
      t1=max(t1,(r))

    else:
      t2=min(t2,(r))

if t1<t2:
    x2=x1+t2*dx
    y2=y1+t2*dy
    x1=x1+t1*dx
    y1=y1+t1*dy

print(x1,y1,x2,y2)

canvas1.create_line(x1,y1,x2,y2,fill="green")
root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
*** Remote Interpreter Reinitialized ***
[-70, 70, -60, 60]
[-10, 80, 10, 40]
30.0 38.57142857142857 66.66666666666666 70.0
#Jashan rangi#
```



line before clipping

line after clipping

Practical-7

Aim: Write a program to implement Cyrus Beck line clipping algorithm

Software Used: Pycharm
Language Used: Python

Code:

```
from tkinter import *
import numpy as np

root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=200,background="white")
canvas.place(x=100,y=100)
canvas1=Canvas(root,width=400,height=200,background="white")
canvas1.place(x=100,y=300)
x1=int(input())
y1=int(input())
x2=int(input())
y2=int(input())
label1=Label(canvas,text="line before clipping").place(x=100,y=50)
label2=Label(canvas1,text="line after clipping").place(x=100,y=50)
canvas.create_polygon(200, 50, 250, 100, 200, 150, 100, 150, 50, 100, 100,
50,outline="black")
canvas1.create_polygon(200, 50, 250, 100, 200, 150, 100, 150, 50, 100, 100,
50,outline="black")
# im will show the overlapped between lines # im1 will show the
clipped line

vertices = [[200, 50], [250, 100], [200, 150], [100, 150], [50, 100], [100, 50]]
n =6
def dot(x1, y1, x2, y2):
    return x1 * x2 + y1 * y2

def CyrusBeckLineClipping(x1, y1, x2, y2):
   normal = [[0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]]
   for i in range(0, n):
      normal[i][1] = vertices[(i + 1) % n][0] - vertices[i][0]
```

```
            normal[i][0] = vertices[i][1] - vertices[(i + 1) % n][1]

        dx = x2 - x1
        dy = y2 - y1
        dp1e = [[0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]]
        for i in range(0, n):
            dp1e[i][0] = vertices[i][0] - x1
            dp1e[i][1] = vertices[i][1] - y1
        numerator = [0, 0, 0, 0, 0, 0]
        denominator = [0, 0, 0, 0, 0, 0]

        for i in range(0, n):
            numerator[i] = dot(normal[i][0], normal[i][1], dp1e[i][0], dp1e[i][1])
            denominator[i] = dot(normal[i][0], normal[i][1], dx, dy)
        t = [0, 0, 0, 0, 0, 0]
        tE = np.array([0])
        tL = np.array([1])
        for i in range(0, n):
            t[i] = float(numerator[i]) / float(denominator[i])
            if denominator[i] > 0:
                tE = np.append(tE, t[i])
            else:
                tL = np.append(tL, t[i])

        temp0 = np.amax(tE)
        temp1 = np.amin(tL)
        if temp0 > temp1:
            return
        New_X1 = float(x1) + float(dx) * float(temp0)
        New_Y1 = float(y1) + float(dy) * float(temp0)
        New_X2 = float(x1) + float(dx) * float(temp1)
        New_Y2 = float(y1) + float(dy) * float(temp1)
        canvas1.create_line(New_X1, New_Y1, New_X2, New_Y2,fill="black")

    def clippingProcess(x1, y1, x2, y2):
        canvas.create_line(x1, y1, x2, y2,fill="black")
        CyrusBeckLineClipping(x1,y1,x2,y2)

    clippingProcess(x1, y1, x2, y2)

    root.mainloop()
```

Output:

```
*** Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
20
25
150
200
>>>
#Jashan Rangi#
```

Practical-8

Aim: Write a program to implement translation and rotation transformation on triangle.

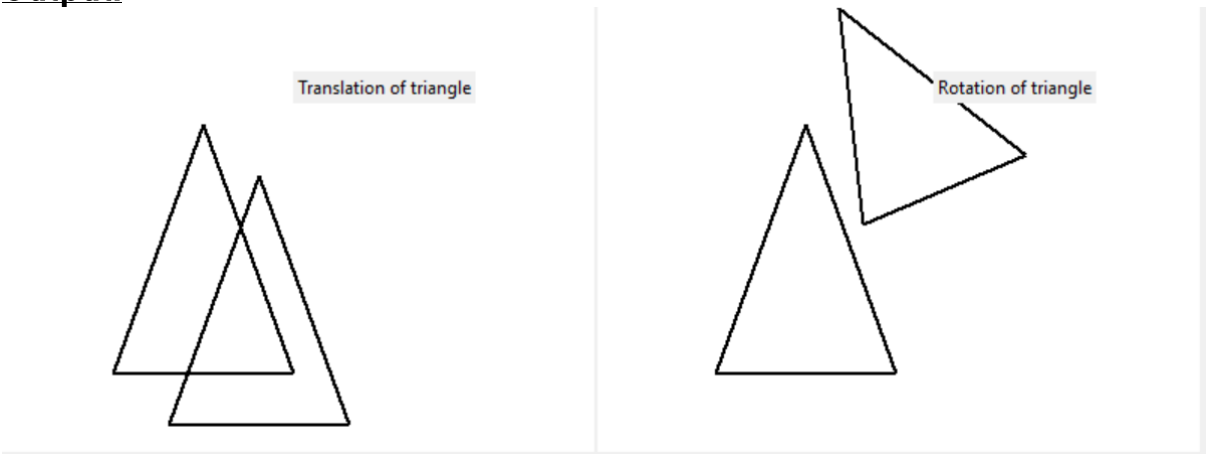
Software Used: Pycharm
Language Used: Python

Code:

```
from tkinter import *
from math import cos,sin,radians,floor

root=Tk()
root.geometry("900x400")
canvas=Canvas(root,width=400,height=300,background="white")
canvas.place(x=50,y=50)
canvas1=Canvas(root,width=400,height=300,background="white")
canvas1.place(x=450,y=50)


x1=80
y1=250
x2=140
y2=85
x3=200
y3=250

# Translation of Triangle
tx=37
ty=34

label1=Label(canvas,text="Translation of
triangle").place(x=200,y=50)
canvas.create_line(x1,y1,x2,y2,width=2,fill="black")
canvas.create_line(x1,y1,x3,y3,width=2,fill="black")
canvas.create_line(x2,y2,x3,y3,width=2,fill="black")

canvas.create_line(x1+tx,y1+ty,x2+tx,y2+ty,width=2,fill="black")
canvas.create_line(x1+tx,y1+ty,x3+tx,y3+ty,width=2,fill="black")
canvas.create_line(x2+tx,y2+ty,x3+tx,y3+ty,width=2,fill="black")
```

```
# Rotation of Triangle
rotate_angle = 25;
angle = radians(rotate_angle)
c = cos(angle)
s = sin(angle)

label2=Label(canvas1,text="Rotation of triangle").place(x=225,y=50)
canvas1.create_line(x1,y1,x2,y2,width=2,fill="black")
canvas1.create_line(x1,y1,x3,y3,width=2,fill="black")
canvas1.create_line(x2,y2,x3,y3,width=2,fill="black")
x1 = floor(x1 * c + y1 * s)
y1 = floor(-x1 * s + y1 * c)
x2 = floor(x2 * c + y2 * s)
y2 = floor(-x2 * s + y2 * c)
x3 = floor(x3 * c + y3 * s)
y3 = floor(-x3 * s + y3 * c)
canvas1.create_line(x1,y1,x2,y2,width=2,fill="black")
canvas1.create_line(x1,y1,x3,y3,width=2,fill="black")
canvas1.create_line(x2,y2,x3,y3,width=2,fill="black")

root.mainloop()
```

Output:



Practical-9

Aim: Write a program to scale a rectangle

Software Used: Pycharm
Language Used: Python

Code:

```
from tkinter import *
root=Tk()
root.geometry("600x600")
canvas=Canvas(root,width=400,height=400,background="white")
canvas.place(x=100,y=100)
x1=20
y1=20
x2=250
y2=150
sv=1.5
canvas.create_rectangle(x1,y1,x2,y2,width=2,outline="black")
label1=Label(canvas,text="Original").place(x=110,y=75)
canvas.create_rectangle(x1*sv,y1*sv,x2*sv,y2*sv,width=2,outline="
black")
label2=Label(canvas,text="Scaled Rectangle").place(x=250,y=200)

root.mainloop()
```

Output: