

## **PRACTICAL 08**

**AIM:** Write a program to implement Cohen Sutherland Line Clipping Algorithm.

**SOFTWARE REQUIRED:** TURBO C++

**PROGRAM:**

```
#include"stdio.h"

#include"conio.h"

#include"graphics.h"

void main()

{

int gd=DETECT, gm;

float i,xmax,ymax,xmin,ymin,x1,y1,x2,y2,m;

float start[4],end[4],code[4];

clrscr();

initgraph(&gd,&gm,"C:\\\\TURBOC3\\\\BGI");

printf("\\n\\tEnter the bottom-left coordinate of viewport: ");

scanf("%f %f",&xmin,&ymin);

printf("\\n\\tEnter the top-right coordinate of viewport: ");

scanf("%f %f",&xmax,&ymax);

printf("\\n\\tEnter the coordinates for starting point of line: ");

scanf("%f %f",&x1,&y1);

printf("\\n\\tEnter the coordinates for ending point of line: ");

scanf("%f %f",&x2,&y2);

for(i=0;i <4;i++)

{

start[i]=0;

end[i]=0;

}

m=(y2-y1)/(x2-x1);

if(x1 <xmin) start[0]=1;

if(x1 >xmax) start[1]=1;

if(y1 >ymax) start[2]=1;

if(y1 <ymin) start[3]=1;

if(x2 <xmin) end[0]=1;

if(x2 >xmax) end[1]=1;

if(y2 >ymax) end[2]=1;

if(y2 <ymin) end[3]=1;

for(i=0;i <4;i++)

code[i]=start[i]&&end[i];
```

```

if((code[0]==0)&&(code[1]==0)&&(code[2]==0)&&(code[3]==0))
{
if((start[0]==0)&&(start[1]==0)&&(start[2]==0)&&(start[3]==0)&&(end[0]==0)&&
(end[1]==0)&&(end[2]==0)&&(end[3]==0))
{
cleardevice();

printf("\n\t\tThe line is totally visible\n\t\t\tand not a clipping candidate");
rectangle(xmin,ymin,xmax,ymax);
line(x1,y1,x2,y2);
getch();
}
else
{
cleardevice();
printf("\n\t\tLine is partially visible");
rectangle(xmin,ymin,xmax,ymax);
line(x1,y1,x2,y2);
getch();
if((start[2]==0)&&(start[3]==1))
{
x1=x1+(ymin-y1)/m;
y1=ymin;
}
if((end[2]==0)&&(end[3]==1))
{
x2=x2+(ymin-y2)/m;
y2=ymin;
}
if((start[2]==1)&&(start[3]==0))
{
x1=x1+(ymax-y1)/m;
y1=ymax;
}
if((end[2]==1)&&(end[3]==0))
{
x2=x2+(ymax-y2)/m;
y2=ymax;
}
if((start[1]==0)&&(start[0]==1))
{
y1=y1+m*(xmin-x1);
x1=xmin;
}
}

```

```

if((end[1]==0)&&(end[0]==1))
{
y2=y2+m*(xmin-x2);
x2=xmin;
}
if((start[1]==1)&&(start[0]==0))
{
y1=y1+m*(xmax-x1);
x1=xmax;
}
if((end[1]==1)&&(end[0]==0))
{
y2=y2+m*(xmax-x2);
x2=xmax;
}
clrscr();
cleardevice();
printf("\n\t\tAfter clipping:");
rectangle(xmin,ymin,xmax,ymax);
line(x1,y1,x2,y2);
getch();
}
}
else
{
clrscr();
cleardevice();
printf("\nLine is invisible");
rectangle(xmin,ymin,xmax,ymax);
}

getch();
closegraph();
}

```

## **OUTPUT**

```

Enter the bottom-left coordinate of viewport: 100 20
Enter the top-right coordinate of viewport: 100 80
Enter the coordinates for starting point of line: 150 40
Enter the coordinates for ending point of line: 200 20

```



## **PRACTICAL 09**

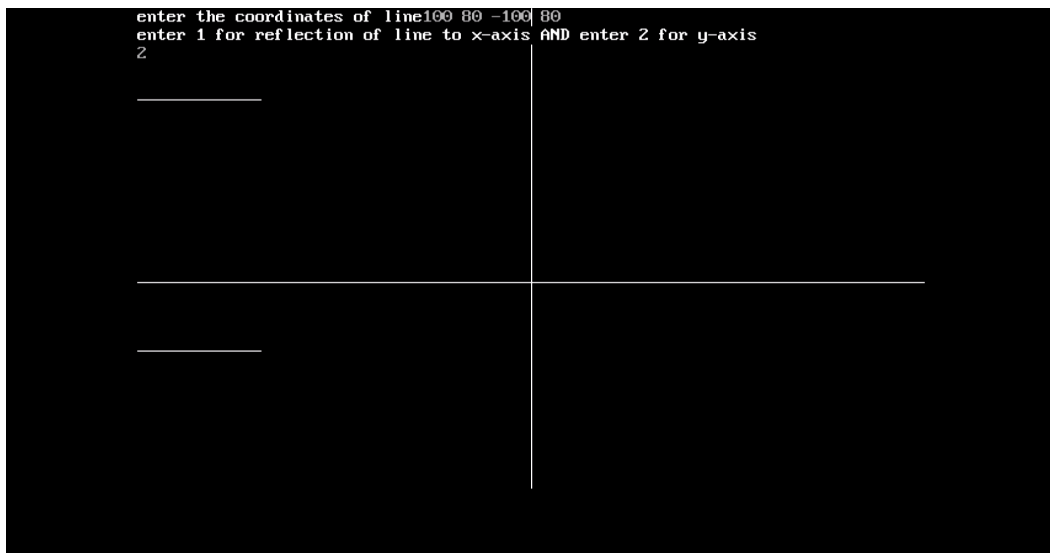
**AIM:** Write a program to implement Reflection of a line.

**SOFTWARE REQUIRED:** TURBO C++

**PROGRAM:**

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
int x1,y1,x2,y2,a;
clrscr();
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("enter the coordinates of line");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
line(x1,y1,x2,y2);
line(320,0,320,420);
line(0,240,640,240);
printf("enter 1 for reflection of line to x-axis AND enter 2 for y-axis\n");
scanf("%d",&a);
if(a==1)
{
line(x1+220,y1,x2+220,y2);
}
else if(a==2)
{
line(x1,y1+220,x2,y2+220);
}
else
{
printf("WRONG INPUT");
}
getch();
closegraph();
}
```

**OUTPUT**



## **PRACTICAL 01**

**AIM:** Write a program to implement DDA Algorithm.

**SOFTWARE REQUIRED:** TURBO C++

**PROGRAM:**

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main(){
int gd=DETECT,gm,x1,y1,x2,y2,x,y,length,dx,dy,xx,yy,i;
initgraph(&gd,&gm,"C:\\Turbo3\\BGI");
printf("enter starting and ending co-ordinates");
scanf("%d %d %d %d",&x1,&y1,&x2,&y2);
x=x2-x1;
y=y2-y1;
if( abs(x2-x1)>=abs(y2-y1) ){
length=abs(x2-x1);}
else
length=abs(y2-y1);
dx=(x2-x1)/length;
dy=(y2-y1)/length;
x=x1+0.5;
y=y1+0.5;
i=1;
while(i<=length){
```

```

putpixel(x,y,BLUE);
x=x+dx;
y=y+dy;
i=i+1;}
getch();
closegraph();}

```

## OUTPUT



## PRACTICAL 02

**AIM:** Write a program to implement Bresenham's Algorithm.

**SOFTWARE REQUIRED:** TURBO C++

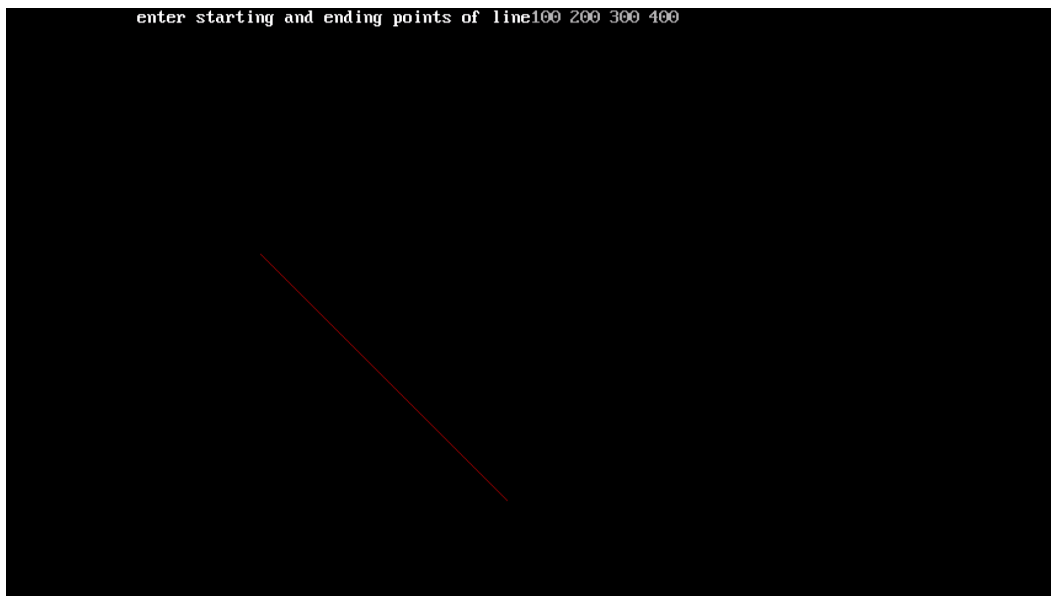
### PROGRAM:

```

#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,x1,y1,x2,y2,x,y,dx,dy,p;
initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
printf("enter starting and ending points of line");
scanf("%d %d %d %d",&x1,&y1,&x2,&y2);
putpixel(x1,y1,RED);
x=x1;
y=y1;
dx=x2-x1;
dy=y2-y1;
p=(2*dy)-dx;
while(x<x2)
{
if(p<0)
{
x=x+1;
y=y;
p=p+(2*dy);
}
else
{
x=x+1;
y=y+1;
p=p+(2*dy)-(2*dx);
}
putpixel(x,y,RED);
}
getch();
closegraph();
}

```

## OUTPUT



### **PRACTICAL 03**

**AIM:** Write a program to draw a circle using Bresenham's Circle Algorithm.

**SOFTWARE REQUIRED:** TURBO C++

**PROGRAM:**

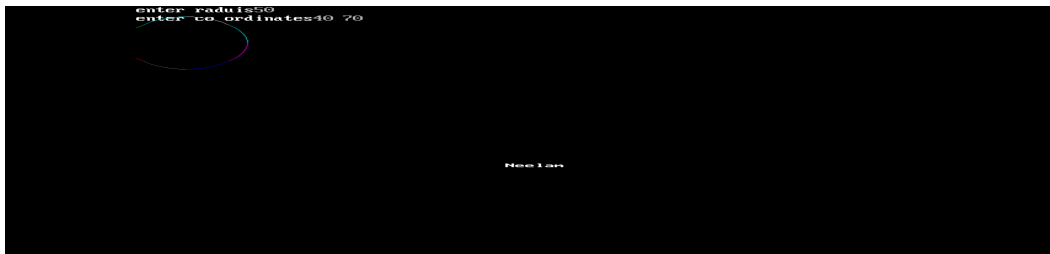
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int xc,yc,x,y,r,p;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
    printf("enter radius");
    scanf("%d",&r);
    printf("enter co ordinates");
    scanf("%d%d",&xc,&yc);
    x=0;
    y=r;
    p=3-(2*r);
    while(x<=y)
    {
        putpixel(xc+x,yc+y,1);
```

```

putpixel(xc-y,yc-x,2);
putpixel(xc+y,yc-x,3);
putpixel(xc-y,yc+x,4);
putpixel(xc+y,yc+x,5);
putpixel(xc-x,yc-y,3);
putpixel(xc+x,yc-y,7);
putpixel(xc-x,yc+y,8);
if(p<0)
{
x=x+1;
y=y;
p=p+(4*x)+6;
}
else
{
x=x+1;
y=y-1;
p=p+(4*(x-y))+10;
}}
getch();
closegraph();
}

```

### **OUTPUT**



## **PRACTICAL 04**

**AIM:** Write a program to perform Scaling.

**SOFTWARE REQUIRED:** TURBO C++

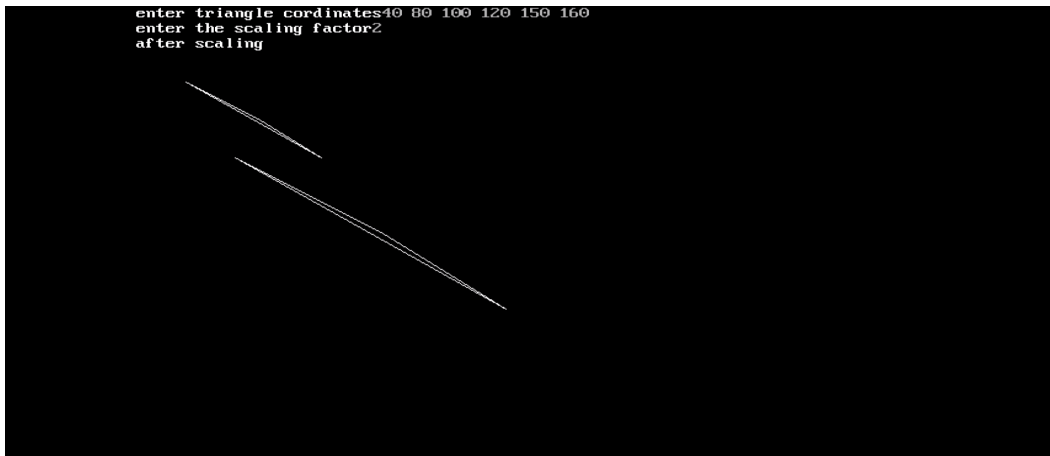


### **PROGRAM:**

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main(){
intgd=DETECT,gm,x1,y1,x2,y2,x3,y3,sc;
initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
printf("enter triangle coordinates");
scanf("%d %d %d %d %d %d",&x1,&y1,&x2,&y2,&x3,&y3);
line(x1,y1,x2,y2);line(x2,y2,x3,y3);line(x1,y1,x3,y3);
printf("enter the scaling factor");
scanf("%d",&sc);
printf("after scaling");
line(x1*sc,y1*sc,x2*sc,y2*sc);
line(x2*sc,y2*sc,x3*sc,y3*sc);
line(x1*sc,y1*sc,x3*sc,y3*sc);

getch();
closegraph();}
```

### **OUTPUT**



## **PRACTICAL 05**

**AIM:** Write a program to perform Rotation.

**SOFTWARE REQUIRED:** TURBO C++

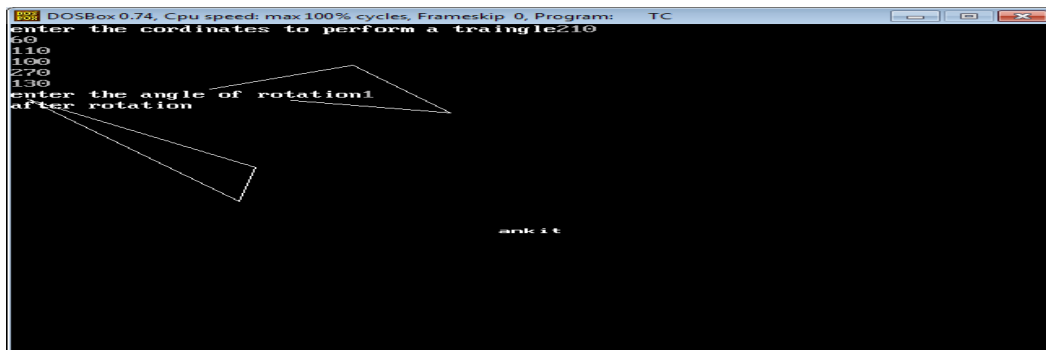
### **PROGRAM:**

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main(){
intgd=DETECT,gm,x1,y1,x2,y2,x3,y3,a;
initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
printf("enter the coordinates to perform a triangle");
scanf("%d %d %d %d %d %d",&x1,&y1,&x2,&y2,&x3,&y3);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x1,y1,x3,y3);
printf("enter the angle of rotation");
scanf("%d",&a);
x1=((x1*cos(a))-(y1*sin(a)));
y1=((x1*sin(a))+(y1*cos(a)));
x2=((x2*cos(a))-(y2*sin(a)));
y2=((x2*sin(a))+(y2*cos(a)));
x3=((x3*cos(a))-(y3*sin(a)));
y3=((x3*sin(a))+(y3*cos(a)));
printf("after rotation");
```

```
line(x1,y1,x2,y2);line(x2,y2,x3,y3);line(x1,y1,x3,y3);
```

```
getch();
closegraph();}
```

## OUTPUT



## PRACTICAL 06

**AIM:** Write a program to perform Composite Scaling.

**SOFTWARE REQUIRED:** TURBO C++

### PROGRAM:

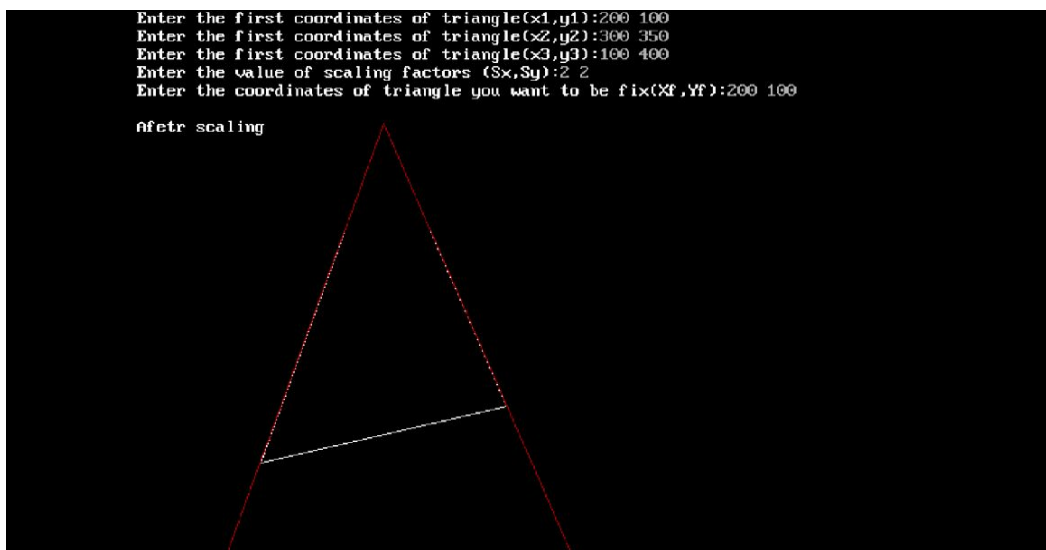
```
#include<stdlib.h>
#include<conio.h>
#include<iostream.h>
#include<math.h>
void main()
{
    int graphdriver=DETECT,graphmode,errorcode;
    int i,Sx,Sy,Xf,Yf,x1,x2,x3,y1,y2,y3,j,k;
    int t[3][3], l[3][3], f[3][3];
    initgraph(&graphdriver,&graphmode,"c:\\turboc3\\bgi");
    cout<<"Enter the first coordinates of triangle(x1,y1):";
    cin>>x1>>y1;
    cout<<"Enter the first coordinates of triangle(x2,y2):";
    cin>>x2>>y2;
    cout<<"Enter the first coordinates of triangle(x3,y3):";
    cin>>x3>>y3;
    line(x1,y1,x2,y2);
    line(x1,y1,x3,y3);
    line(x2,y2,x3,y3);
    cout<<"Enter the value of scaling factors (Sx,Sy):";
    cin>>Sx>>Sy;
    cout<<"Enter the coordinates of triangle you want to be fix(Xf,Yf):";
    cin>>Xf>>Yf;
    t[0][0]=Sx;
    t[0][1]=0;
    t[0][2]=Xf*(1-Sx);
    t[1][0]=0;
    t[1][1]=Sy;
    t[1][2]=Yf*(1-Sy);
    t[2][0]=0;
    t[2][1]=0;
    t[2][2]=1;
    l[0][0]=x1;
    l[0][1]=x2;
    l[0][2]=x3;
    l[1][0]=y1;
    l[1][1]=y2;
    l[1][2]=y3;
    l[2][0]=1;
    l[2][1]=1;
    l[2][2]=1;
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
```

```

{
f[i][j] = 0;
for (k = 0; k < 3; k++)
f[i][j] += t[i][k]*l[k][j];
}
}
x1=f[0][0];
x2=f[0][1];
x3=f[0][2];
y1=f[1][0];
y2=f[1][1];
y3=f[1][2];
cout<<"\nAfter scaling\n\n";
setcolor(RED);
line(x1,y1,x2,y2);
line(x1,y1,x3,y3);
line(x2,y2,x3,y3);
getch();
closegraph();}

```

## **OUTPUT**



## **PRACTICAL 07**

**AIM:** Write a program to perform Composite Rotation.

**SOFTWARE REQUIRED:** TURBO C++

```

#include<graphics.h>
#include<stdlib.h>
#include<conio.h>
#include<iostream.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
double i,Xr,Yr,x1,x2,x3,y1,y2,y3,j,k;
double t[3][3], l[3][3], f[3][3];
double th;
initgraph(&graphdriver,&graphmode,"c:\\turboc3\\bgi");
cout<<"Enter the first coordinates of line(x1,y1).:";
cin>>x1>>y1;
cout<<"Enter the first coordinates of line(x2,y2).:";
cin>>x2>>y2;
cout<<"Enter the first coordinates of line(x3,y3).:";
cin>>x3>>y3;
line(x1,y1,x2,y2);
line(x1,y1,x3,y3);
line(x2,y2,x3,y3);
cout<<"Enter the rotation angle (theta).:";
cin>>th;
}

```

```

cout<<"Enter the coordinates of pivot point(Xr,Yr).:";
cin>>Xr>>Yr;
t[0][0]=cos((th*3.1428)/180);
t[0][1]=(-sin((th*3.1428)/180));
t[0][2]=(Xr*(1-cos((th*3.1428)/180))+Yr*(sin((th*3.1428)/180)));
t[1][0]=sin((th*3.1428)/180);
t[1][1]=cos((th*3.1428)/180);
t[1][2]=(Yr*(1-cos((th*3.1428)/180))-Xr*(sin((th*3.1428)/180)));
t[2][0]=0;
t[2][1]=0;
t[2][2]=1;
l[0][0]=x1;
l[0][1]=x2;
l[0][2]=x3;
l[1][0]=y1;
l[1][1]=y2;
l[1][2]=y3;
l[2][0]=1;
l[2][1]=1;
l[2][2]=1;
for (i = 0; i < 3; i++)
{
for (j = 0; j < 3; j++)
{
f[i][j] = 0;
for (k = 0; k < 3; k++)
f[i][j] += t[i][k]*l[k][j];
}
}
x1=f[0][0];
x2=f[0][1];
x3=f[0][2];
y1=f[1][0];
y2=f[1][1];
y3=f[1][2];
cout<<"\nAfeetr rotation\n\n";
line(x1,y1,x2,y2);
line(x1,y1,x3,y3);
line(x2,y2,x3,y3);

getch();
closegraph();
}

```

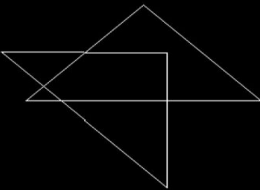
## OUTPUT

```

Enter the first coordinates of line(x1,y1):300 100
Enter the first coordinates of line(x2,y2):200 200
Enter the first coordinates of line(x3,y3):400 200
Enter the rotation angle (theta):45
Enter the coordinates of pivot point(Xr,Yr):250 150

Afeetr rotation

```



Activate Windows  
Go to PC settings to activate Windows.