Ex. No. 3	GENERATION OF LINE, CIRCLE AND ELLIPSE ATTRIBUTES

AIM

To draw line, circle and ellipse attributes using C program.

ALGORITHM

Step 1: Start
Step 2: Declare the necessary variables and functions to draw line, circle and ellipse.
Step 3: Initialize the gd, gm.
Step 4: Create a while and get the input choice form the user.
Step 5: Using switch case statement, call the required function to draw line, circle and ellipse attributes.
Step 6: If input choice is 1, Get the input co-ordinates points to draw a line using DDA algorithm. Get the input for the line style and fill style.
Step 7: Draw the line using the line style and fill style and exit.
Step 8: If input choice is 2, Get the radius to draw the circle. Get the centre co-ordinates, number for color, and number for fill style.
Step 9: Draw the circle using the number of color and fill style and exit.
Step 10: If the input choice is 3, Get the X radius and Y radius of the ellipse, get the centre co- ordinates, number for colour, and number for fill style.
Step 11: Stop the program.

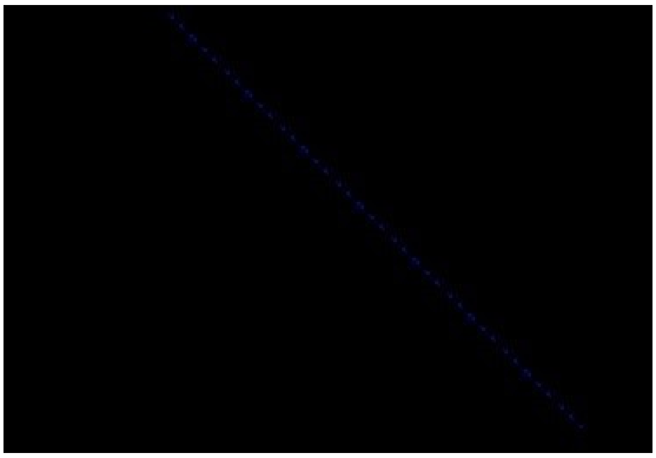
PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void lineatt();
void ciratt();
void ellatt();
void main()
{
int gd=DETECT,gm,f=1,ch;
initgraph(&gd,&gm,"..//bgi");
while(f==1)
{
clrscr();
cleardevice();
printf("\n *********MENU**********");
printf("1:Line attributes\n");
printf("2:Circle attributes\n");
printf("3:Ellipse attributes\n");
printf("4:Exit\n");
printf("\n *************************");
printf("\n \n Enter your choice : ");
scanf("%d",&ch);
switch(ch)
{
case 1:
lineatt();
break;
case 2:
ciratt();
break;
case 3:
ellatt();
break;
case 4:
f=0;
exit(0);
}}
getch();
closegraph();
}
void lineatt()
{
/* the names of the line styles supported
{0->"SOLID_LINE",1->"DOTTED_LINE",2->"CENTER_LINE",3-
>"DASHED_LINE"}*/
int x1,y1,x2,y2,sl,lt,lc;
printf("\n Enter line co-ordinate points:");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
printf("\n Enter a number for line style: \n");
scanf("%d",&sl);
printf("\n Enter for 1/2/3 for thickness:\n");
scanf("%d",&lt);
printf("\n Enter a number for color: \n");
scanf("%d",&lc);
clrscr();
cleardevice();
setcolor(lc);
setlinestyle(sl,1,lt);
line(x1,y1,x2,y2);
 getch();
}
void ciratt()
{
int x,y,c,cc,cf;
```

```
/* Fillstyle: { "EMPTY_FILL","SOLID_FILL","LINE_FILL","LTSLASH_FILL",
"SLASH_FILL","BKSLASH_FILL","LTBKSLASH_FILL",
"HATCH_FILL","XHATCH_FILL","INTERLEAVE_FILL", "WIDE_DOT_FILL",
"CLOSE_DOT_FILL", "USER_FILL"} */
printf("\n Enter radius:");
scanf("%d",&c);
printf("\n Enter center co-ordinates:");
scanf("%d%d",&x,&y);
printf("\n Enter a number for color:");
scanf("%d",&cc);
printf("\n Enter a number for fill style:");
scanf("%d",&cf);
clrscr();
cleardevice();
setcolor(cc);
setfillstyle(cf,cc);
circle(x,y,c);
floodfill(x,y,cc);
getch();
}
void ellatt()
{
int x,y,xc,yc,ec,ef;
printf("\n Enter X radius:");
scanf("%d",&xc);
printf("\n Enter Y radius:");
scanf("%d",&yc);
printf("\n Enter center co-ordinates:");
scanf("%d%d",&x,&y);
printf("\n Enter a number for color:");
scanf("%d",&ec);
printf("\n Enter a number for fill style:");
scanf("%d",&ef);
clrscr();
cleardevice();
setcolor(ec);
setfillstyle(ef,ec);
ellipse(x,y,0,360,xc,yc);
fillellipse(x,y,xc,yc);
getch();
}
```

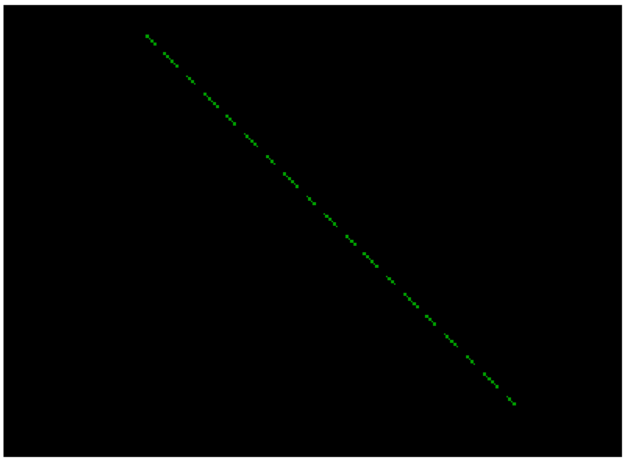
OUTPUT

```
*******MENU*********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
**********************
Enter your choice: 1
Enter line co-ordinate points: 100 100 250 250
Enter a number for line style : 1
Enter for 1/2/3 for thickeness : 1
Enter a number for color: 1
```

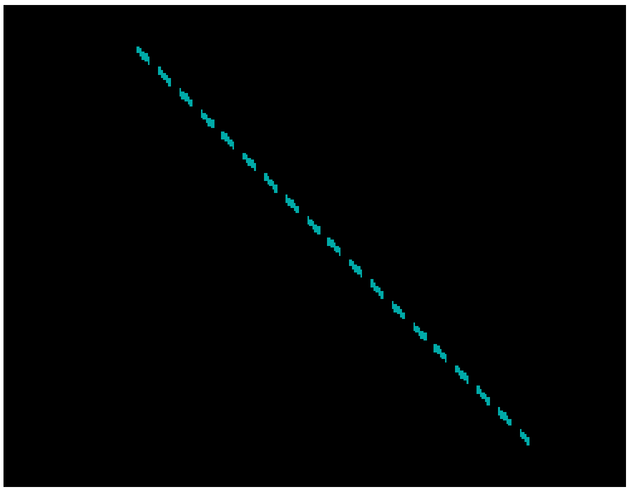


```
*******MENU*********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
**********************
Enter your choice: 1
Enter line co-ordinate points: 100 100 250 250
Enter a number for line style: 2
Enter for 1/2/3 for thickness: 2
Enter a number for color: 2

*******MENU*********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
**********************
Enter your choice: 1
Enter line co-ordinate points: 100 100 250 250
Enter a number for line style: 2
Enter for 1/2/3 for thickness: 2
Enter a number for color: 2
```

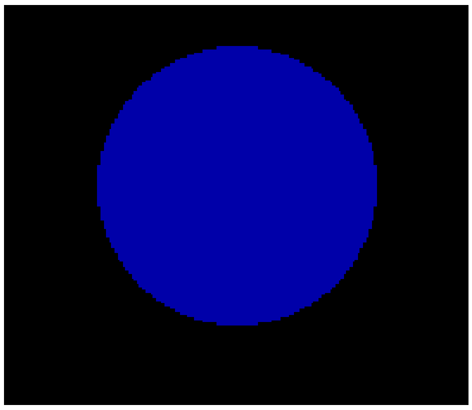


```
*******MENU********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
*********************
Enter your choice: 1
Enter line co-ordinate points: 100 100 250 250
Enter a number for line style: 3
Enter for 1/2/3 for thickness: 3
Enter a number for color: 3
```

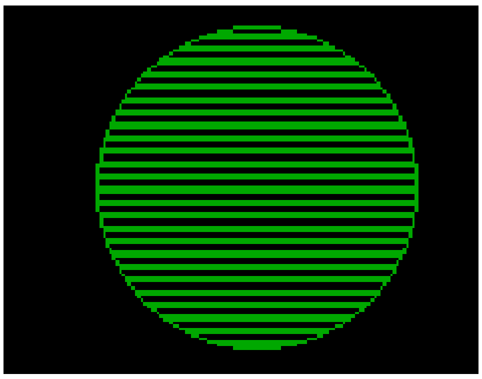


```
*******MENU********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
*********************
```

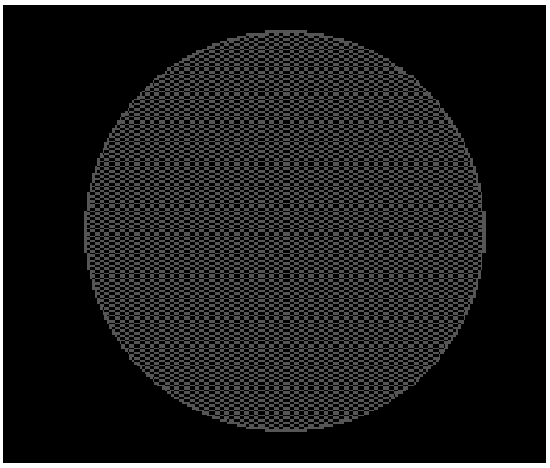
```
Enter your choice: 2
Enter radius: 50
Enter centre co-ordinates: 100 100
Enter a number for color:1
Enter a number for fillstyle: 1
```



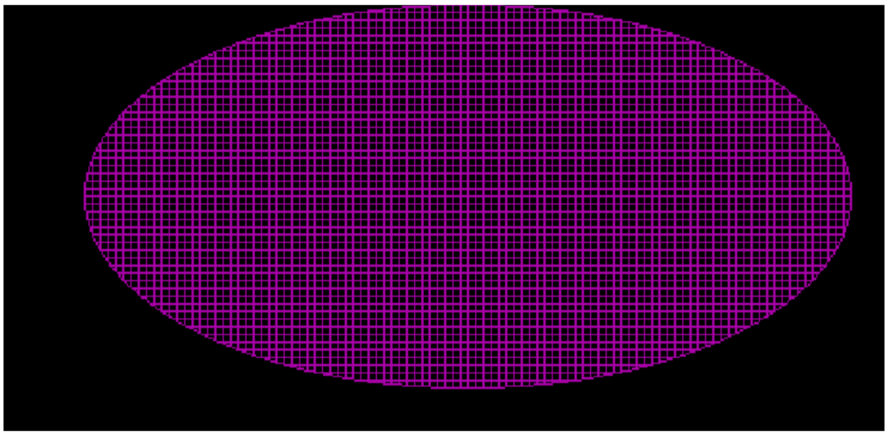
```
*******MENU*********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
*********************
Enter your choice: 2
Enter radius: 50
Enter centre co-ordinates: 100 100
Enter a number for color:2
Enter a number for fillstyle: 2
```



```
*******MENU*********
1.      Line attributes
2.      Circle attributes
3.      Ellipse attributes
4.      Exit
*********************
Enter your choice: 2
Enter radius: 80
Enter centre co-ordinates: 1580 150
Enter a number for color:
Enter a number for fillstyle: 9
```



```
*******MENU*********
1.        Line attributes
2.        Circle attributes
3.        Ellipse attributes
4.        Exit
**********************
Enter your choice: 3
Enter x radius: 200
Enter y radius: 100
Enter center co-ordinates: 200 100
Enter a number for color: 5
Enter a number for fill style: 7
```



RESULT

Thus the program to draw line, circle and ellipse attributes is implemented and executed successfully.

Ex. No. 4	TWO DIMENSIONAL TRANSFORMATIONS - TRANSLATION, ROTATION, SCALING, REFLECTION AND SHEAR

AIM

To implement 2D transformations using C program.

i) Translation
ii) Rotation
iii) Scaling
iv) Reflection
v) Shear

ALGORITHM

```
Step 1: Start the program.
Step 2: Declare the necessary variables and initialize the graph, gd, gm.
Step 3: Use do-while loop and declare the function clear device.
Step 4: Create four cases translation, scaling, rotation and exit.
Step 5: In case 1 enter the translation values and print the translation object.
Step 6: In case 2 enter the scaling values and print the scaling object.
Step 7: In case 3 enter the rotation values and print rotation object.
Step 8: Clockwise rotation and counter clockwise rotation use the same equation.
Step 9: Stop the program.
```

PROGRAM

```c
#include<stdio.h>
#include<conio.h>
 #include<math.h>
#include<graphics.h>
int ch,x,y,az,i,w,ch1,ch2,xa,ya,ra,a[10],b[10],da,db;
float x1,y1,az1,w1,dx,dy,theta,x1s,y1s,sx,sy,a1[10],b1[10];
void main()
{
int gm, gr;
clrscr();
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
printf("Enter the upper left corner of the rectangle:\n");
scanf("%d%d",&x,&y);
printf("Enter the lower right corner of the rectangle:\n");
scanf("%d%d",&az,&w);
rectangle(x,y,az,w);
da=az-x;
db=w-y;
a[0]=x;
```

```
b[0]=y;
a[1]=x+da;
b[1]=y;
a[2]=x+da;
b[2]=y+db;
a[3]=x;
b[3]=y+db;
while(1)
{
printf("******2DTransformations*******\n");
printf("1.Translation\n 2.Rotation\n 3.Scaling\n 4.Reflection\n 5.Shearing\n 6.Exit\n
Enter your choice:\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Translation*******\n\n");
printf("Enter the value of shift vector:\n");
scanf("%f%f",&dx,&dy);
 x1=x+dx;
y1=y+dy;
az1=az+dx;
w1=w+dy;
rectangle(x1,y1,az1,w1);
break;
case 2:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Rotation*******\n\n");
printf("Enter the value of fixed point and angle of rotation:Enter the value of fixed point
and angle of rotation:\n");
scanf("%d%d%d",&xa,&ya,&ra);
theta=(float)(ra*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((a[i]-xa)*cos(theta)-(b[i]-ya)*sin(theta)));
b1[i]=(ya+((a[i]-xa)*sin(theta)+(b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
```

```
if(i!=3) line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 3:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("********Scaling*******\n\n");
printf("Enter the value of scaling factor:\n");
scanf("%f%f",&sx,&sy);
x1=x*sx;
y1=y*sy;
az1=az*sx;
w1=w*sy;
rectangle(x1,y1,az1,w1);
break;
case 4:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Reflection*********\n");
printf("1.About x-axis\n2.About y-axis\n3.About both axis\nEnter your choice:\n");
scanf("%d",&ch1);
switch(ch1)
{
case 1:
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(90*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((a[i]-xa)*cos(theta)-(-b[i]-ya)*sin(theta)));
b1[i]=(ya+((a[i]-xa)*sin(theta)+(-b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
 if(i!=3)
line(a1[i],b1[i],a1[i+1],b1[i+1]);
else line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 2:
```

```
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(270*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((-a[i]-xa)*cos(theta)-(b[i]-ya)*sin(theta)));
b1[i]=(ya+((-a[i]-xa)*sin(theta)+(b[i]-ya)*cos(theta)));
}
for(i=0;i<4;i++)
{
if(i!=3) line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
case 3:
printf("Enter the fixed point\n");
scanf("%d%d",&xa,&ya);
theta=(float)(180*(3.14/180));
for(i=0;i<4;i++)
{
a1[i]=(xa+((-a[i]-xa)*cos(theta)-(-b[i]-ya)*sin(theta)));
b1[i]=(ya+((-a[i]-xa)*sin(theta)+(-b[i]-ya)*cos(theta)));
 }
for(i=0;i<4;i++)
{
if(i!=3) line(a1[i],b1[i],a1[i+1],b1[i+1]);
else
line(a1[i],b1[i],a1[0],b1[0]);
}
break;
}
break;
case 5:
detectgraph(&gm,&gr);
initgraph(&gm,&gr,"d:\\tc\\BGI");
rectangle(x,y,az,w);
printf("*******Shearing******\n\n");
printf("1.x-direction shear\n2.y-direction shear\nEnter your choice:\n");
scanf("%d",&ch2);
switch(ch2)
{
case 1:
printf("Enter the value of shear:\n");
```

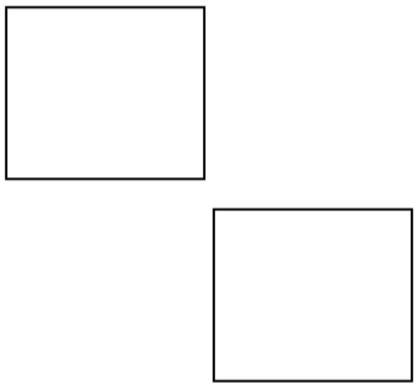
```
scanf("%f",&x1s);
x1=x+(y*x1s);
y1=y;
az1=az+(w*x1s);
w1=w;
rectangle(x1,y1,az1,w1);
break;
case 2:
printf("Enter the value of shear:\n");
scanf("%f",&y1s);
x1=x;
y1=y+(x*y1s);
az1=az;
w1=w+(az*y1s);
rectangle(x1,y1,az1,w1);
break;
}
break;
case 6:
exit(0);
}}
getch();
}
```

OUTPUT

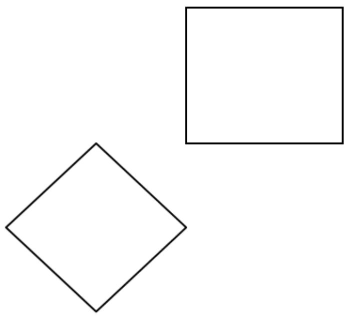
```
Enter the upper left corner of the rectangle: 210 210
Enter the bottom left corner of the rectangle: 240 240
```

```
******* 2D Transformation *********
1.      Translation
2.      Rotation
3.      Scaling
4.      Reflection
5.      Shearing
6.      Exit
Enter your choice : 1
******** Translation *******
```

```
Enter the value of shift vector: 30 40
```

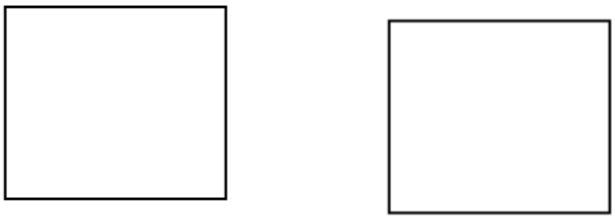


```
******* 2D Transformation *********
1.      Translation
2.      Rotation
3.      Scaling
4.      Reflection
5.      Shearing
6.      Exit
Enter your choice: 2
Enter the fixed point and angle of rotation: 30 30 30
```

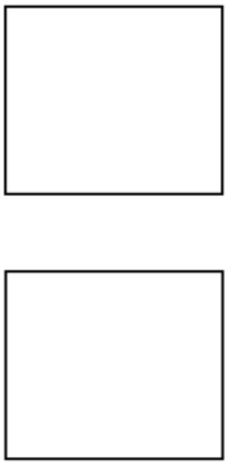


```
******* 2D Transformation *********
1.      Translation
2.      Rotation
3.      Scaling
4.      Reflection
5.      Shearing
6.      Exit
Enter your choice: 3
********* Scaling ***********
Enter the value of scaling factor:
2
1
```

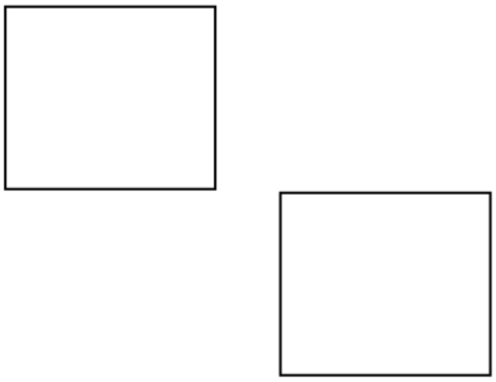
```
******* 2D Transformation ***********
1.        Translation
2.        Rotation
3.        Scaling
4.        Reflection
5.        Shearing
6.        Exit
Enter your choice: 4
******** Reflection ************
1.        About x-axis
2.        About y-axis
3.        About both axis
Enter your choice:1
Enter the fixed point 40 40
```



```
******* 2D Transformation ***********
1.  Translation
2.  Rotation
3.  Scaling
4.  Reflection
5.  Shearing
6.  Exit
Enter your choice: 4
******** Reflection ************
1.  About x-axis
2.  About y-axis
3.  About both axis
Enter your choice: 2
Enter the fixed point  40 40
```

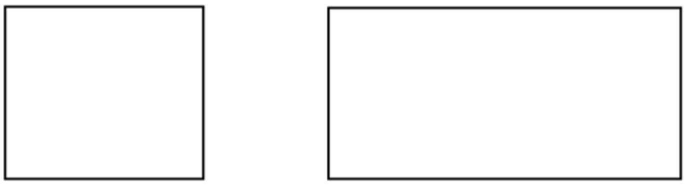


```
******* 2D Transformation ***********
1.  Translation
2.  Rotation
3.  Scaling
4.  Reflection
5.  Shearing
6.  Exit
Enter your choice: 4
******** Reflection ************
1.  About x-axis
2.  About y-axis
3.  About both axis
Enter your choice: 3
Enter the fixed point  40 40
```

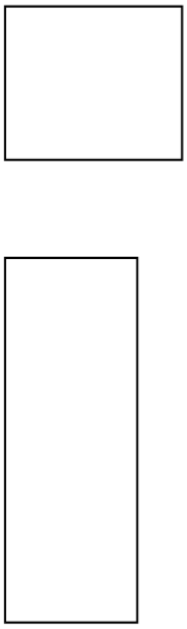


```
    ******* 2D Transformation ***********
    1.          Translation
    2.          Rotation
    3.          Scaling
    4.          Reflection
    5.          Shearing
    6.          Exit
```

```
Enter your choice: 5
********* Shearing *********
1.        X-direction shear
2.        Y-direction shear
Enter your choice: 1
Enter the value of shear: 0.5
```



```
******* 2D Transformation ***********
1.        Translation
2.        Rotation
3.        Scaling
4.        Reflection
5.        Shearing
6.        Exit
Enter your choice: 5
********* Shearing *********
1.        X-direction shear
2.        Y-direction shear
Enter your choice: 2
Enter the value of shear: 0.5
```



RESULT

Thus the program to implement 2D transformations is executed successfully.

Ex. No. 5 (A)	COHEN - SUTHERLAND 2D LINE CLIPPING ALGORITHM

AIM

To implement Cohen - Sutherland 2D line clipping algorithm using C program.

ALGORITHM

```
Step 1: Start.
Step 2: Get the co-ordinates of line (x11,y11) & (x22,y22)
Step 3: Get the co-ordinate of clipping window (xmin,ymin,xmax,ymax)
Step 4: Get the code region for line
        (a)if(x11<xmin) a[3]=1
        (b)if(x11>xmax) a[2]=1
        (c)if(y11<ymin) a[1]=1;
        (d)if(y11>ymax) a[0]=1;
        (e)if(x22<xmin) b[3]=1;
        (f)if(x22>xmax) b[2]=1;
        (g)if(y22<ymin) b[1]=1;
        (h)if(y22>ymax) b[0]=1;
Step 5: Calculate slope of line i.e m using following formula: m=(y22-y11)/(x22-x11)
Step 6: If a[3],a[2],a[1],a[0],b[3],b[2],b[1],b[0] all are zero Then the line is totally
visible and not a clipping candidate
Step 7: Draw line with coordinates (x11,y11) & (x22,y22) and window with coordinates
(xmin,ymin,xmax,ymax)
Step 8: Line will partially visible if any of following conditions is satisfied:
        (i) if((a[0]=0)&(a[1]=1))
        x11=x11+(ymin-y11)/m
        y11=ymin
        else if((b[0]=0)&(b[1]=1))
        x22=x22+(ymin-y22)/m
        y22=ymin

        (ii) if((a[0]==1)&(a[1]==0))
        x11=x11+(ymax-y11)/m
        y11=ymax
        else if((b[0]=1)&(b[1]=0))
        x22=x22+(ymax-y22)/m;
        y22=ymax;

        (iii)if((a[2]=0)&(a[3]==1))
        y11=
        }  }
        else
        {
            clrscr();
            clearviewport();
            printf("\nLine is invisible");
            rectangle(xmin,ymin, y11+m*(xmin-x11)
            x11=xmin
            else if((b[2]=0)&(b[3]=1))
            y22=y22+m*(xmin-x22) x22=xmin

            (iv)if((a[2]=1)&(a[3]=0))
            y11=y11+m*(xmax-x11) x11=xmax
            else if((b[2]=1)&(b[3]=0))
            y22=y22+m*(xmax-x22)
            x22=xmax
```

Step 9: Draw line with coordinates (x11,y11) & (x22,y22) and window with coordinates (xmin,ymin,xmax,ymax)

Step 10: If any condtion described in step-5 and step-7 is not satisfied, then line is invisible i.e outside of clipping window.

Step 11: Stop

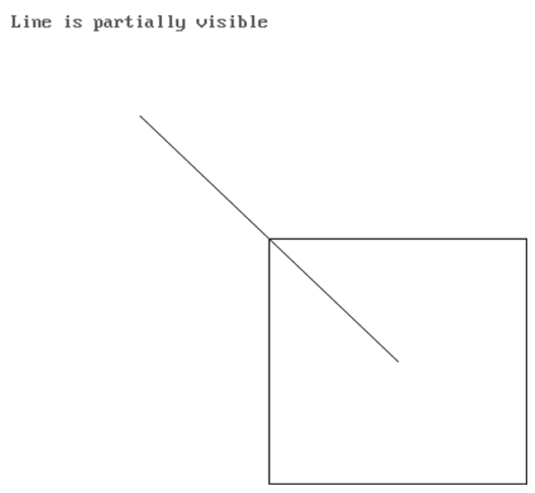
PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
void main()
{
int gd=DETECT, gm;
float i,xmax,ymax,xmin,ymin,x11,y11,x22,y22,m;
float a[4],b[4],c[4],x1,y1;
clrscr();
initgraph(&gd,&gm,"c:\\dosapp~1\\tc\\bgi");
printf("\nEnter the 1st co-ordinate of line: ");
scanf("%f %f ",&x11,&y11);
printf("\nEnter the 2nd co-ordinate of line: ");
scanf("%f %f ",&x22,&y22);
printf("\nEnter the co-ordinates of clipping window: ");
scanf("%f %f %f %f",&xmin,&ymin,&xmax,&ymax);
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
for(i=0;i<4;i++)
{
a[i]=0;
b[i]=0;
}
```

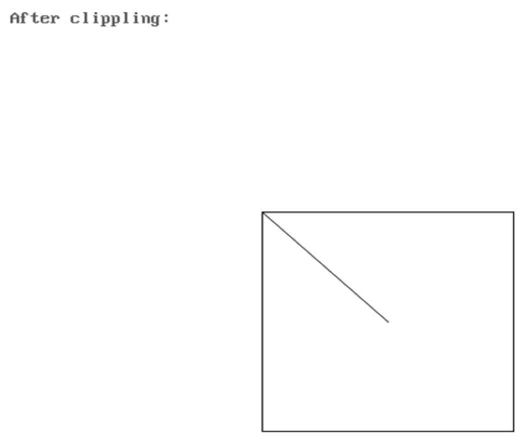
```
m=(y22-y11)/(x22-x11);
if(x11<xmin) a[3]=1;
if(x11>xmax) a[2]=1;
if(y11<ymin) a[1]=1;
if(y11>ymax) a[0]=1;
if(x22<xmin) b[3]=1;
if(x22>xmax) b[2]=1;
if(y22<ymin) b[1]=1;
if(y22>ymax) b[0]=1;
for(i=0;i<4;i++)
{
c[i]=a[i]&&b[i];
}
if((c[0]==0)&&(c[1]==0)&&(c[2]==0)&&(c[3]==0))
{
if((a[0]==0)&&(a[1]==0)&&(a[2]==0)&&(a[3]==0)&&(b[0]==0)&&(b[1]==0)&&(b
[2]==0)&&(b[3]==0))
{
 clrscr();
clearviewport();
printf("\n The line is totally visible\n and not a clipping candidate");
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
getch();
}
else   {
clrscr();
clearviewport();
printf("\nLine is partially visible");
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
getch();
if((a[0]==0)&&(a[1]==1))
{
x1=x11+(ymin-y11)/m; x11=x1;
y11=ymin;
}
else if((b[0]==0)&&(b[1]==1))
{
x1=x22+(ymin-y22)/m;
x22=x1;
y22=ymin;
}
if((a[0]==1)&&(a[1]==0))
{
x1=x11+(ymax-y11)/m;
x11=x1;
y11=ymax;
}
else if((b[0]==1)&&(b[1]==0))
{
x1=x22+(ymax-y22)/m;
x22=x1;
y22=ymax;
}
if((a[2]==0)&&(a[3]==1))
{
y1=y11+m*(xmin-x11);
y11=y1;
x11=xmin;
}
else if((b[2]==0)&&(b[3]==1))
{
y1=y22+m*(xmin-x22);
y22=y1;
x22=xmin;
}
if((a[2]==1)&&(a[3]==0))
{
y1=y11+m*(xmax-x11);
y11=y1;
x11=xmax;
}
else if((b[2]==1)&&(b[3]==0))
{
y1=y22+m*(xmax-x22);
y22=y1;
x22=xmax;
}
clrscr();
clearviewport();
printf("\nAfter clippling:");
rectangle(xmin,ymin,xmax,ymax);
line(x11,y11,x22,y22);
getch();
(xmax,ymax);
getch();
}}
```

OUTPUT

```
Enter the 1-st co-ordinates of line: 150 300
Enter the 2nd co-ordinates of line: 450 300
Enter the co-ordinates of clipping window: 200 200 400 400
```

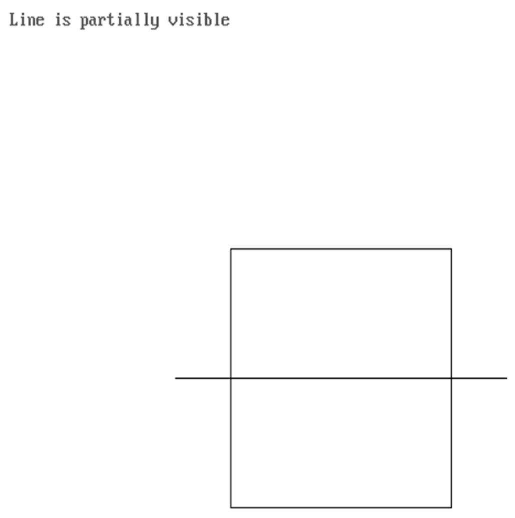


```
Line is partially visible
```

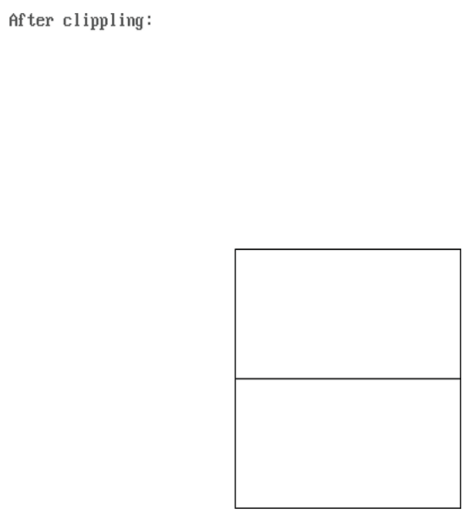


```
After clippling:
```

```
Enter the 1-st co-ordinates of line : 150 300
Enter the 2nd co-ordinates of line: 450 300
Enter the co-ordinates of clipping window: 200 200 400 400
```



```
Line is partially visible
```



```
After clippling:
```

RESULT

Thus the program to implement Cohen-Sutherland 2D line clipping is executed successfully.

Ex. No. 5 (B)	WINDOWING TRANSFORMATION

AIM

To implement windowing transformation using C program.

ALGORITHM

```
Step 1: Start the program.
Step 2: Initialize the variables gd=DETECT, gm for graphics mode.
Step 3: Get the co-ordinate points to draw the object.
Step 4: Assign the co-ordinates for the world port.
Step 5: Using rectangle function for drawing the world port.
Step 6: Using line function draw the object.
Step 7: Assign the co-ordinates for the window port.
Step 8: Using rectangle function for drawing the window port.
Step 9: Apply scaling factor for the object in world port.
Step 10: Using line function draw the object.
Step 11: Stop the program.
```

PROGRAM

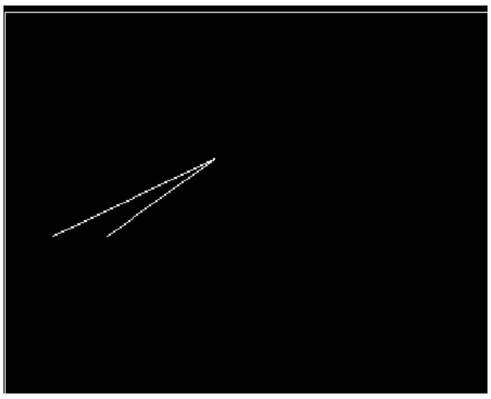
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
main()
{
float sx,sy;
int w1,w2,w3,w4,x1,x2,x3,x4,y1,y2,y3,y4,v1,v2,v3,v4;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..//bgi");
printf("Enter the Co-ordinates x1,y1,x2,y2,x3,y3\n");
scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);
cleardevice();
w1=5; w2=5; w3=635; w4=465;
rectangle(w1,w2,w3,w4);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
getch();
v1=425; v2=75; v3=550; v4=250;
sx=(float)(v3-v1)/(w3-w1);
sy=(float)(v4-v2)/(w4-w2);
rectangle(v1,v2,v3,v4);
x1=v1+floor(((float)(x1-w1)*sx)+0.5);
x2=v1+floor(((float)(x2-w1)*sx)+0.5);
x3=v1+floor(((float)(x3-w1)*sx)+0.5);
y1=v2+floor(((float)(y1-w2)*sy)+0.5);
```

```
        y2=v2+floor(((float)(y2-w2)*sy)+0.5);
        y3=v2+floor(((float)(y3-w2)*sy)+0.5);
        line(x1,y1,x2,y2);
        line(x2,y2,x3,y3);
        getch();
        return 0;   }
```

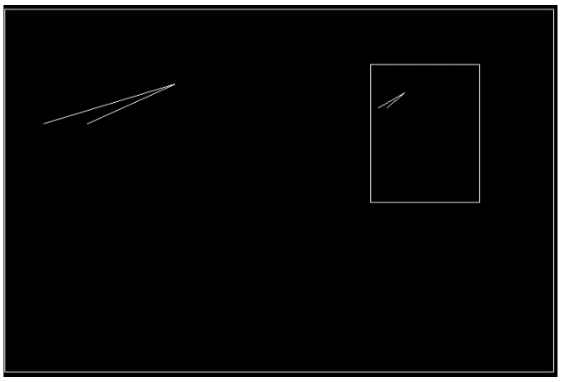
OUTPUT

```
        Enter the Co-ordinates x1,y1,x2,y2,x3,y3 100
        150
        200
        100
        50
        150
```

Before Transformation



After Transforation



RESULT

Thus the C program to implement windowing transformation is implemented successfully.