EXCEL ENGINEERING COLLEGE

KOMARAPALAYAM –637 303

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

20CS405-COMPUTER GRAPHICS LABORATORYRECORD NOTE BOOK

INDEX

S.No	Date	Name of Experiment	Page No.	Signature
1.A		IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVELINE-DDA ALGORITHM		
1.B		IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVELINE-BRESENHEM ALGORITHM		
1.C		IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVELINE-MIDPOINT CIRCLE DRAWING		
1.D		ELLIPSE DRAWING		
2		2D TRANSFORMATION		
3		IMPLEMENTATION OF COMPOSITE TRANSFORMATION		
4		COHEN SUTHERLAND LINE CLIPPING ALGORITHM		
5		3D TRANSFORMATION-TRANSLATION,ROTATION AND SCALING		
6		3D PROJECTIONS-PARALLEL,PERSPECTIVE		
7.		CREATING 3D SCENES		
8.		IMAGE EDITING MANIPULATION		
9.		2D ANIMATION		

EX NO	**IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVELINE-DDA ALGORITHM**
DATE:	

AIM:

To implement the algorithms for drawing a line using DDA algorithm.

ALGORITHM:

1. Start the algorithm
2. Declare the necessary variables
3. Initialize the graph using dx,dy,gd and gm
4. Assign values xa and ya,xb and yb
5. Let the pixel position as 57
6. By using printf and while loop values, write a condition steps
7. Compute dx=xb-xa;dy=yb-ya
8. If check abs(dx)>abs(dy),then condition is true mabs(dy).assign the steps
9. Then da/(float)steps,operation is performed
10. Then store xtncrement same as yincrement
11. Repeat step8 when condition is false
12. Stop the program

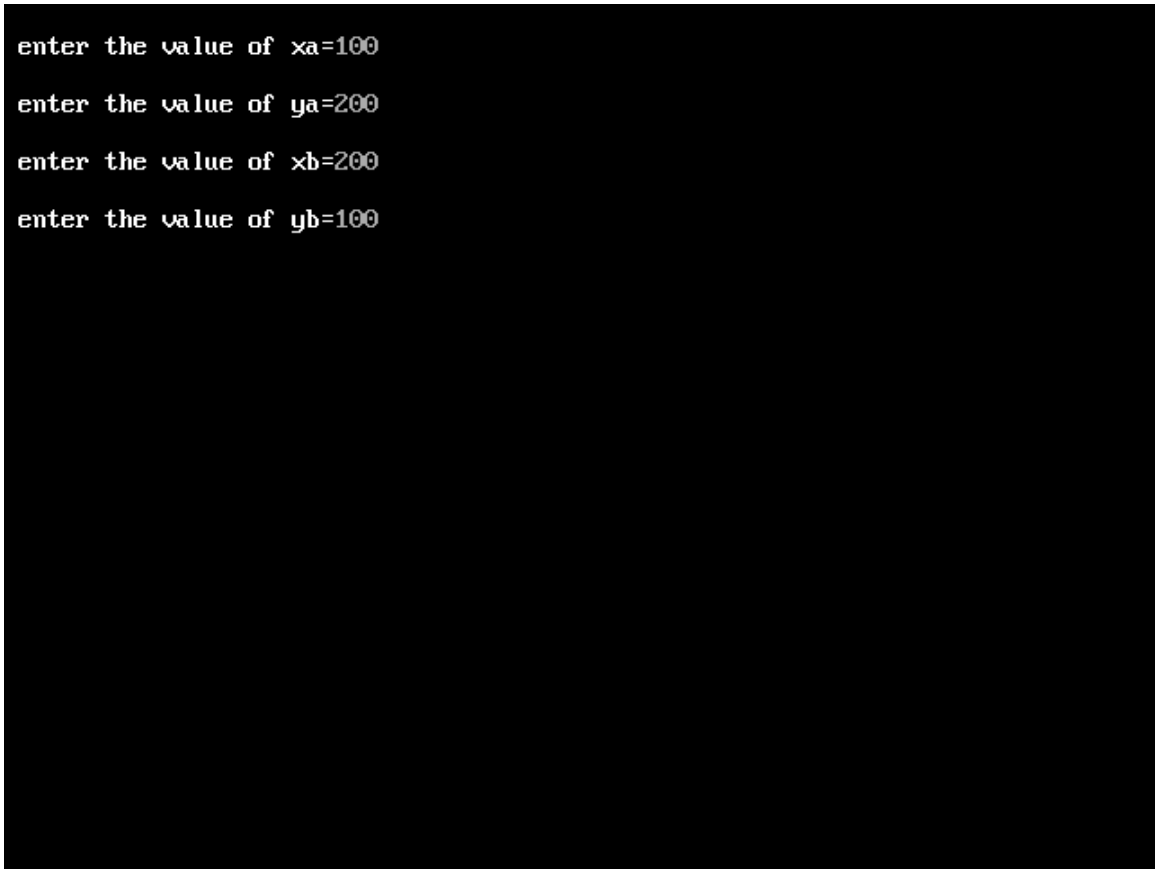
1. A.IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVE LINE-DDA

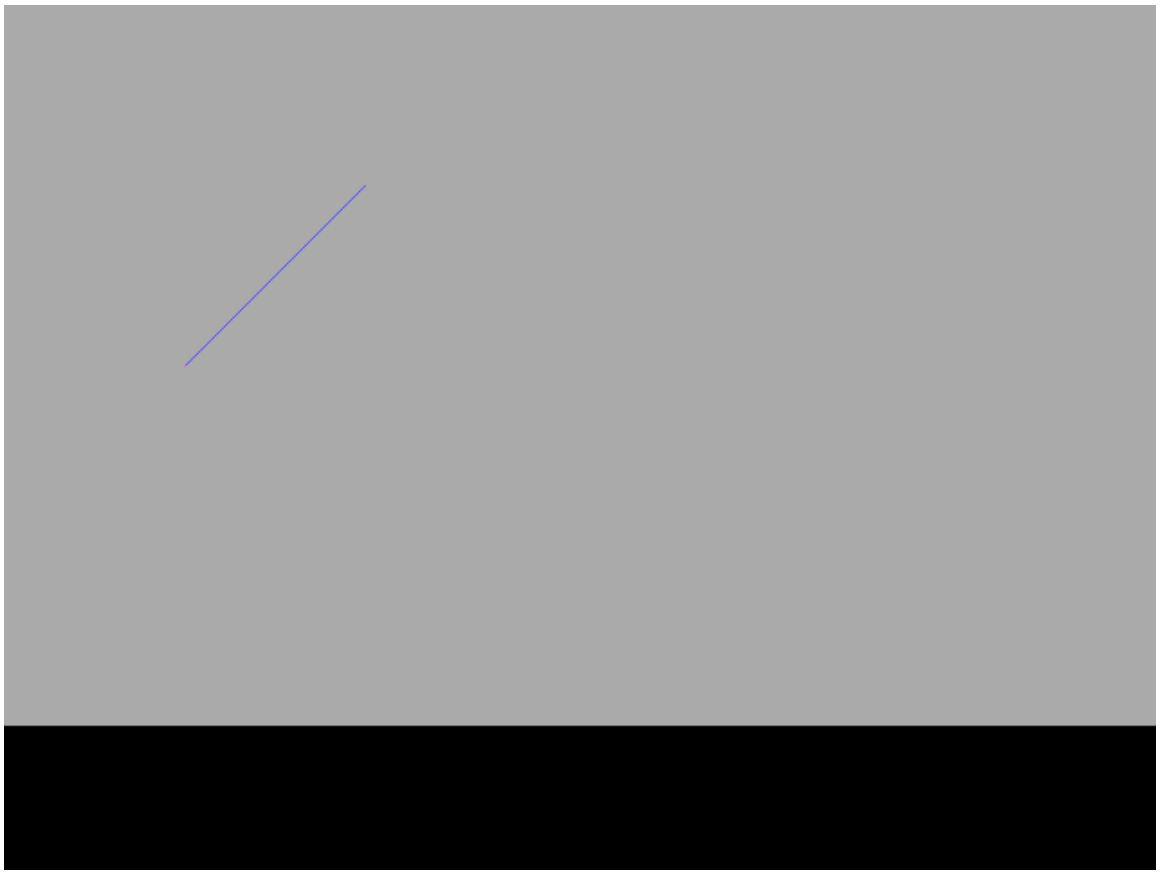
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c://turboc3//bgi");
int xa,yb,ya,steps,k,xb;
printf("\nenter the values of xa:");
scanf("%d",&xa);
printf("\n enter the values of xb:");
scanf("%d",&xb);
printf("\nenter the values of ya");
scanf("%d",&ya);
printf("\nenter the value of yb");
scanf("%d",&yb);
int dx=xb-xa;
int dy=yb-ya;
```

```
float xIncrement,yIncrement,x=xa,y=ya;
clrscr();
if(abs(dx)>abc(dy))
steps=abs(dx);
else
steps=abc(dy);
xIncrement=dx/(float)steps;
yIncrement=dy/(float)steps;
putpixel(x,y,125);
for(k=0;k<steps;k++)
{
x+=xIncrement;
y+=yIncrement;
putpixel(x,y,57);
}
getch();
}
```

OUTPUT:


enter the value of xa=100
enter the value of ya=200
enter the value of xb=200
enter the value of yb=100



RESULT:

Thus the implementation of drawing algorithm of 2D primitive using DDA algorithm is executed successfully.

EX NO:

DATE:

IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVELINE-BRESENHEM ALGORITHM

AIM:

To implement the algorithms for drawing a line using bresenhem algorithm and also drawing a circle using midpoint algorithm.

ALGORITHM:

1. Input the two line endpoints and store the left end point in(x2,y2)
2. Load(x1,y1)into the frame buffer that is plot the first point
3. Calculate constants dxdy,2dy and 2dy-2dx and obtain the starting value for the decision parameter as p0=2dy-dx
4. At each xk along the line,starting at k=0,perform test:
    - If pk<0,the next point to plot is (xk+1,yk) and pk+1=pk+2dy
    - Otherwise,the next point to plot is (xk+1,yk+1) and pk+1=pk+2dy-2dx
5. Repeat step 4 dx times

IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVE LINE- BRESENHAM ALGORITHM

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
int x1,x2,y1,y2;
void linebres(int,int,int,int);
clrscr();
initgraph(&gd,&gm,"c://turboc3//bgi");
printf("enter the string point\n");
scanf("%d%d",&x1,&y1);
printf("enter the end point\n");
scanf("%d%d",&x2,&y2);
linebres(x1,y1,x2,y2);
getch();
}
void linebres(int xa,int ya,int xb,int yb)
{
int dx,dy,p,twody,twodxdy,x,y,xend;
```

```
dx=abs(xa-xb);
dy=abs(ya-yb);
p=2*dy-dx;
twody=2*dy;
twodxdy=2*(dy-dx);
if(xa>xb)
{
x=xb;
y=yb;
xend=xa;
}
else
{
x=xa;
y=ya;
xend=xb;
}
putpixel(x,y,15);
while(x<xend)
{
x++;
if(p<0)
p+=twody;
else
{
```

```
y++;
p+=twodxdy;
}
putpixel((int) x,(int)y,15);
}
}
```

OUTPUT:

RESULT:

Thus the implementation of the algorithms for drawing a line using Bresenham algorithm has been implemented and the output was verified.

EX NO:	**IMPLEMENTATION OF ALGORITHM FOR DRAWING 2D PRIMITIVELINE-MIDPOINT CIRCLE DRAWING**
DATE:	

AIM:

To implement the algorithms for drawing a line using midpoint circle algorithm and and also drawing a circle using midpoint algorithm

ALGORITHM:

Input radiuys r and circle center (x1,y1) and obtain the first point on the circumference of circle entered on the origin (x0,y0)=(0,r)

1. Calculate the initial value of decoision parameter as p0=5/4-r
2. At each xk position,starting at k=0,perform the following
    - If (pk<0),point along the circle centered on (0,0) is (xk,yk) and pk+1=pk+2xk+1+1-2yk+1
    - Otherwise,point along the circle is (xk+1,yk-1) and pk+1= pk+2xk+1+1-2yk+1

4.determine the symmetry points in other seven octants

5.move each calculated pixel position (x,y) onto the circular path centered on (xc,yc) and plot the coordinate values:x=x+xc,y=y+yc

6.repeat steps 3 through 5 until x>=y

1.C IMPLEMENTATION OF ALGORITHM FOR DRAWING PRIMITIVE LINE-CIRCLE

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
int x1,y1,r;
void circlemid(int,int,int);
clrscr();
initgraph(&gd,&gm,"c://turboc3//bgi");
printf("\n Enter the x and y value");
scanf("%d%d",&x1,&y1);
printf("enter the radius");
scanf("%d",&r);
circlemid(x1,y1,r);
getch();
}
void circlemid(int x1,int y1,int r)
{
int x,y,p;
```

```
void circleplot(int,int,int,int);
x=0;
y=r;
p=1-r;
circleplot(x1,y1,x,y);
while(x<y)
{
x++;
if(p<0)
p+=2*x+1;
else
{
y--;
p+=2*(x-y);
}
circleplot(x1,y1,x,y);
}
}
void circleplot(int xc,int yc,int x,int y)
{
putpixel(xc+x,yc+y,15);
putpixel(xc-x,yc+y,15);
putpixel(xc+x,yc-y,15);
```

```
putpixel(xc-x,yc-y,15);
putpixel(xc+y,yc+x,15);
putpixel(xc-y,yc+x,15);
putpixel(xc+y,yc-x,15);
putpixel(xc-y,yc-x,15);
}
```

OUTPUT:

RESULT:

Thus the implementation of the algorithms for drawing a line using a circle using midpoint algorithm has been implemented and the output was verified

EX NO:	IMPLEMENTATION OF ALGORITHMS FOR DRAWING 2D PRIMITIVES - ELLIPSE ALGORITHM
DATE:	

AIM:

To implement the algorithms for drawing a line using ellipse algorithm using midpoint algorithm.

ALGORITHM:

1. Input the radius rx,ry and ellipse center (xcen,ycen) and obtain the first point on an ellipse centered on the origin as (x0,y0)=(0,r)
2. Calculate the initial value of the decision p[arameter as p10=r2y-r2(ry)+1/4(rx2)
3. At each xk position,starting at k=0.perform the following
    - If R<<0,point alo0ng the cvircle centered on (0,0) is (xk,yk) and p1k+1=p1k+2rx2k+1+ry2
    - Otherwise,point along the circle is (xk+1,yk-1) and pk1k+1=pik+2rx2xk+1-2rx2yk+1ry2
4. calculate the initaial value of the decision parameter 2 as p20=ry2o(x0+1/2)2+rx2(y0+1/2)2-rx2ry2
5. At each xk position,starting at k=0,perform the following
    - If p2k>0,point along the circle centered on (0,0) is (xk,yk) and p2k+1=p2x+2xrxk+12+rx2
    - Otherwise,point along the circle is (xk+1,yk91) and p2k+1=p2k+2ry2xk+1-2rx2yk+1rx
6. Determine symmetry points in the other three octants
7. Move each calculated pixel position (x,y) onto the circular path centered on (xc,yc) and the coordinate values: x=x+xc;y=y+yc
8. Repeat steps of region 2ry2x>=2rx2y

IMPLEMENTATION OF ALGORITHM FOR DRAWING PRIMITIVELINE-ELLIPSE

PROGRAM:

```
#define round(a)((int)(a+0.25))
#include<stdio.h>
#include<stdlib.h>
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
void ellipsedraw(int,int,int,int);
int gd=DETECT,gm,xcen,ycen,rx,ry;
initgraph(&gd,&gm,"c://turboc3//bgi");
cout<<"enter the x radius and y radius:";
cin>>rx>>ry;
xcen=getmaxx()/2;
ycen=getmaxy()/2;
ellipsedraw(xcen,ycen,rx,ry);
getch();
}
void ellipsedraw(int xc,int yc,int rx,int ry)
{
int rx2,ry2,tworx2,twory2,p,x=0,y,px=0,py;
rx2=rx*rx;
ry2=ry*ry;
tworx2=2*rx2;
twory2=2*ry2;
y=ry;
py=tworx2*y;
void ellipseplot(int,int,int,int);
ellipseplot(xc,yc,x,y);
p=round(ry2-(rx2*ry)+(0.25*rx2));
while(px<py)
{
x++;
px+=twory2;
if(p<0)
```

```
p+=ry2+px;
else
{
y--;
py-=tworx2;
p+=ry2+px-py;
}
ellipseplot(xc,yc,x,y);
}
p=round(ry2*(x+0.5)*(x+0.5)+rx2*(y-1)*-rx2*ry2);
while(y>0)
{
y--;
py-=tworx2;
if(p>0)
p+=rx2-py;
else
{
x++;
px+=twory2;
p+=rx2-py+px;
}
ellipseplot(xc,yc,x,y);
}
}
void ellipseplot(int xc,int yc,int x,int y)
{
putpixel(xc+y,yc+x,15);
putpixel(xc-y,yc+x,15);
putpixel(xc+y,yc-x,15);
putpixel(xc-y,yc-x,15);
}
```

OUTPUT:


enter the x radius and y radius:20 30

RESULT:

Thus the implementation of the algorithms for drawing a line using ellipse algorithm has been implemented and the output was verified

EX NO:	IMPLEMENTATION OF 2D GEOMETRIC TRANSFORMATION
DATE:	

AIM:

To implement the following 2D Geometric Transformations

a. Translation
b. Rotation
c. Scaling
d. Reflection
e. Shearing
f. Window-View Port

ALGORITHM:

1. Get the coordinates of triangle (x1, y1, x2, y2, x3, y3).
2. Draw the original triangle.
3. Print the menu for choosing 2D Geometric Transformation.

a. If user choose translation then get the translation factors(x, y) and draw the translated
triangle in the following coordinates (x1+x, y1+y, x2+x, y2+y, x3+x, y3+y).

b. (i) If user choose rotation then get the rotation angle (t) and reference point of the rotation (rx, ry).

(ii) Change the t value to t = t * (3.14 / 180) and calculate the rotated coordinates by the following formulae

rx1 = rx + (x1 - rx) * cos (t) - (y1 - ry) * sin (t);
ry1 = ry + (x1 - rx) * sin (t) + (y1 - ry) * cos (t);

(iii) Similarly calculate the coordinates rx2, ry2, rx3, ry3 and draw the rotated triangle in the following coordinates (rx1, ry1, rx2, ry2, rx3, ry3).

c. If user choose scaling then get the scaling factors(x, y) and draw the scaleded triangle
in the following coordinates (x1*x, y1*y, x2*x, y2*y, x3*x, y3*y).

d. If user choose reflection then rotate the triangle in 180
0
at (x2, y2) and draw the
rotated triangle (which is reflected triangle).

e. If user choose shearing then get the shear value and draw the sheared triangle in the
following coordinates (x1, y1, x2+x, y2, x3, y3).

f. (i) If user choose window-view port then draw the rectangle in window port

coordinates (w1, w2, w3, w4) and draw the original triangle.
(ii) calculate the x, y and view port coordinates by following formulae
x = (v3 - v1) / (w3 - w1);
y = (v4 - v2) / (w4 - w2);
vx1 = v1 + floor (((x1 - w1) * x) + 0.5);
vy1 = v2 + floor (((y1 - w2) * y) + 0.5);
(iii) Similarly calculate the coordinates vx2, vy2, vx3, vy3
(iv) Draw the rectangle in view port coordinates (v1, v2, v3, v4) and draw the triangle in view port by the following coordinates (vx1, vy1, vx2, vy2, vx3, vy3)

2.IMPLEMENTATION OF 2D GEOMETRIC TRANSFORMATIONS

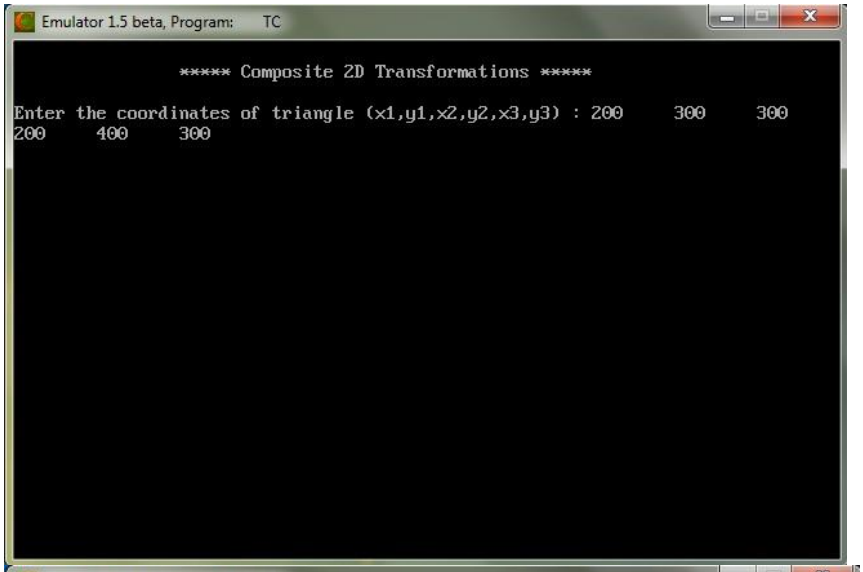
PROGRAM:

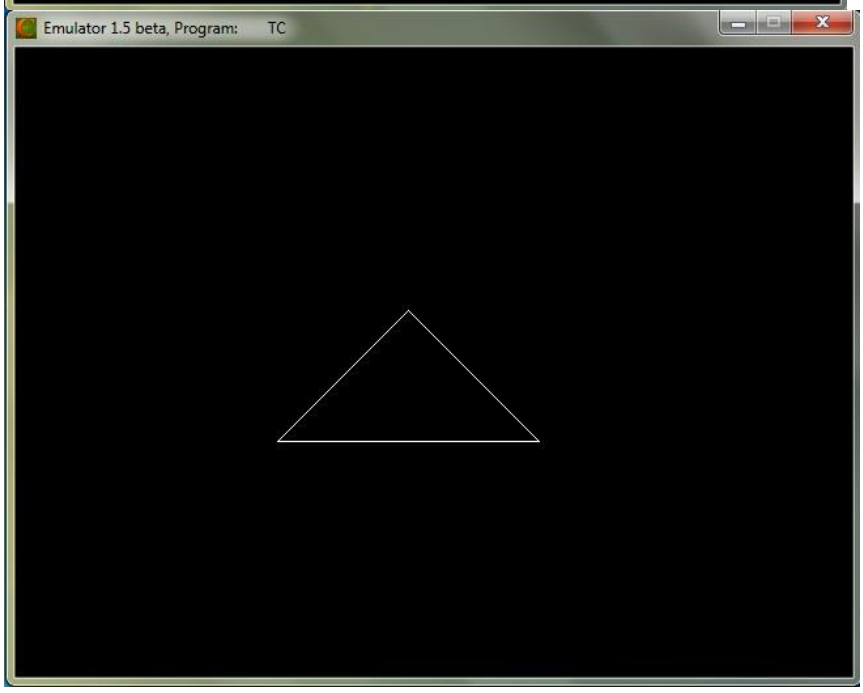
```
#include <graphics.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
float x1, y1, x2, y2, x3, y3, x, y, rx1, ry1, rx2, ry2, rx3, ry3, t, vx1, vy1, vx2, vy2, vx3,
vy3;
floatw1 = 5, w2 = 5, w3 = 635, w4 = 465, v1 = 425, v2 = 75, v3 = 550, v4 = 250;
intgd, gm, ch;
void original ();
void triangle (float, float, float, float, float, float);
void rotate (float, float, float);
void main ()
{
 clrscr ();
 cout<< "\n\t\t ***** 2D Geometric Transformations *****";
 cout<< "\n\n Enter the coordinates of triangle (x1,y1,x2,y2,x3, y3): \n";
 cin>> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
 original ();
 closegraph ();
 do
 {
  cout<< "\n\n Choose any one Transformation : ";
  cout<< "\n\t 1.Translation \n\t 2.Rotation \n\t 3.Scaling \n\t 4.Reflection";
  cout<< "\n\t 5.Shearing \n\t 6.Window-Viewport";
  cout<< "\n\n Enter your choice : \t";
  cin>>ch;
  switch (ch)
  {
   case 1:
    cout<< "\n Enter the translation factors (x,y): \t\t";
    cin>> x >> y;
    original ();
    triangle (x1+x, y1+y, x2+x, y2+y, x3+x, y3+y);
    closegraph ();
    break;
   case 2:
    cout<< "\n Enter the angle of rotation : \t\t";
```

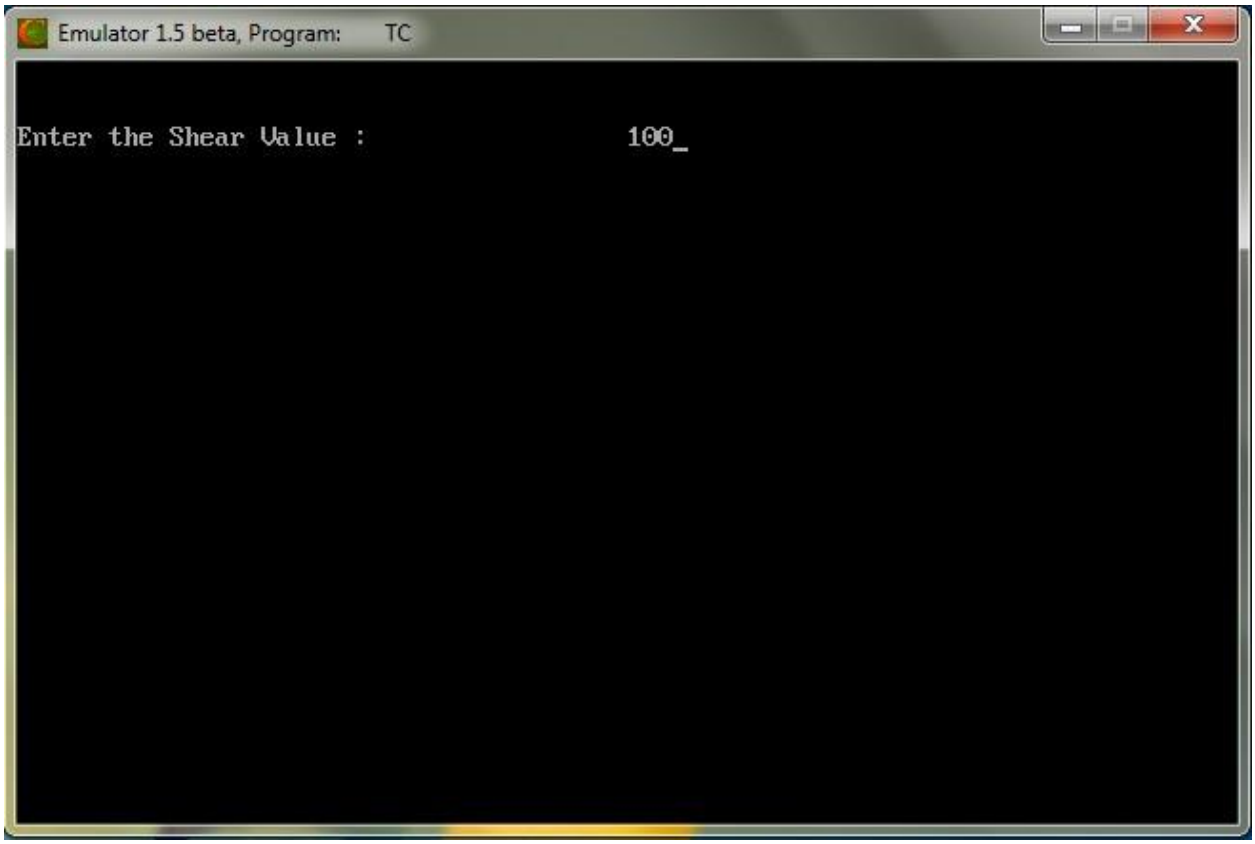
```
    cin>> t;
    cout<< "\n Enter the reference point of rotation (rx,ry) : \t";
    cin>> x >> y;
    original ();
    rotate (t, x, y);
    closegraph ();
    break;

case 3:   cout<< "\n Enter the scaling factors (x,y): \t\t";
   cin>> x >> y;
   original ();
   triangle (x1*x, y1*y, x2*x, y2*y, x3*x, y3*y);
   closegraph ();
   break;
  case 4:
   original ();
   rotate (180, x2, y2);
   closegraph ();
   break;
  case 5:
   cout<< "\n Enter the Shear Value : \t\t";
   cin>> x;
   original ();
   triangle (x1, y1, x2+x, y2, x3, y3);
   closegraph ();
   break;
  case 6:
   initgraph (&gd, &gm, "C:\\TC\\BGI");
   rectangle (w1, w2, w3, w4);
   outtextxy (300, 10, "Window Port");
   triangle (x1, y1, x2, y2, x3, y3);
   x = (v3 - v1) / (w3 - w1);
   y = (v4 - v2) / (w4 - w2);
   vx1 = v1 + floor (((x1 - w1) * x) + 0.5);
   vy1 = v2 + floor (((y1 - w2) * y) + 0.5);
   vx2 = v1 + floor (((x2 - w1) * x) + 0.5);
   vy2 = v2 + floor (((y2 - w2) * y) + 0.5);
   vx3 = v1 + floor (((x3 - w1) * x) + 0.5);
   vy3 = v2 + floor (((y3 - w2) * y) + 0.5);
   rectangle (v1, v2, v3, v4);
   outtextxy (450, 85, "View Port");
```

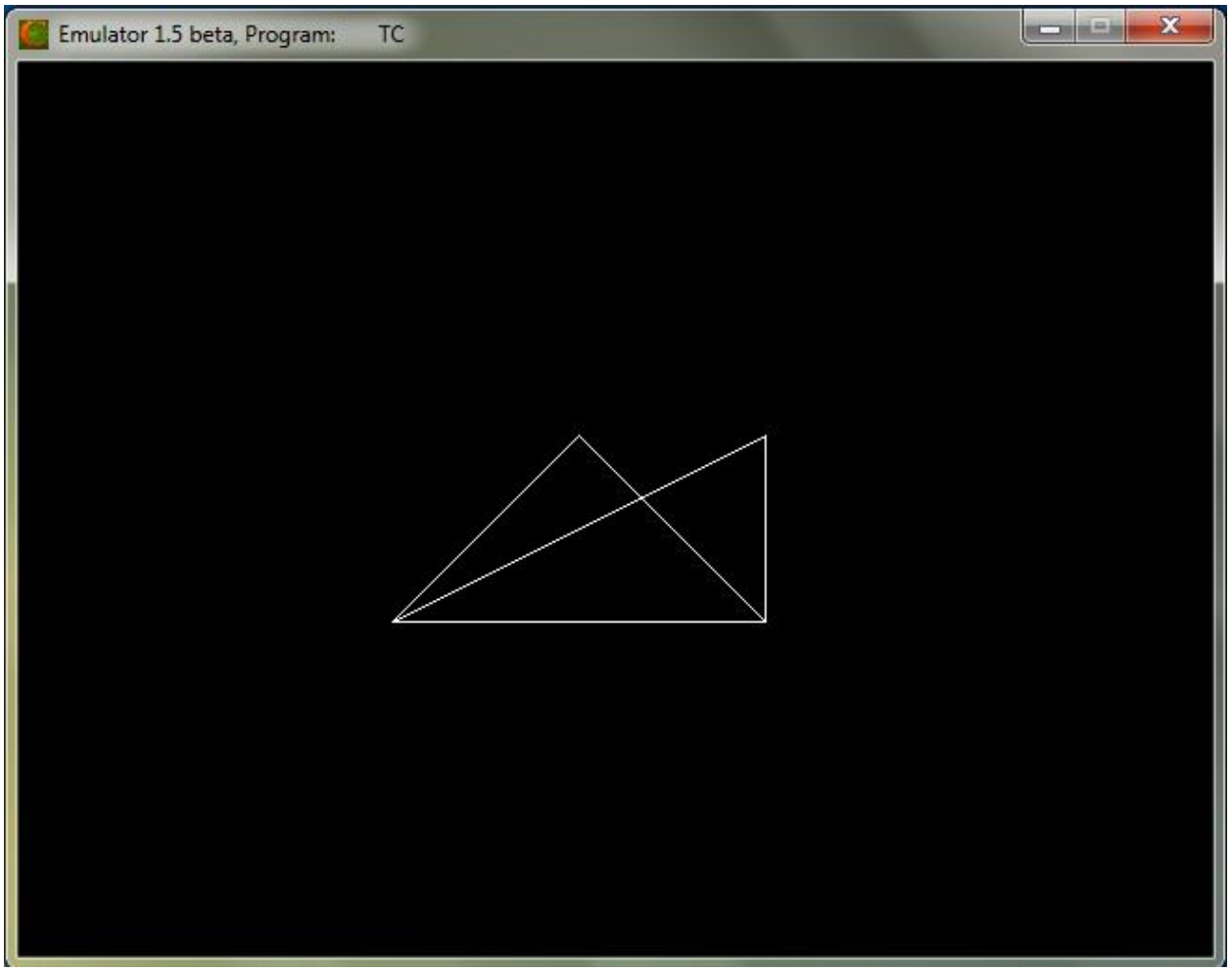
```
   triangle (vx1, vy1, vx2, vy2, vx3, vy3);
   closegraph ();
 }
} while (ch<= 6);
getch ();
void original ()
initgraph (&gd, &gm, "C:\\TC\\BGI");
triangle (x1, y1, x2, y2, x3, y3);
void triangle (float x1, float y1, float x2, float y2, float x3, float y3)
{
 line (x1, y1, x2, y2);
 line (x2, y2, x3, y3);
 line (x3, y3, x1, y1);
 getch ();
}
void rotate (float t, float rx, float ry)
{
 t = t * (3.14 / 180);
 rx1 = rx + (x1 - rx) * cos (t) - (y1 - ry) * sin (t);
 ry1 = ry + (x1 - rx) * sin (t) + (y1 - ry) * cos (t);
 rx2 = rx + (x2 - rx) * cos (t) - (y2 - ry) * sin (t);
 ry2 = ry + (x2 - rx) * sin (t) + (y2 - ry) * cos (t);
 rx3 = rx + (x3 - rx) * cos (t) - (y3 - ry) * sin (t);
 ry3 = ry + (x3 - rx) * sin (t) + (y3 - ry) * cos (t);
 triangle (rx1, ry1, rx2, ry2, rx3, ry3);
}
```

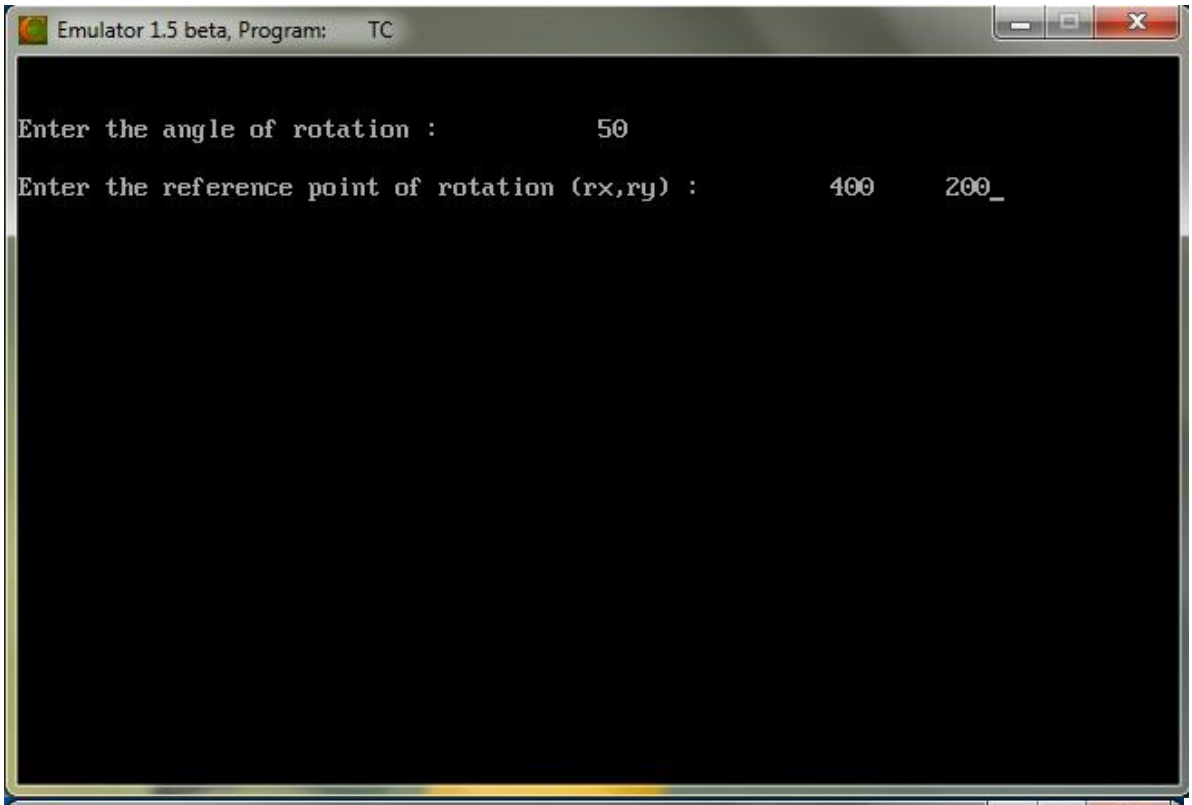
OUTPUT:

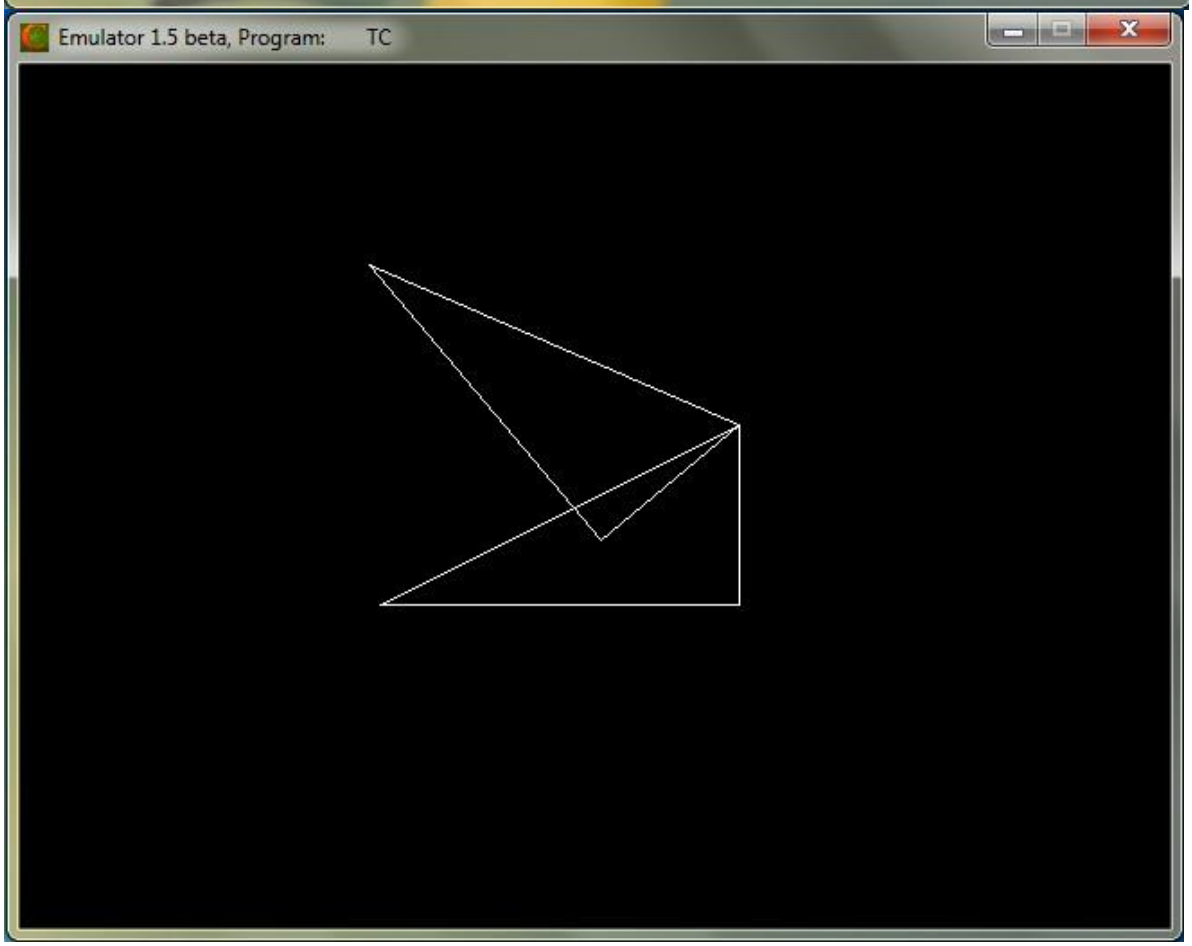

Emulator 1.5 beta, Program: TC
***** Composite 2D Transformations *****
Enter the coordinates of triangle (x1,y1,x2,y2,x3,y3) : 200 300 300
200 400 300

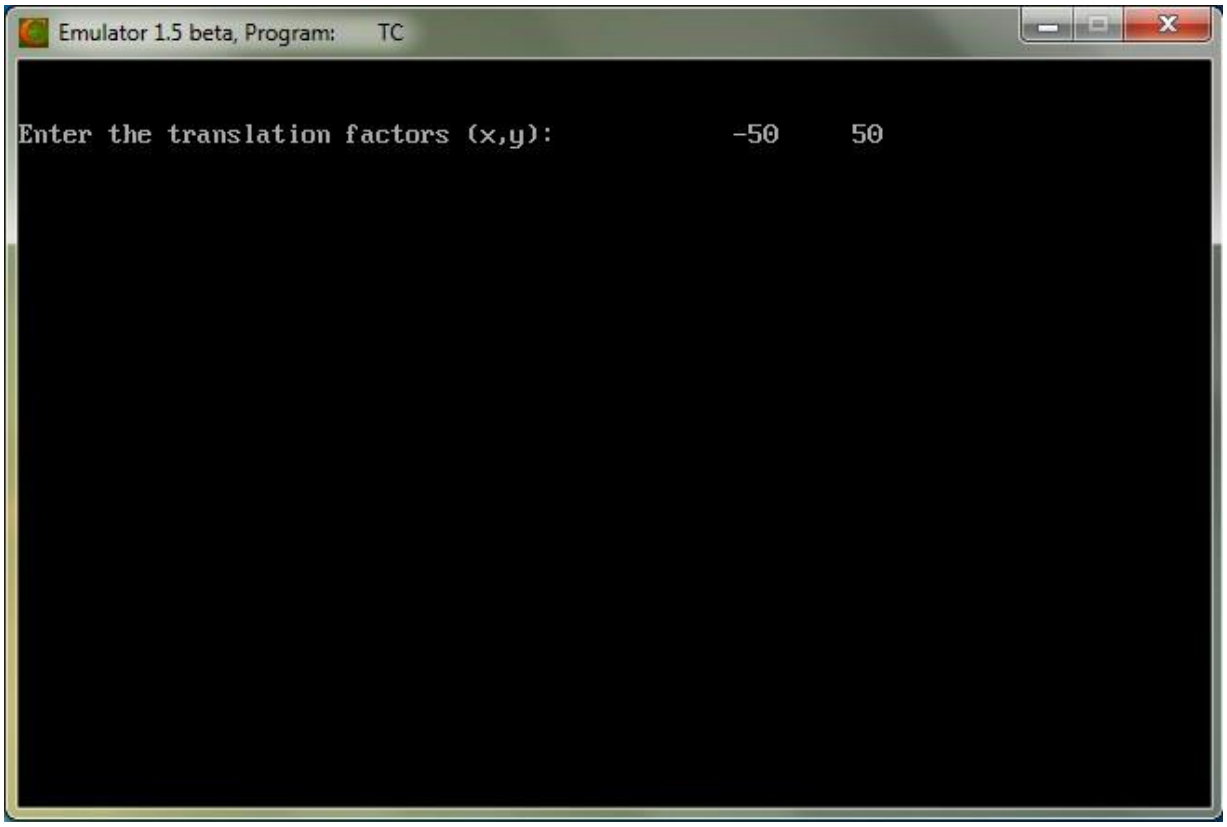

Emulator 1.5 beta, Program: TC

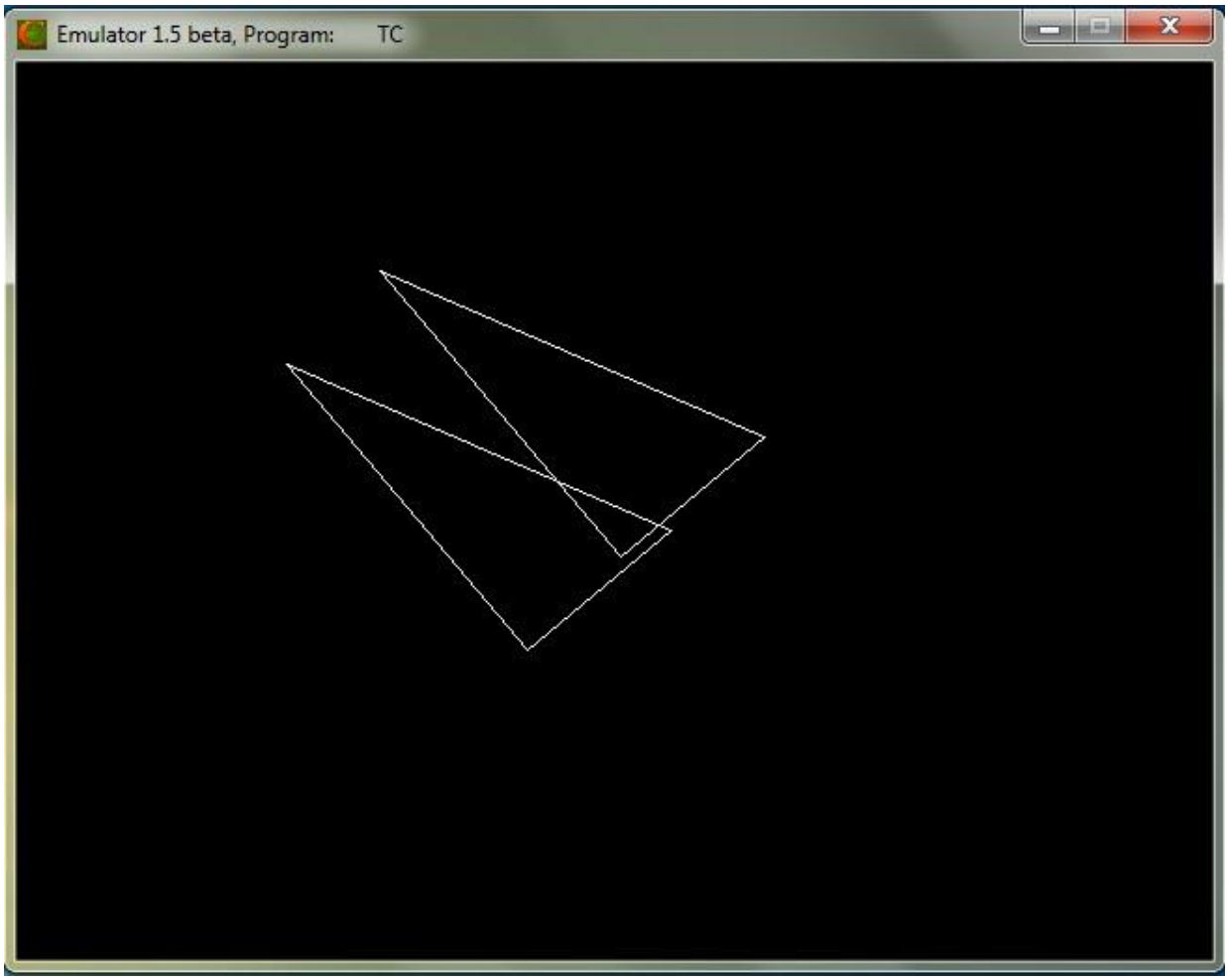

Emulator 1.5 beta, Program: TC
Enter the Shear Value : 100_

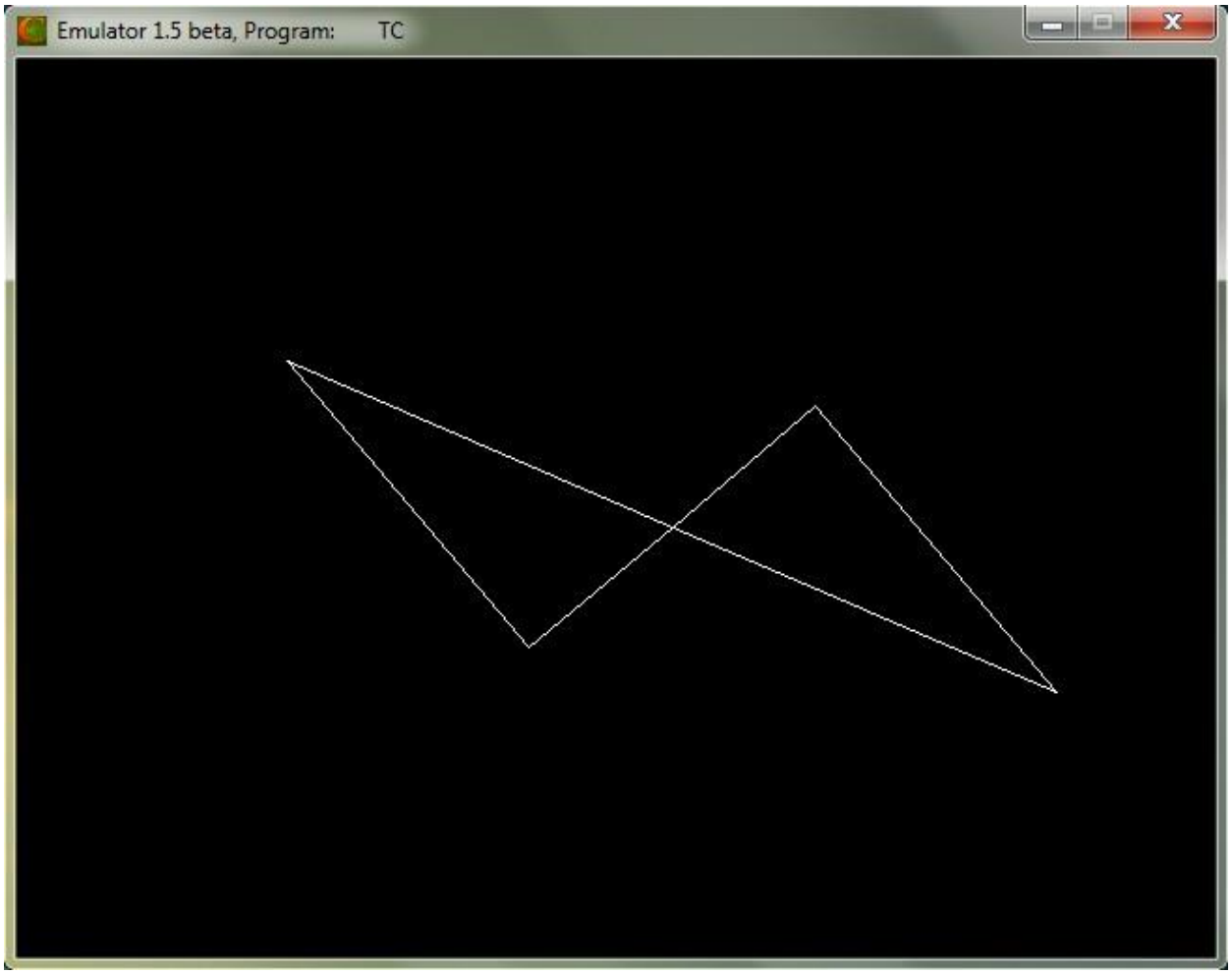

Emulator 1.5 beta, Program: TC

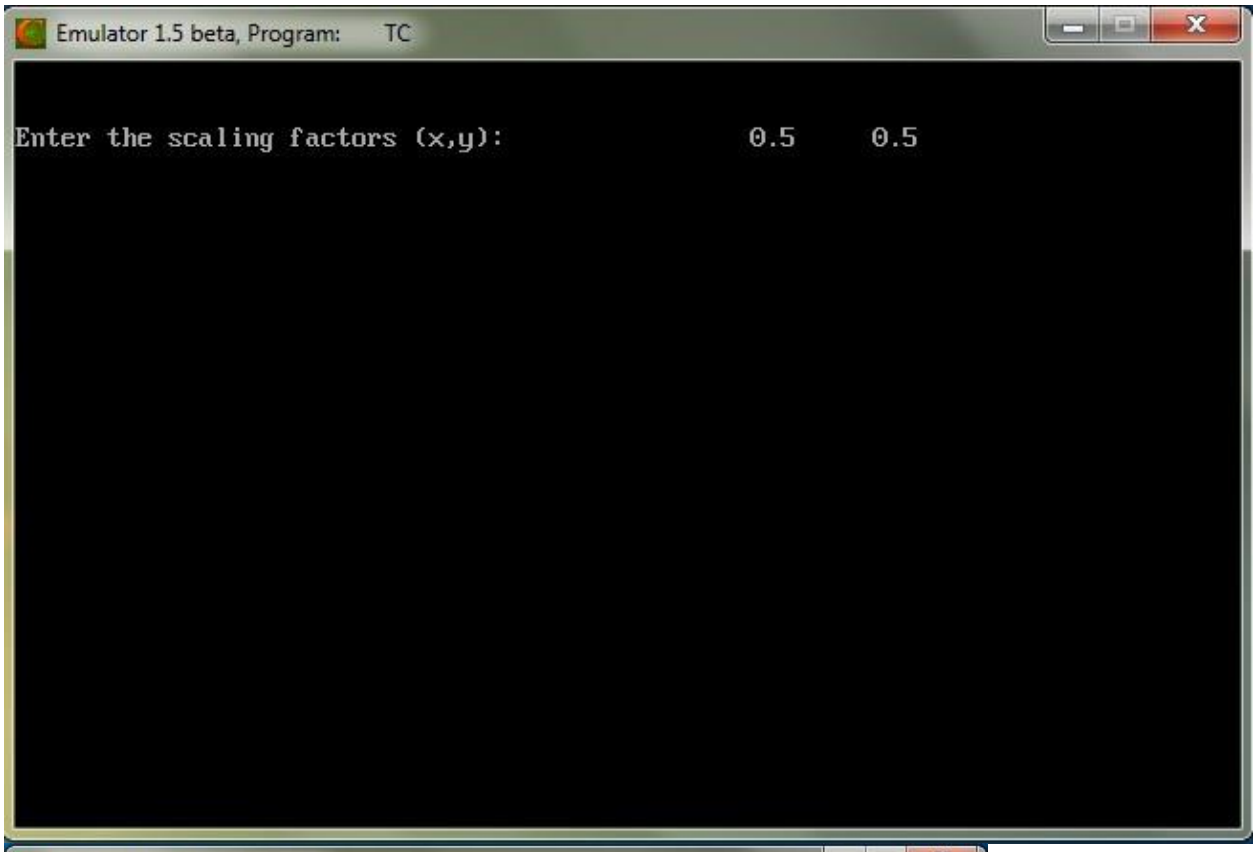

Emulator 1.5 beta, Program: TC
Enter the angle of rotation : 50
Enter the reference point of rotation (rx,ry) : 400 200_

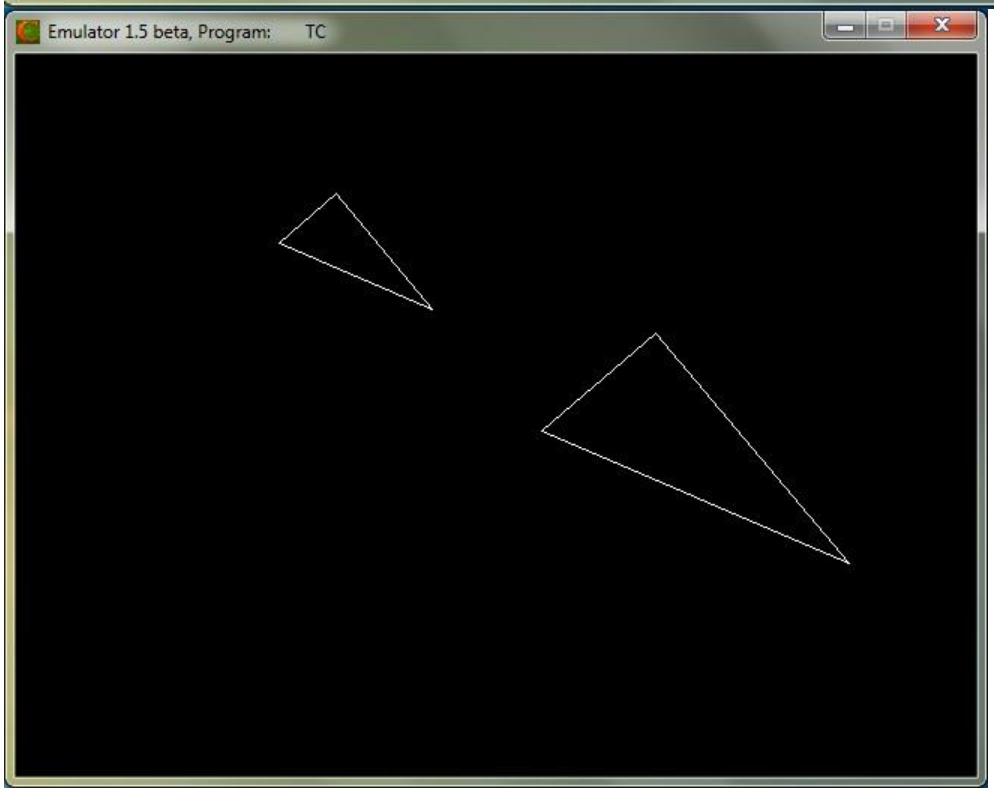

Emulator 1.5 beta, Program: TC

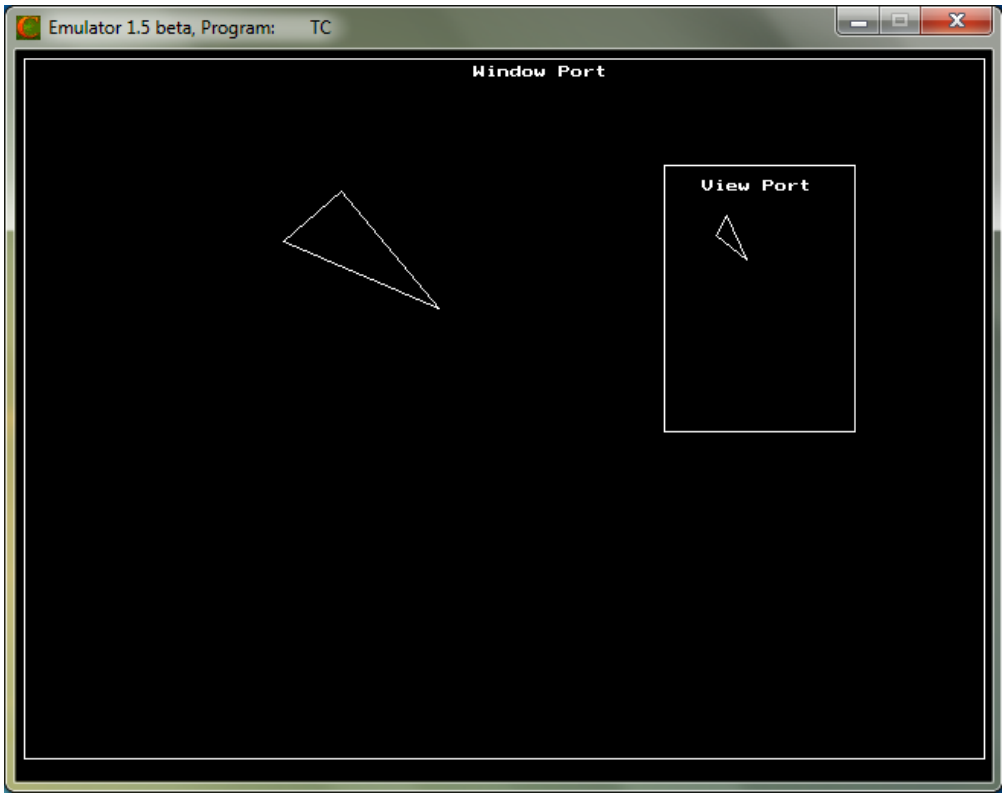

Emulator 1.5 beta, Program: TC
Enter the translation factors (x,y): -50 50


Emulator 1.5 beta, Program: TC


Emulator 1.5 beta, Program: TC


Emulator 1.5 beta, Program: TC
Enter the scaling factors (x,y): 0.5 0.5


Emulator 1.5 beta, Program: TC


Emulator 1.5 beta, Program: TC
Window Port
View Port

RESULT:

Thus the implementation of the 2D Geometric Transformations (translation, rotation, scaling, reflection, shearing, and window-view port) has been implemented and the output was verified.

EX NO:	IMPLEMENTATION OF COMPOSITE TRANSFROMATION
DATE:	

AIM:

To write a c program for implementing composite two dimensional transformation

ALGORITHM:

1. Input the object coordinates
2. Translation
   - **Enter the translation factor tx and ty**
   - **Move the original coordinate position(x,y) to a new portion (x1,y1) ie)x=x+x1;y=y+y1**

3.rotation
   - Enter the radian for rotation angle 0
   - Roatae a point a portion (x,y) through an angle 0 about origin x1=xcos0-ysin0;y1=ycos0+xsin0

4.scaling:
   - Input the scaled factors x and y
   - The transformed coordinates (x1,y1),x1=x-x and y1=y-y

5.Finally display the transformation object after the successive transformation

6.Stop the program

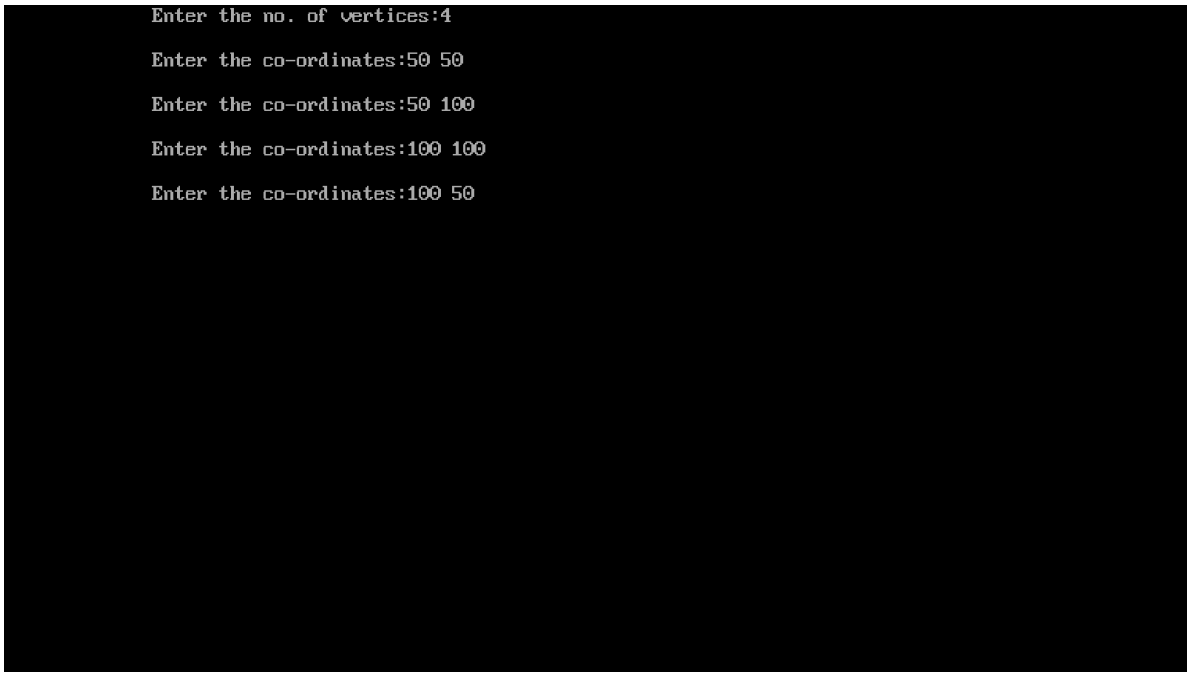
IMPLEMENTATION OF COMPOSITE TRANSFROMATION

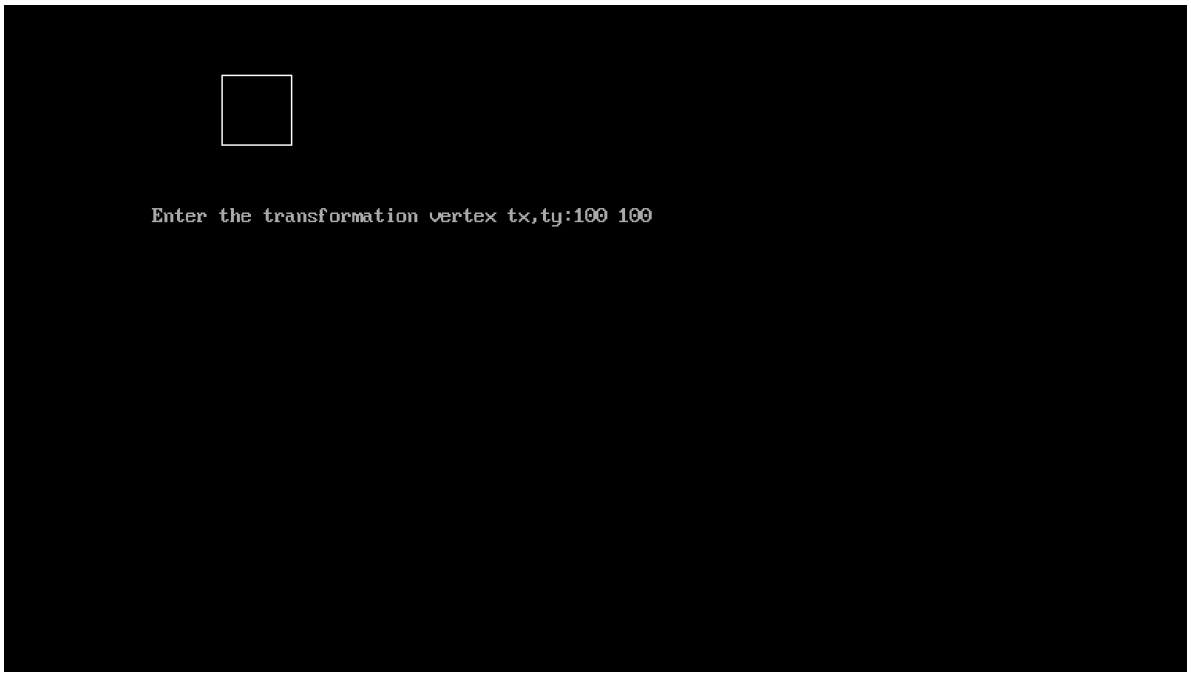
PROGRAM:

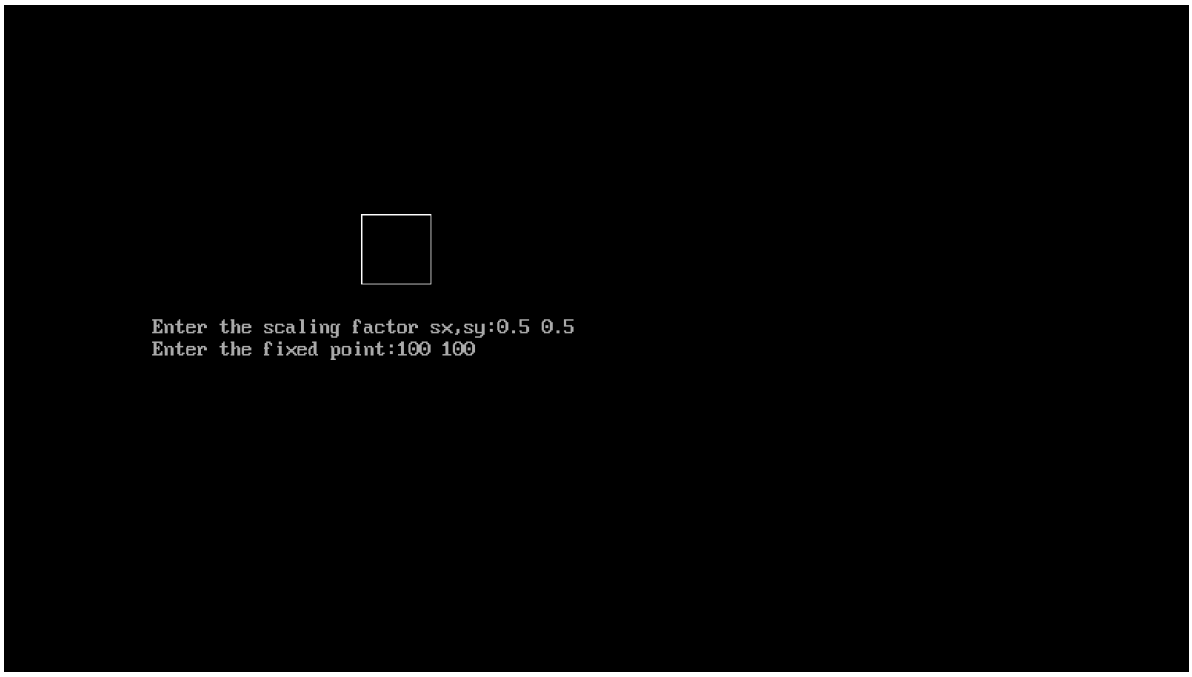
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int n,i,a[20][2],fx,fy,y,tx,ty,temp;
float sx,sy,k;
void input()
{
printf("Enter the no. of vertices:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the co-ordinates:");
scanf("%d%d",&a[i][0],&a[i][1]);
}
}
void output()
{
cleardevice();
for(i=0;i<n-1;i++)
{
line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
}
line(a[i][0],a[i][1],a[0][0],a[0][1]);
}
void rotation()
{
output();
printf("Enter the rotating angle:");
scanf("%d",&y);
printf("Enter the pivot point:");
scanf("%d%d",&fx,&fy);
k=(y*3.14)/180;
for(i=0;i<=n;i++)
{
temp=a[i][0]-a[i][1]*sin(k);
a[i][1]=a[i][0]*sin(k)+a[i][1];
a[i][0]=temp;
```

```
}
output();
}
void scaling()
{
output();
printf("\n\n\n\nEnter the scaling factor sx,sy:");
scanf("%f%f",&sx,&sy);
printf("Enter the fixed point:");
scanf("%d%d",&fx,&fy);
for(i=0;i<=n;i++)
{
a[i][0]=a[i][0]*sx+fy*(1-sx);
a[i][1]=a[i][1]*sy+fy*(1-sy);
}
rotation();
}
void composite()
{
output();
printf("Enter the transformation vertex tx,ty:");
scanf("%d%d",&tx,&ty);
for(i=0;i<=n;i++)
{
a[i][0]=a[i][0]+tx;
a[i][1]=a[i][1]+ty;
}
scaling();
}
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c://turboc3//bgi");
input();
composite();
getch();
}
```

OUTPUT:


Enter the no. of vertices:4
Enter the co-ordinates:50 50
Enter the co-ordinates:50 100
Enter the co-ordinates:100 100
Enter the co-ordinates:100 50


Enter the transformation vertex tx,ty:100 100


Enter the scaling factor sx,sy:0.5 0.5
Enter the fixed point:100 100


Enter the rotating angle:30
Enter the pivot point:100 100

RESULT:

Thus, the implementation of composite transformation is executed successfully

EX NO:	COHEN SUTHERLAND LINE CLIPPING
DATE:	ALGORITHM

AIM:

To write a c program top implement cohen Sutherland line clipping algorithm.

ALGORITHM:

1. Start the program
2. Declare the variables and initialize graphics driver
3. Get window coordinates and end coordinates of the line from the user
4. Every line endpoint is assigned a code that identified the location of the point relative to the boundaries of the clipping square
5. Check whether line ties inside the window then it is entirely drawn
6. Check whether the line ties inside the window then it is entirely clipped(not visible)
7. Otherwise,check whether the line intersects window by using following steps
   - Calculate differences between end point and clip boundaries
   - Detyermine ther intersection point and how much of line is to be discarded

8.starting from leftmost bit,each bit of code is set or true(1) or false(0) according to the scheme bit1(end point is above window)=sign(y-ymax);bit2(endpoint is below window)=sign(ymin-y)

9.display the corresponding output of cluipped line

10.stop the driver

11. stop the program

4.COHEN-SUTHERLAND LINE CLIPING ALGORITHM

PROGRAM:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
typedef unsigned int outcode;
enum{TOP=0x1,BOTTOM=0x2,RIGHT=0x4,LEFT=0x8};
void lineclip(x0,y0,x1,y1,xwmin,ywmin,xwmax,ywmax)
float x0,y0,x1,y1,xwmin,ywmin,xwmax,ywmax;
{
int gd,gm,accept=0,done=0;
outcode code0,code1,codeout;
code0=calcode(x0,y0,xwmin,ywmin,xwmax,ywmax);
code1=calcode(x1,y1,xwmin,ywmin,xwmax,ywmax);
do
{
if(!(code0|code1))
{
accept=1;
done=1;
}
else if(code0 &code1)done=1;
else
```

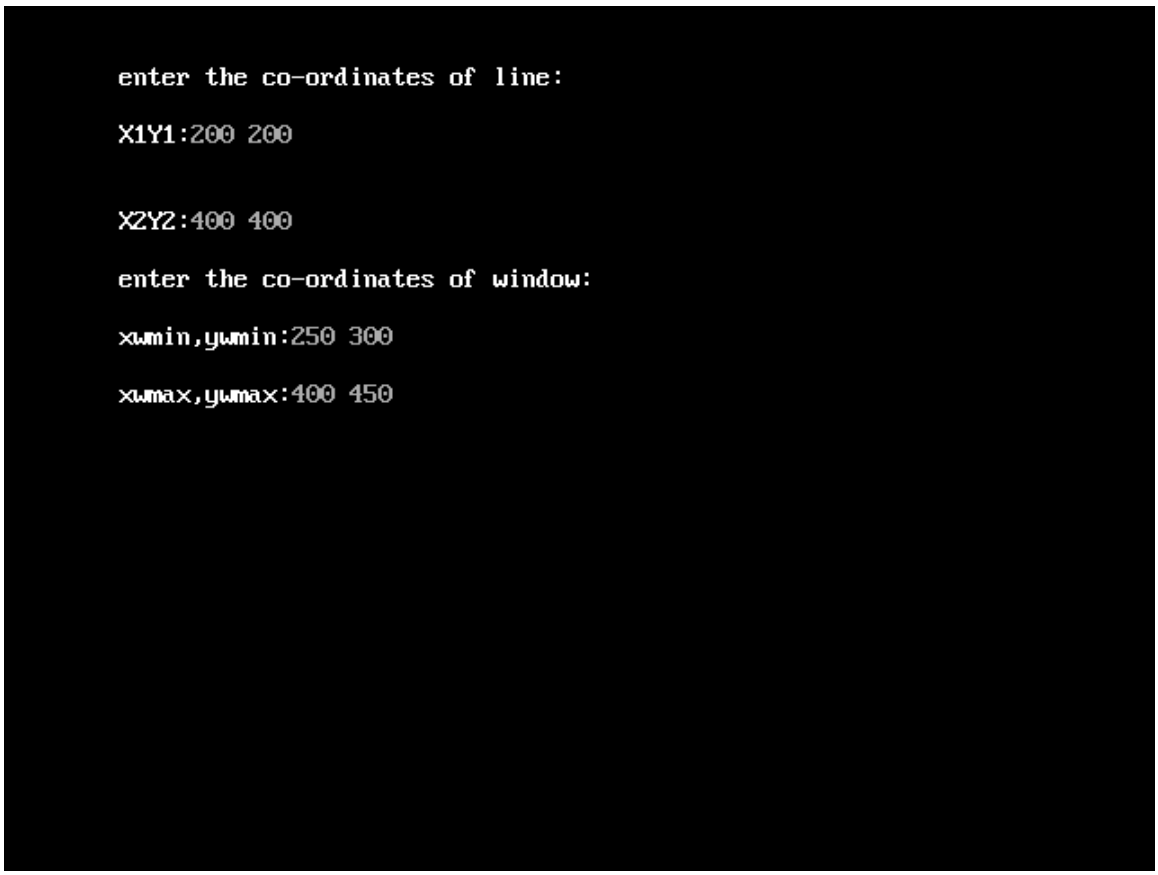
```
{
float x,y;
codeout=code0?code0:code1;
if(codeout&TOP)
{
x=x0+(x1-x0)*(ywmax-y0)/(y1-y0);
y=ywmax;
}
else if(codeout&BOTTOM)
{
x=x0+(x1-x0)*(ywmin-y0)/(y1-y0);
y=ywmin;
}
else if(codeout&RIGHT)
{
y=y0+(y1-y0)*(xwmax-x0)/(x1-x0);
x=xwmax;
}
else
{
y=y0+(y1-y0)*(xwmin-x0)/(x1-x0);
x=xwmin;
}
```

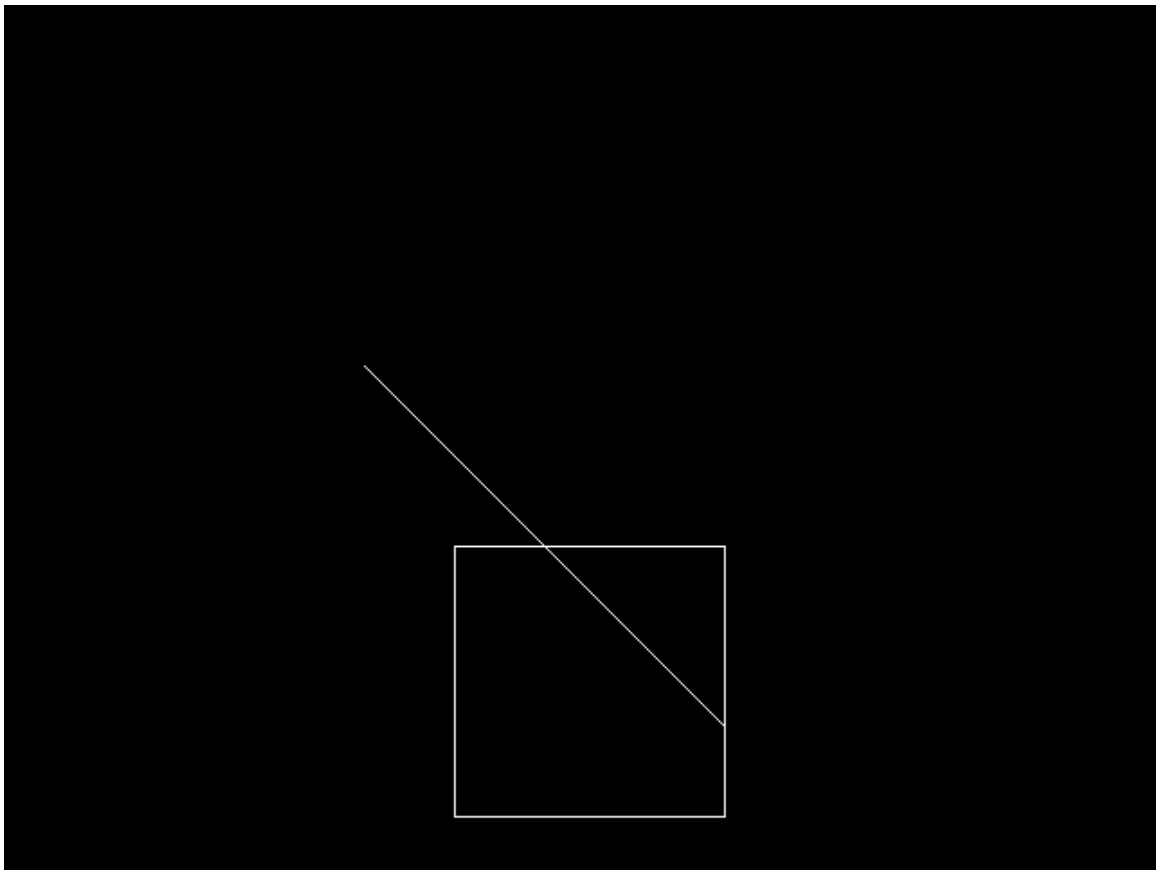
```
if(codeout==code0)
{
x0=x;
y0=y;
code0=calcode(x0,y0,xwmin,ywmin,xwmax,ywmax);
}
else
{
x1=x;
y1=y;
code1=calcode(x1,y1,xwmin,ywmin,xwmax,ywmax);
}
}}
while(done==0);
if(accept)line(x0,y0,x1,y1);
rectangle(xwmin,ywmin,xwmax,ywmax);
getch();
return 0;
}
int calcode(x,y,xwmin,ywmin,xwmax,ywmax)
float x,y,xwmin,xwmax,ywmin,ywmax;
{
int code=0;
```

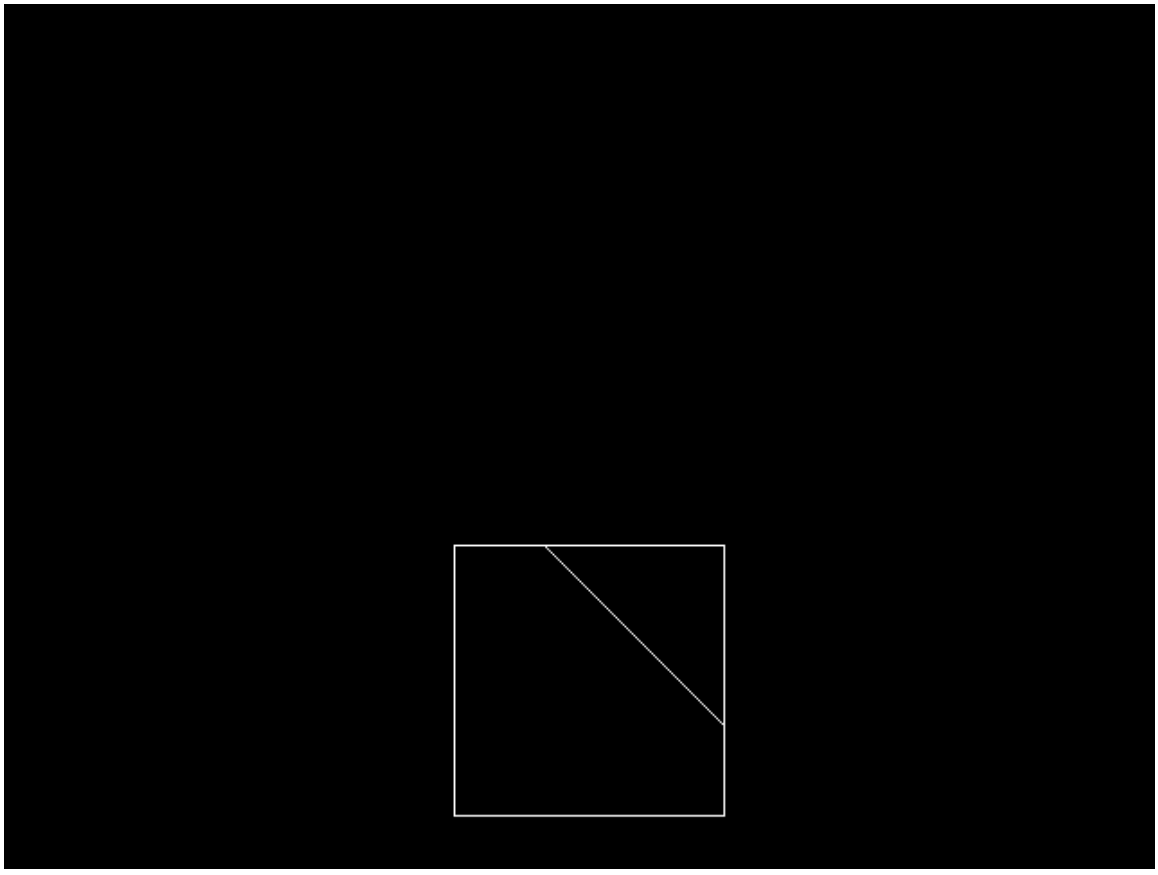
```
if(y>ywmax)
code|=TOP;
else if(y<ywmin)
code|=BOTTOM;
else if(x>xwmax)
code|=RIGHT;
else if(x<xwmin)
code|=LEFT;
return(code);
}
void main()
{
float x2,y2,x1,y1,xwmin,ywmin,xwmax,ywmax;
int gd=DETECT,gm;
clrscr();
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
printf("\n\n\tenter the co-ordinates of line:");
printf("\n\n\tX1Y1:");
scanf("%f%f",&x1,&y1);
printf("\n\n\tX2Y2:");
scanf("%f%f",&x2,&y2);
printf("\n\tenter the co-ordinates of window:\n");
printf("\n\txwmin,ywmin:");
```

```
scanf("%f%f",&xwmin,&ywmin);
printf("\n\txwmax,ywmax:");
scanf("%f%f",&xwmax,&ywmax);
cleardevice();
line(x1,y1,x2,y2);
rectangle(xwmin,ywmin,xwmax,ywmax);
getch();
cleardevice();
lineclip(x1,y1,x2,y2,xwmin,ywmin,xwmax,ywmax);
getch();
closegraph();
}
```

OUTPUT:


enter the co-ordinates of line:
X1Y1:200 200
XZY2:400 400
enter the co-ordinates of window:
xwmin,ywmin:250 300
xwmax,ywmax:400 450





RESULT:

Thus, the cohen Sutherland line clipping algorithm is executed successfully

EX NO:	3D TRANSFORMATION-TRANSLATION, ROTATION
DATE:	AND SCALING

AIM:

To write a c program to implement 3d transformation

ALGORITHM:

1. Start the program
2. Initialize the variable
3. Translation
    - A point is transferred from position (x,y,x) to position (x1.y1.x1) with matrix operation x1100|xxy|xo}0|yyx|00|xx|0011
    - The value are x1=x+1x ; y1=y+1y and x1=x+|x
    - Displaythe transformed object
4. Rotation
    - Coordinate are rotation x1=xcosϴ-ysinϴ; y1=xsinϴ+ycosϴ;z1=x
    - Y-axis rotation: z1=zcosϴ-xsinϴ; x1=zsinϴ+xcosϴ; y1=y
    - X axis rotation: y1=ycosϴ-zsinϴ; z1=ysinϴ+zcosϴ; x1=x
    - Display the rotate object

5.scaling

- Translate the fixed point to the origin
- Scale the object
- Translate the fixed point back to the orginal position
- Display the scaled object

6.stop the program

5.3D TRANSFORMATION-TRANSLATION,ROTATION AND SCALING

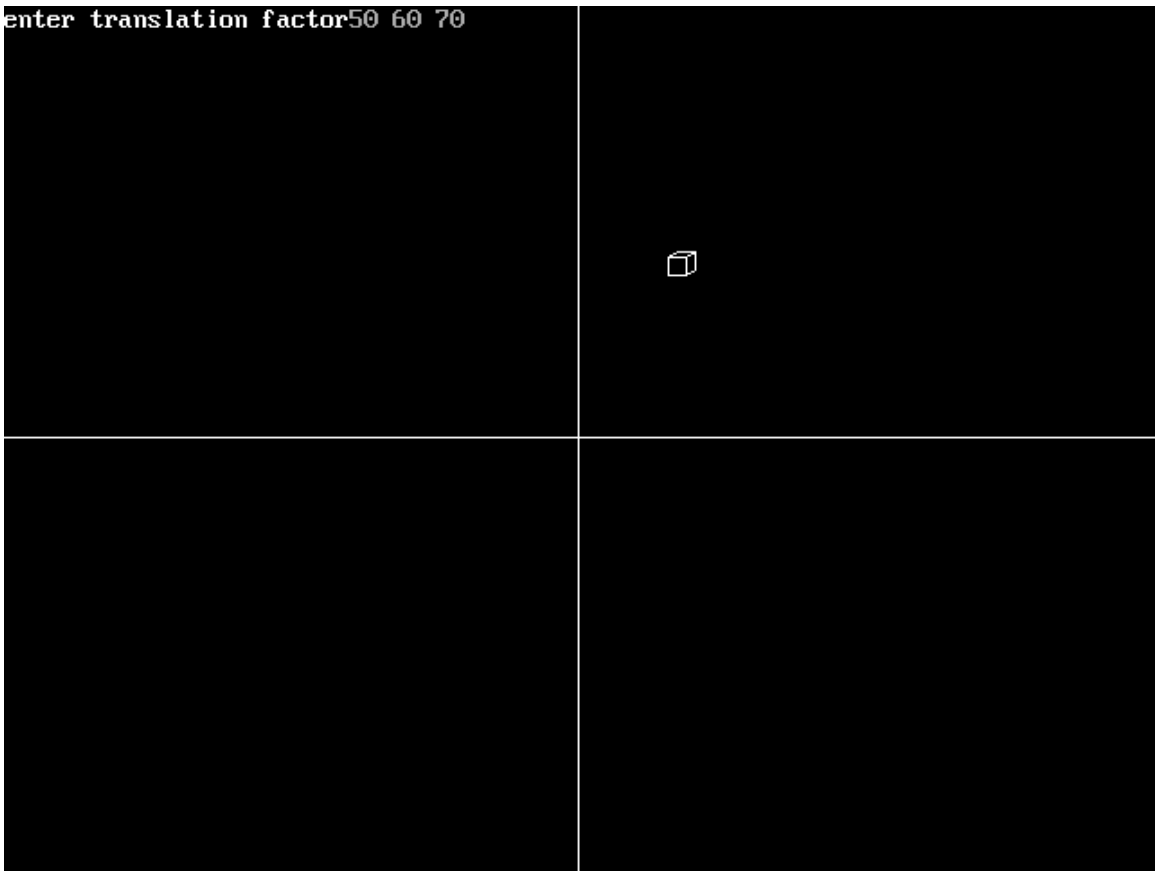
PROGRAM:

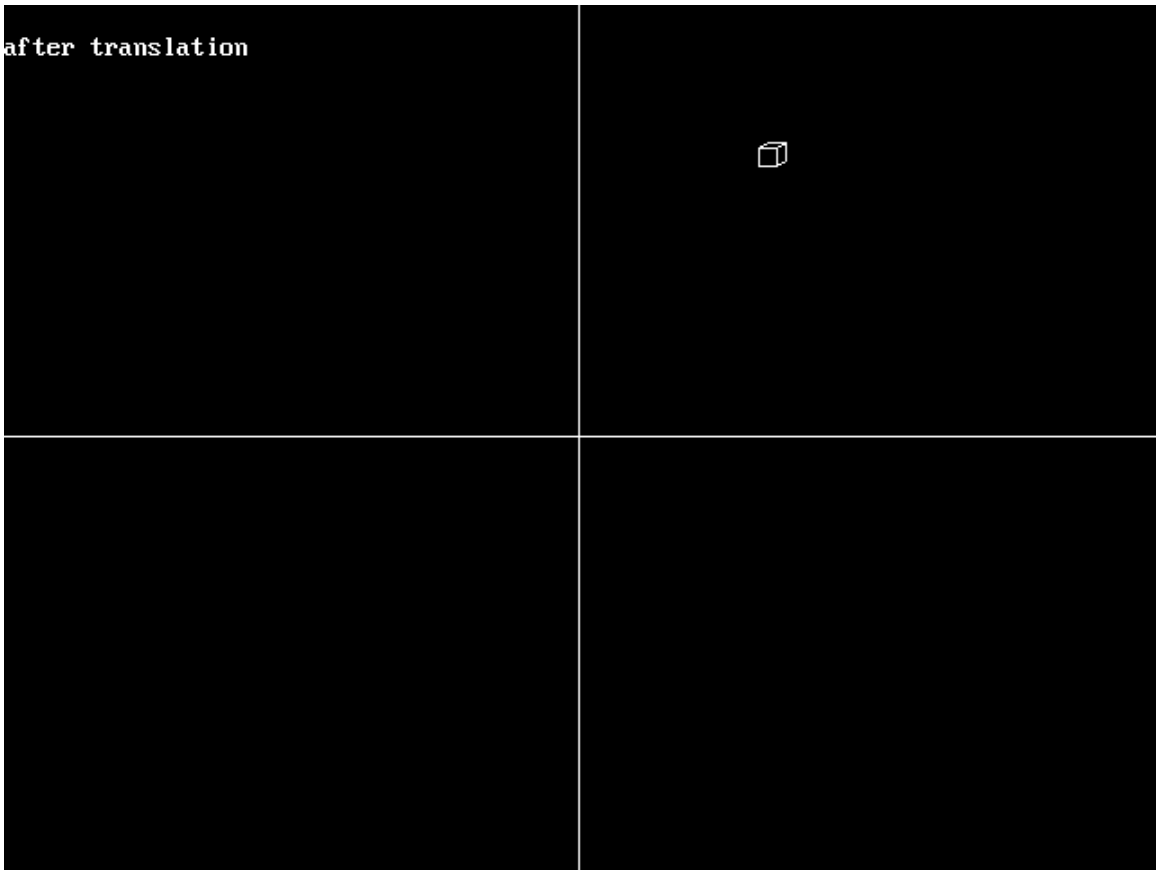
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int gd,gm,x,y,z,x1,x2,y1,y2,o;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
setfillstyle(0,getmaxcolor());
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
```

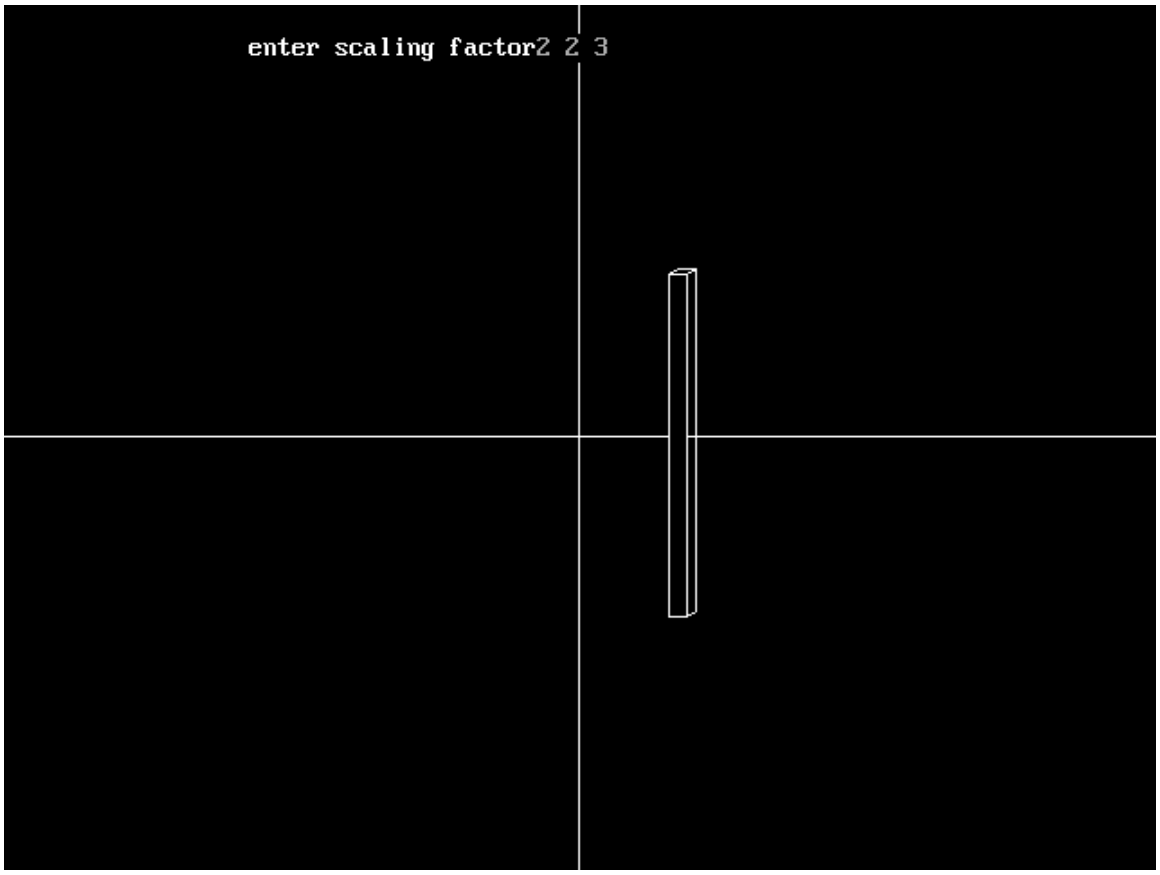
```
midy=maxy/2;
axis();
bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("enter translation factor");
scanf("%d%d%d",&x,&y,&z);
axis();
printf("after translation");
bar3d(midx+(x+50),midy-(y+100),midx+x+60,midy-(y+90),5,1);
axis();
bar3d(midx+50,midy+100,midx+60,midy-90,5,1);
printf("enter scaling factor");
scanf("%d%d%d",&x,&y,&z);
axis();
printf("after scaling");
bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);
axis();
bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("enter the rotating angle");
scanf("%d",&o);
x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
y1=50*cos(o*3.14/180)+100*sin(o*3.14/180);
x2=60*sin(o*3.14/180)-90*cos(o*3.14/180);
y2=60*cos(o*3.14/180)+90*cos(o*3.14/180);
```

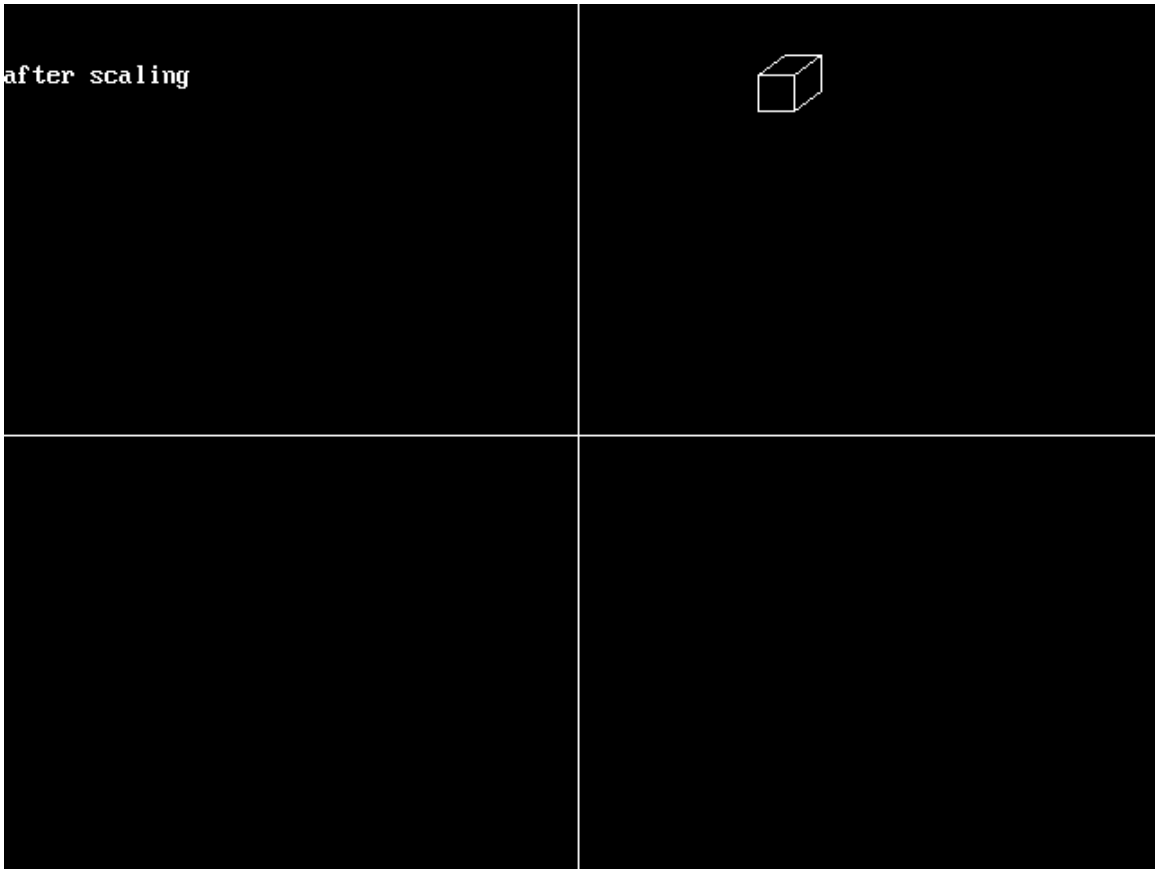
```
axis();
printf("\nafter rotation about z axis");
bar3d(midx+x1,midy-y1,midx+x2,midy-y2,5,1);
axis();
 printf("\nafter rotation about x axis");
 bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);
 axis();
 printf("\nafter rotation about Y axis");
 bar3d(midx+x1,midy-100,midx+x2,midy-90,5,1);
 getch();
 closegraph();
 }
```

OUTPUT:




after translation


enter scaling factor2 2 3


after scaling

enter the rotating angle35

after rotation about z axis

after rotation about x axis

after rotation about Y axis

EX NO:	**3D PROJECTIONS - PARALLEL AND PERSPECTIVE**
DATE:	

AIM:

To create the 3D Scene in C++.

ALGORITHM:

1. Set the background color.
2. Find the coordinates to draw a 3D scene.
3. Plot that coordinates using pre-defined functions like line( ), rectangle( ), fillellipse(4. Set the color of the objects by setcolor( ).

7.3D-SCENES

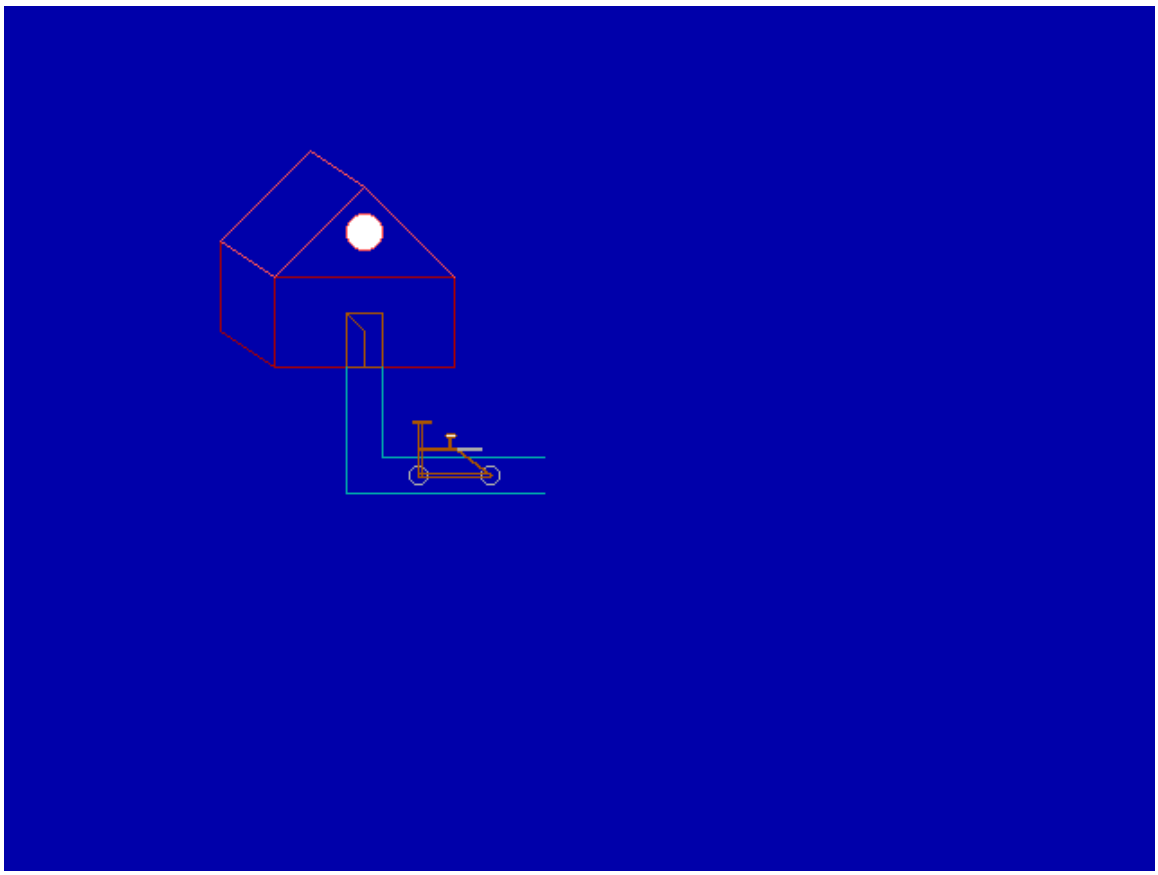
PROGRAM:

```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
int gd,gm;
void main()
{
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
setbkcolor(BLUE);
setcolor(RED);
rectangle(150,150,250,200);
line(150,200,120,180);
line(120,180,120,130);
setcolor(LIGHTRED);
line(120,130,150,150);
line(150,150,200,100);
line(200,100,250,150);
fillellipse(200,125,10,10);
line(120,130,170,80);
line(170,80,200,100);
setcolor(BROWN);
```

```
rectangle(190,170,210,200);
line(190,170,200,180);
line(200,180,200,200);
setcolor(CYAN);
line(190,200,190,270);
line(210,200,210,250);
line(190,270,300,270);
line(210,250,300,250);
setcolor(LIGHTGRAY);
circle(230,260,5);
circle(270,260,5);
setcolor(BROWN);
line(230,259,270,259);
line(230,261,270,261);
line(230,230,230,261);
line(232,230,232,260);
line(227,230,237,230);
line(227,231,237,231);
line(270,260,250,245);
line(271,260,251,245);
line(250,245,230,245);
line(250,246,230,245);
line(250,246,230,246);
```

```
line(248,245,248,240);
line(247,245,247,240);
fillellipse(248,238,3,1);
setcolor(LIGHTGRAY);
line(252,245,265,245);
line(252,246,265,246);
getch();}
```

OUTPUT:



RESULT:
Thus the 3D scene has been created and the output was verified.

EX NO:	3D SCENES
DATE:	

AIM:

To write a program to implement the following 3D projections:

a. Parallel Projection

b. Perspective Projection

ALGORITHM:

1. Get the coordinate to draw the cube.
2. Print the menu : 1. Parallel Projection 2. Perspective Projection
3. If user choose Parallel Projection then find the coordinates of following views a them:

a) Side View

b) Front View

c) Top View

4. If user choose Perspective Projection then simply draw the cube using bar3d(

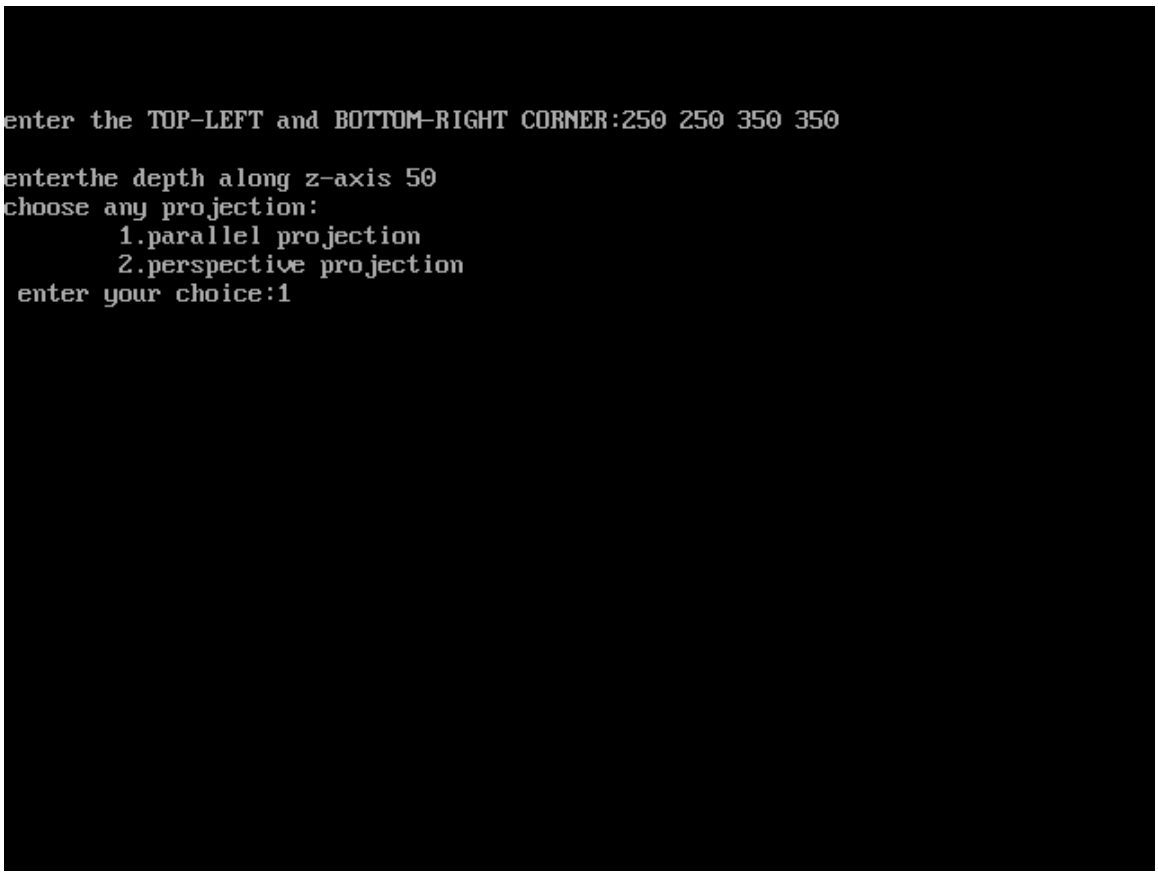
7.3D PROJECTION-PARALLEL AND PERSPECTIVE

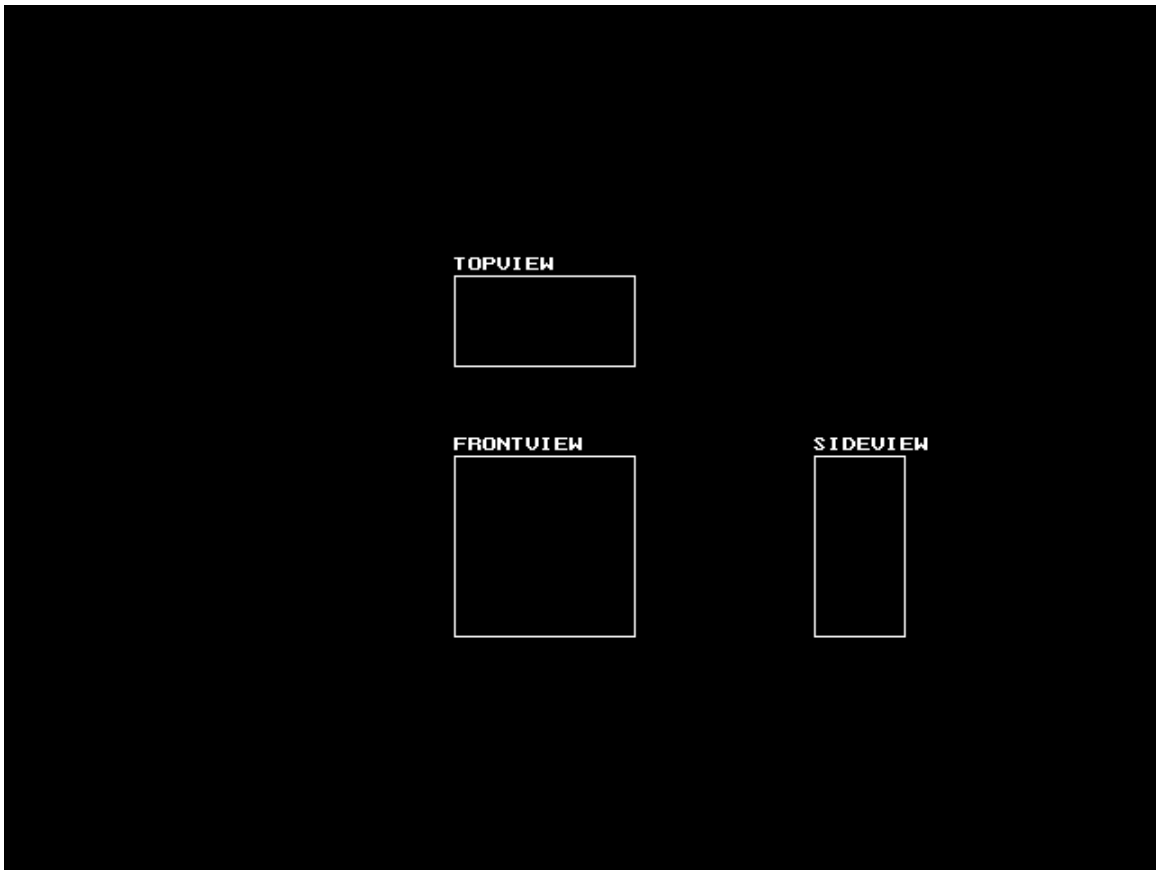
PROGRAM:

```
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
int gd,gm,x1,y1,x2,y2,dep,ch;
void main()
{
cout<<"\nenter the TOP-LEFT and BOTTOM-RIGHT CORNER:";
cin>>x1>>y1>>x2>>y2;
cout<<"\nenterthe depth along z-axis";
cin>>dep;
do
{
cout<<"choose any projection:\n\t1.parallel projection\n\t2.perspective projection\n
enter your choice:";
cin>>ch;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
switch(ch)
{
case 1:
rectangle(x2+100,y1,x2+100+dep,y2);
outtextxy(x2+100,y1-10,"SIDEVIEW");
rectangle(x1,y1,x2,y2);
```

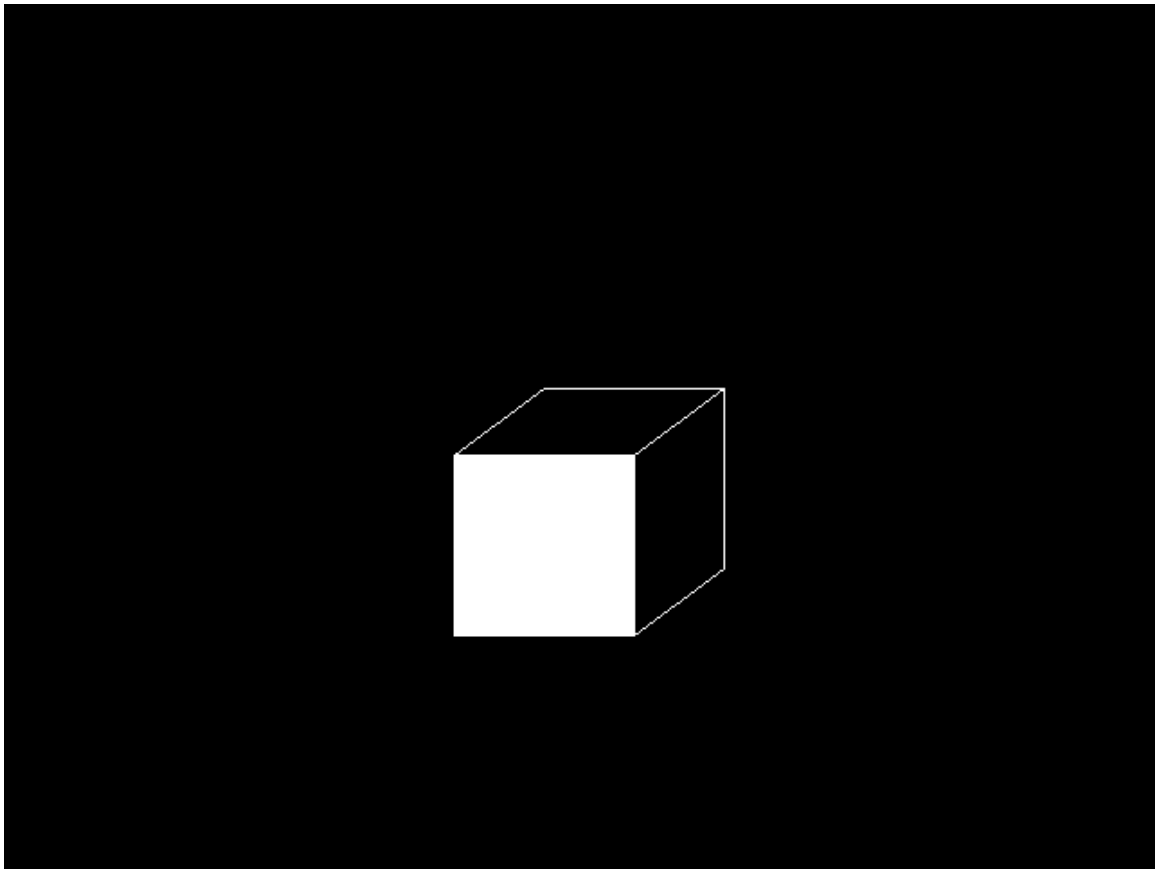
```
outtextxy(x1,y1-10,"FRONTVIEW");
rectangle(x1,y1-(y2-y1),x2,x1+dep-(y2-y1));
outtextxy(x1,y1-(y2-y1)-10,"TOPVIEW");
getch();
closegraph();
break;
case 2:
bar3d(x1,y1,x2,y2,dep,1);
getch();
closegraph();
break;
}
}while(ch<3);
}
```

OUTPUT:


enter the TOP-LEFT and BOTTOM-RIGHT CORNER:250 250 350 350
enterthe depth along z-axis 50
choose any projection:
1.parallel projection
2.perspective projection
enter your choice:1


TOPVIEW
FRONTVIEW
SIDEVIEW

```
choose any projection:
       1.parallel projection
       2.perspective projection
 enter your choice: 2
```



RESULT:

Thus the implementations of 3D projections (Parallel and Perspective) has been created and the output was verified.

EX NO:	IMAGE EDITING AND MANIPULATION
DATE:	

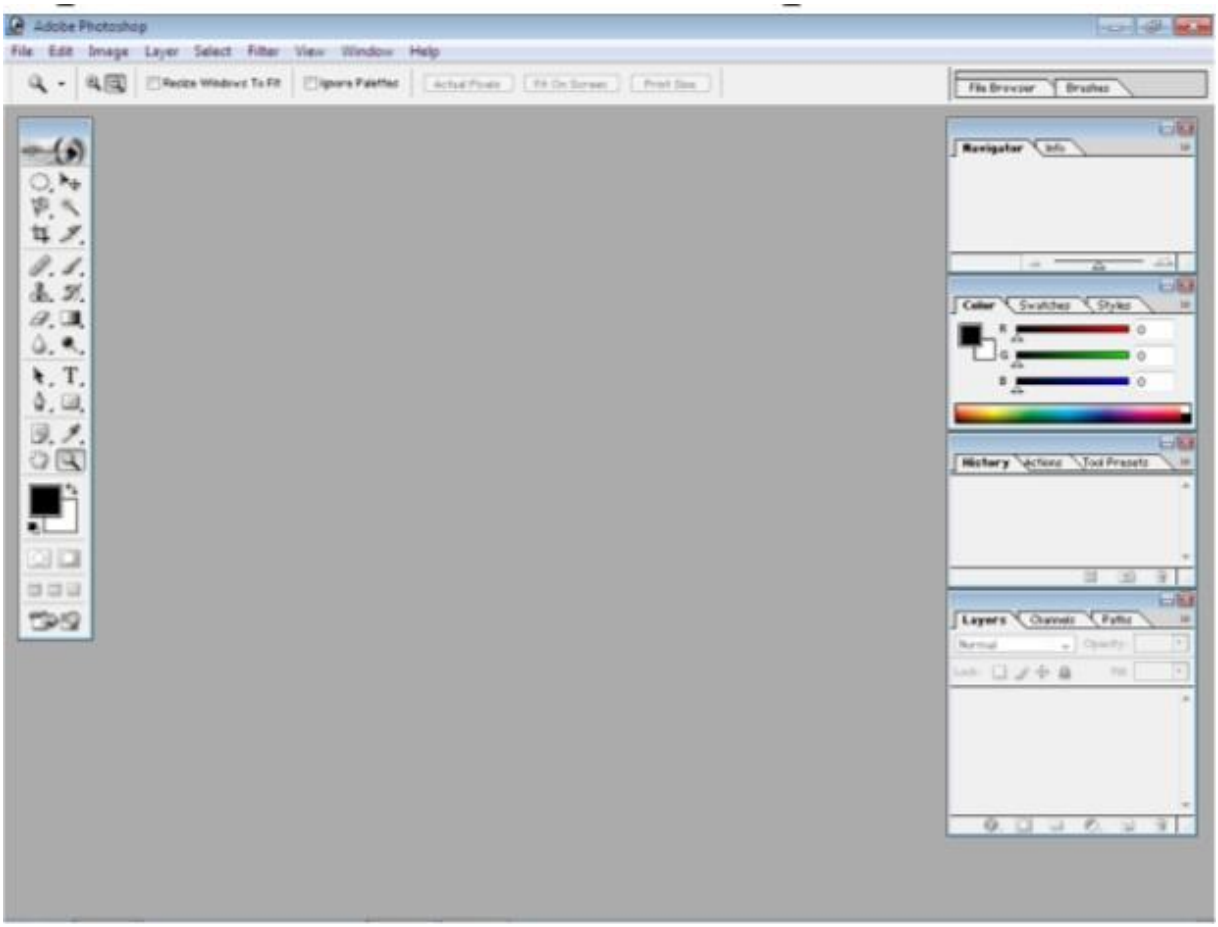
AIM:

To perform the following operations, (i) Basic Operations on image using Adobe Photoshop 7.0 (ii) Creating gif animated images in PhotoScape (iii) To optimize the image in Adobe Photoshop 7.0

PROCEDURE:

1. To perform the basic operations on image using Adobe Photoshop 7.0

i. Open the Adobe Photoshop 7.0.



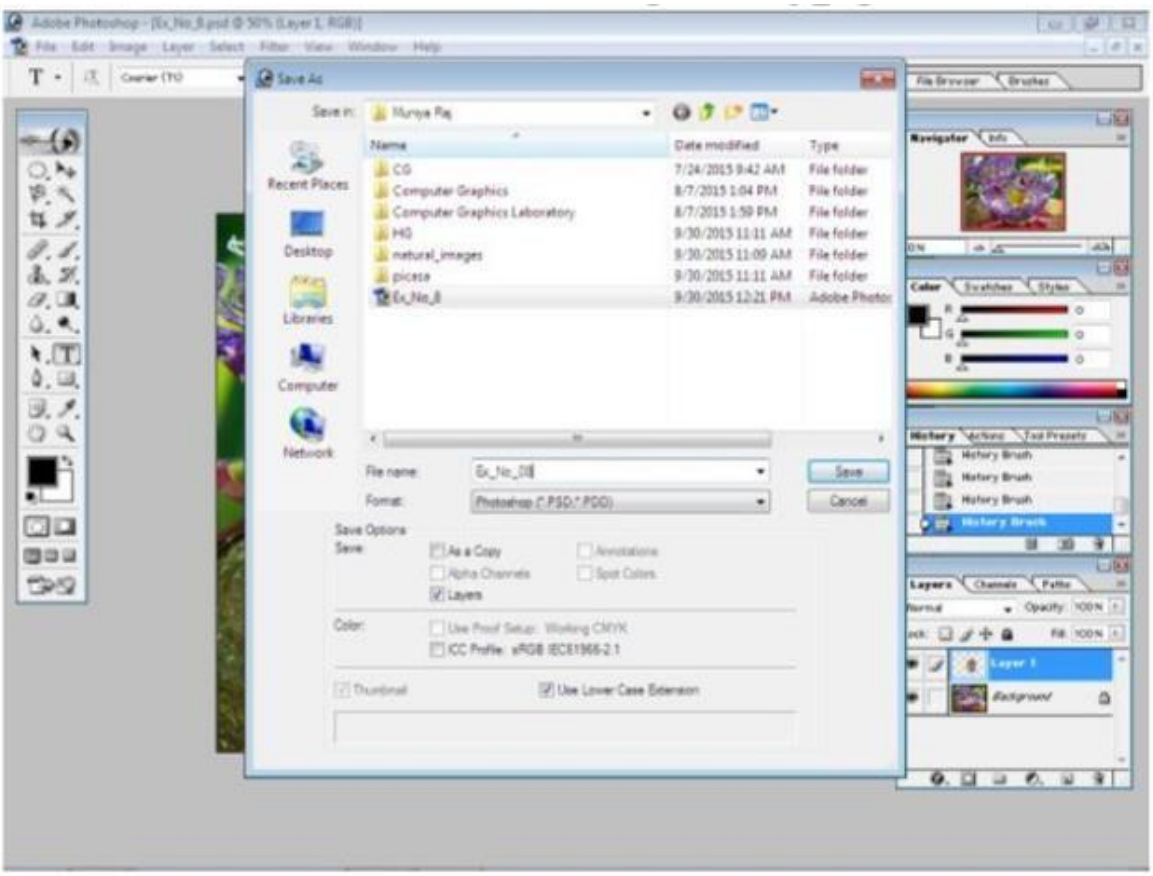
ii. Open the images.

iii. Select the particular portion of the image by Elliptical Marquee Tool.

iv. Move that selected portion to another image by using Move Tool.

v. Eliminate the unwanted portion of the image by using History Brush Tool.

vi. Save the modified image in jpg format



2. To Create the gif animated images in PhotoScape

i. Draw the images in different scenes to animate. Save all the images in jpg format.

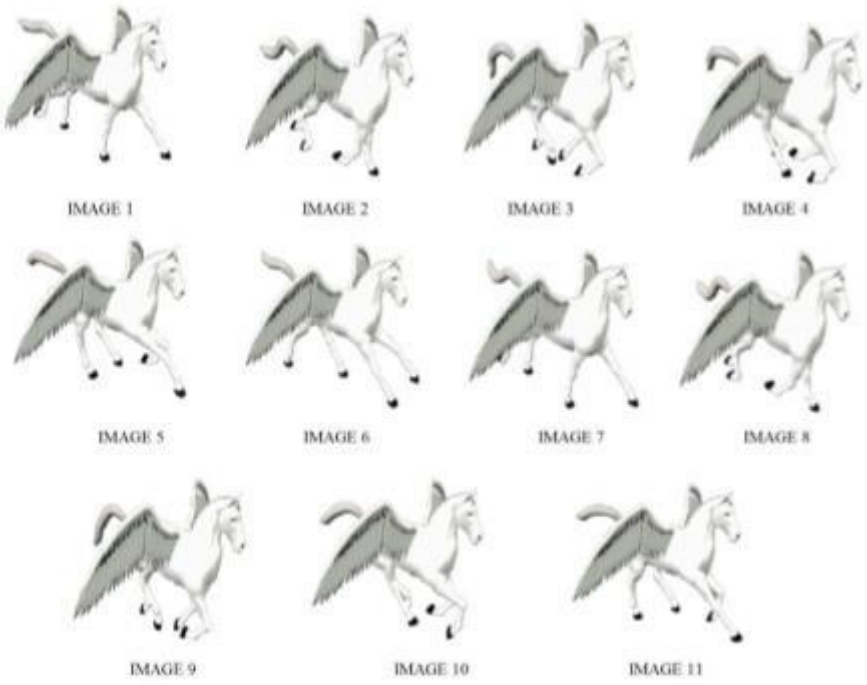


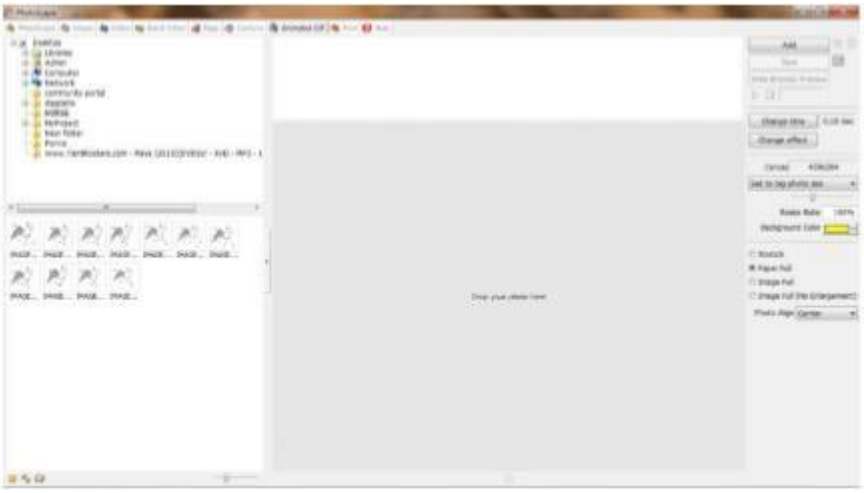
IMAGE 1 IMAGE 2 IMAGE 3 IMAGE 4

IMAGE 5 IMAGE 6 IMAGE 7 IMAGE 8

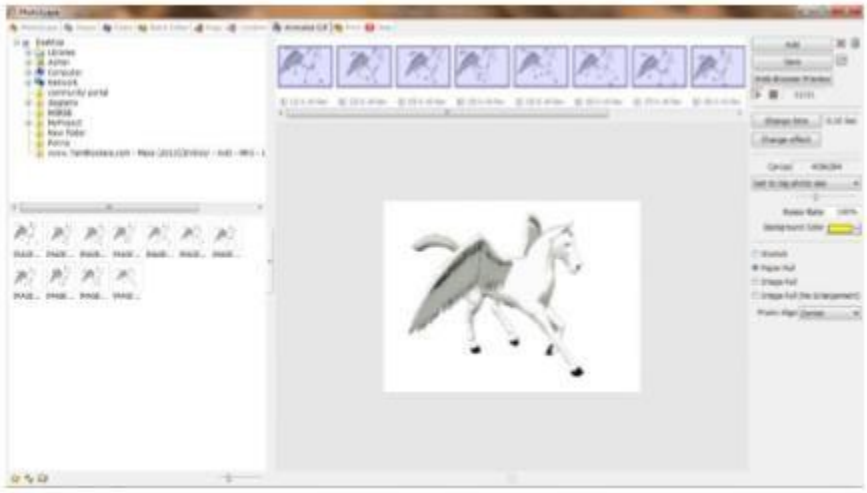
IMAGE 9 IMAGE 10 IMAGE 11

ii. Open PhotoScape.

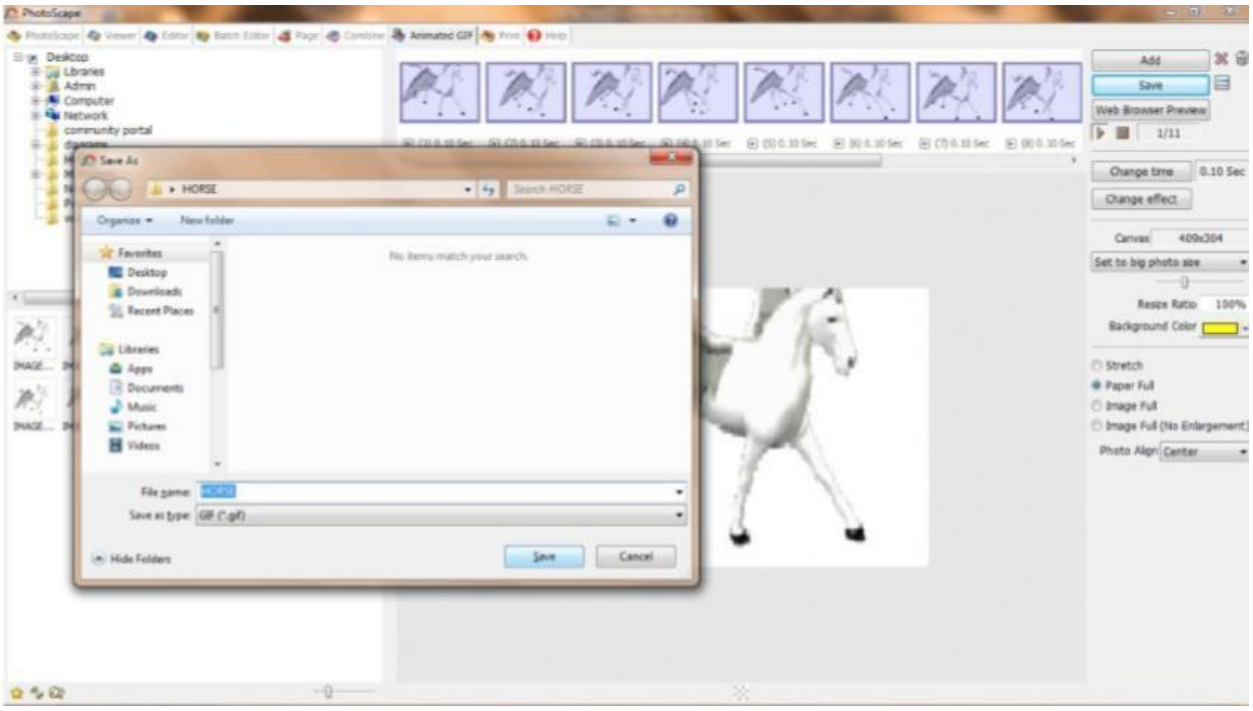
iii. Choose Animated GIF from the menu.



iv. Drop all the images in center bottom portion of the window.



v. Save the animated file in gif format.



3. To optimize the image in Adobe Photoshop 7.0

i. Open the image in Adobe Photoshop 7.0 which size is 85.2 KB

ii. File Save for Web (or press Alt + Shift + Ctrl + S)

iii. Adjust the Quality and Blur of the image. Finally the optimized image in jpg format which size is 40.3 KB.

RESULT:

Thus the operations (Basic Operations on image, creating gif animated image, optimize the image) has been performed and the output was verified.

EX NO:	**2D ANIMATION**
DATE:	

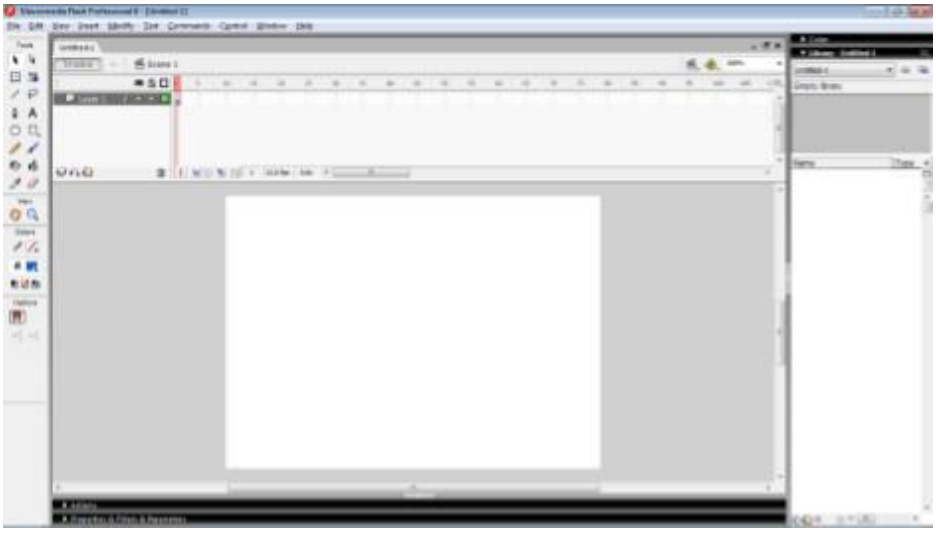
AIM:

To create interactive animation using Macromedia Flash 8.

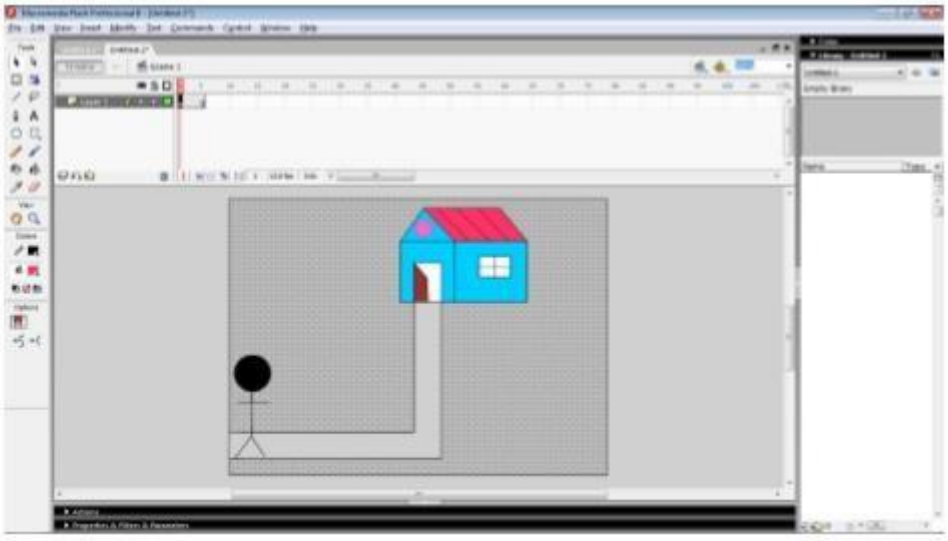
PROCEDURE:

1. Open Macromedia Flash 8.

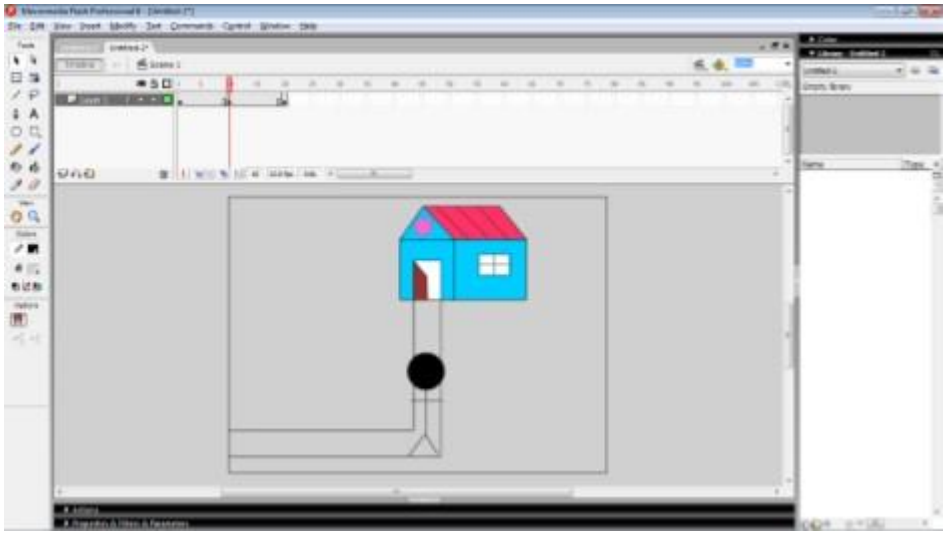
2. Create New Flash Document.



3. Draw the 2D image (scene) to animate.



4. Insert a new frame and change the scene for new frame.



5. Similarly create many frames.

6. Control Play (or press Enter) to run the Animation.

RESULT:

Thus the interactive 2D Animation has been created and the output was verified.