DON BOSCO COLLEGE

[Affiliated to Kannur University & Approved by A.I.C.T.E]

ANGADIKADAVU

KANNUR

THIRD SEMESTER MCA PRACTICAL RECORD

IN

COMPUTER GRAPHICS

&

PRINCIPLES OF INTELLIGENT SYSTEMS

Submitted by

NAMBIAR VIVEK VINOD NIRMALA

REG.NO:C0GMCA2111

(2020-2022)

DEPARTMENT OF COMPUTER APPLICATIONS

DON BOSCO COLLEGE, ANGADIKADAVU

KANNUR

DEPARTMENT OF COMPUTER APPLICATIONS

DONBOSCO COLLEGE, ANGADIKADAVU

KANNUR

CERTIFICATE

Certified that this is a bonafide record of work done by Mr. NAMBIAR VIVEK VINOD NIRMALA (Reg.No:C0GMCA2111) Department of Computer Applications, Don Bosco College, Angadikadavu in MCA3 P01–(COMPUTER GRAPHICS & PRINCIPLES OF INTELLIGENT SYSTEMS) during the third semester, DECEMBER 2022.

Head of Department

Faculty in charge

External Examiners

1:

2:

Place: Angadikadavu

Date:

INDEX 1: COMPUTER GRAPHICS

INDEX 2: PRINCIPLES OF INTELLIGENT SYSTEMS

COMPUTER GRAPHICS

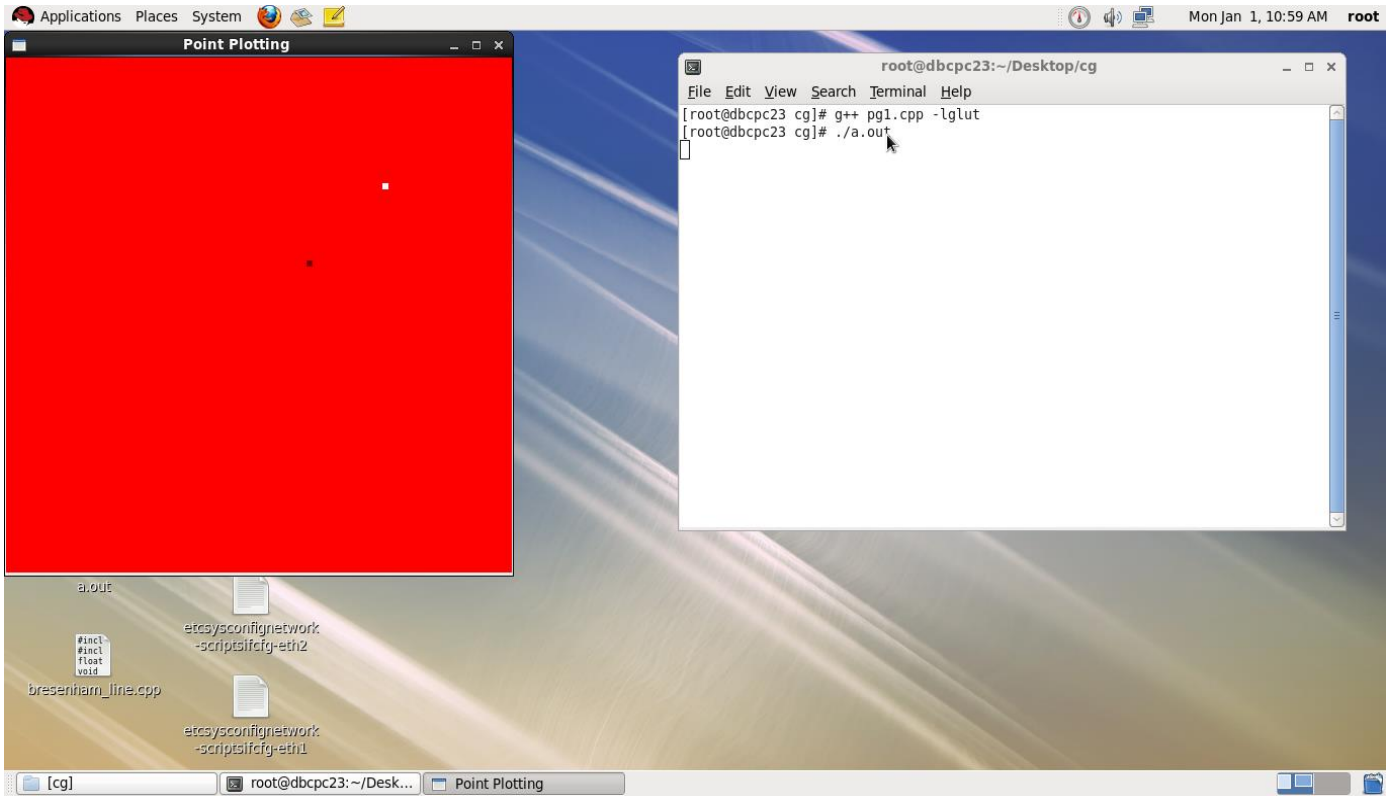
PROGRAM NO 1

AIM: Program to plot a point.

PROGRAM CODE:

```
#include<GL/glut.h>
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glPointSize(6.0);
   glBegin(GL_POINTS);
   glVertex2i(50,50);
   glColor3f(0.5,0.0,0.0);
   glVertex2i(20,20);
   glColor3f(0.1,0.2,0.3);
   glEnd();
   glFlush();
}
main(int argc,char **argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(0,0);
   glutCreateWindow("Point Plotting");
   glClearColor(1.2,0.0,0.0,0.0);
   glOrtho(-100,100,-100,100,-10,10);
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

OUTPUT:


Applications Places System
Mon Jan 1, 10:59 AM root
Point Plotting
root@dbcpc23:~/Desktop/cg
File Edit View Search Terminal Help
[root@dbcpc23 cg]# g++ pg1.cpp -lglut
[root@dbcpc23 cg]# ./a.out
a.out
etcsysconfignetwork-scriptsifcfg-eth2
bresenham_line.cpp
etcsysconfignetwork-scriptsifcfg-eth1
[cg]
root@dbcpc23:~/Desk...
Point Plotting

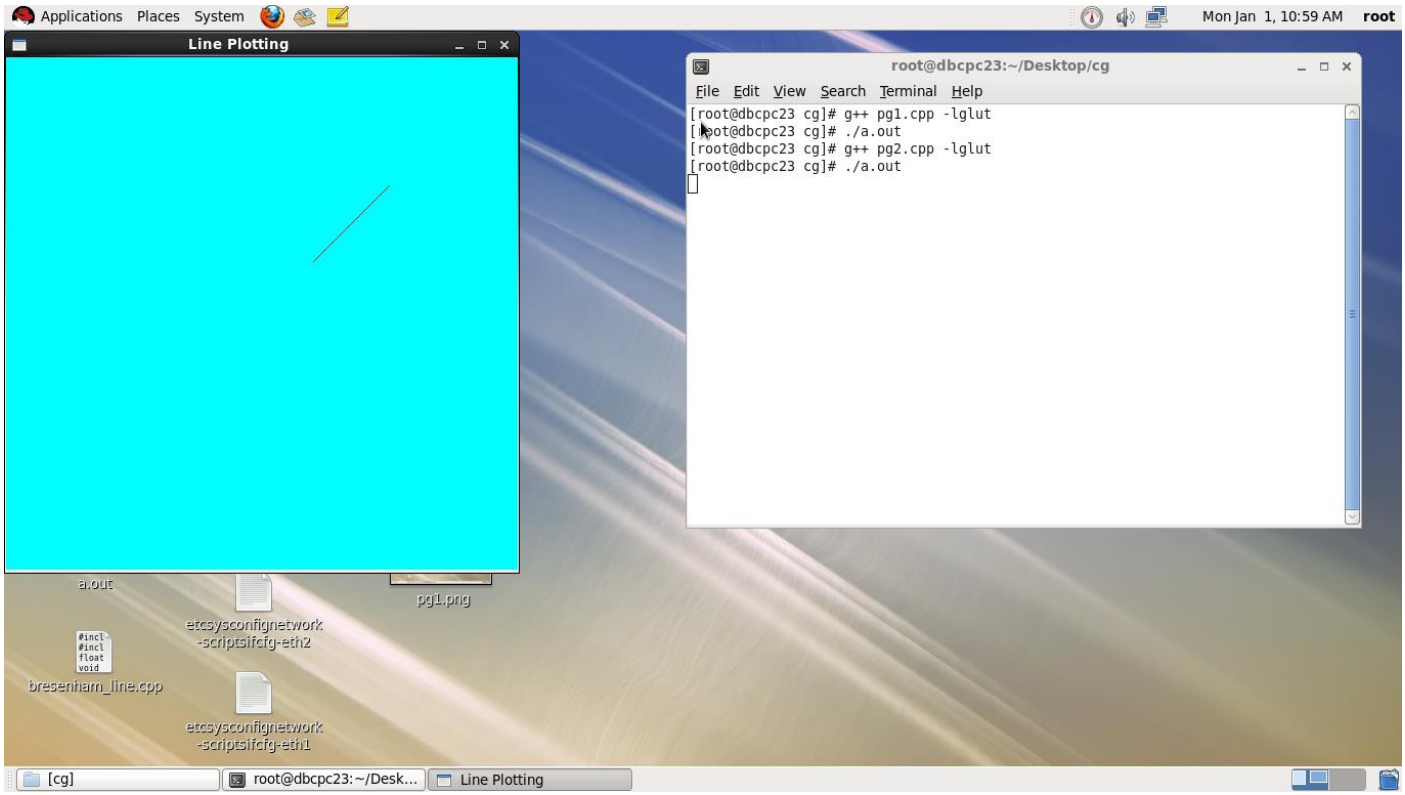
PROGRAM NO 2

AIM: Program to plot a line.

PROGRAM CODE :

```
#include<GL/glut.h>
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glPointSize(6.0);
	glBegin(GL_LINES);
	 glColor3f(1.0,0.0,0.0);
	glVertex2i(50,50);
	glVertex2i(20,20);
	glEnd();
	glFlush();
}
main(int argc,char **argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(0,0);
	glutCreateWindow("Line Plotting");
	glClearColor(0.0,1.0,1.0,0.0);
	glOrtho(-100,100,-100,100,-10,10);
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```

OUTPUT :


Applications Places System
Mon Jan 1, 10:59 AM root
Line Plotting
root@dbcpc23:~/Desktop/cg
File Edit View Search Terminal Help
[root@dbcpc23 cg]# g++ pg1.cpp -lglut
[root@dbcpc23 cg]# ./a.out
[root@dbcpc23 cg]# g++ pg2.cpp -lglut
[root@dbcpc23 cg]# ./a.out
a.out
pg1.png
etcsysconfignetwork-scriptsifcfg-eth2
bresenham_line.cpp
etcsysconfignetwork-scriptsifcfg-eth1
[cg]
root@dbcpc23:~/Desk...
Line Plotting

PROGRAM NO 3

AIM: DDA line drawing algorithm.

PROGRAM CODE :

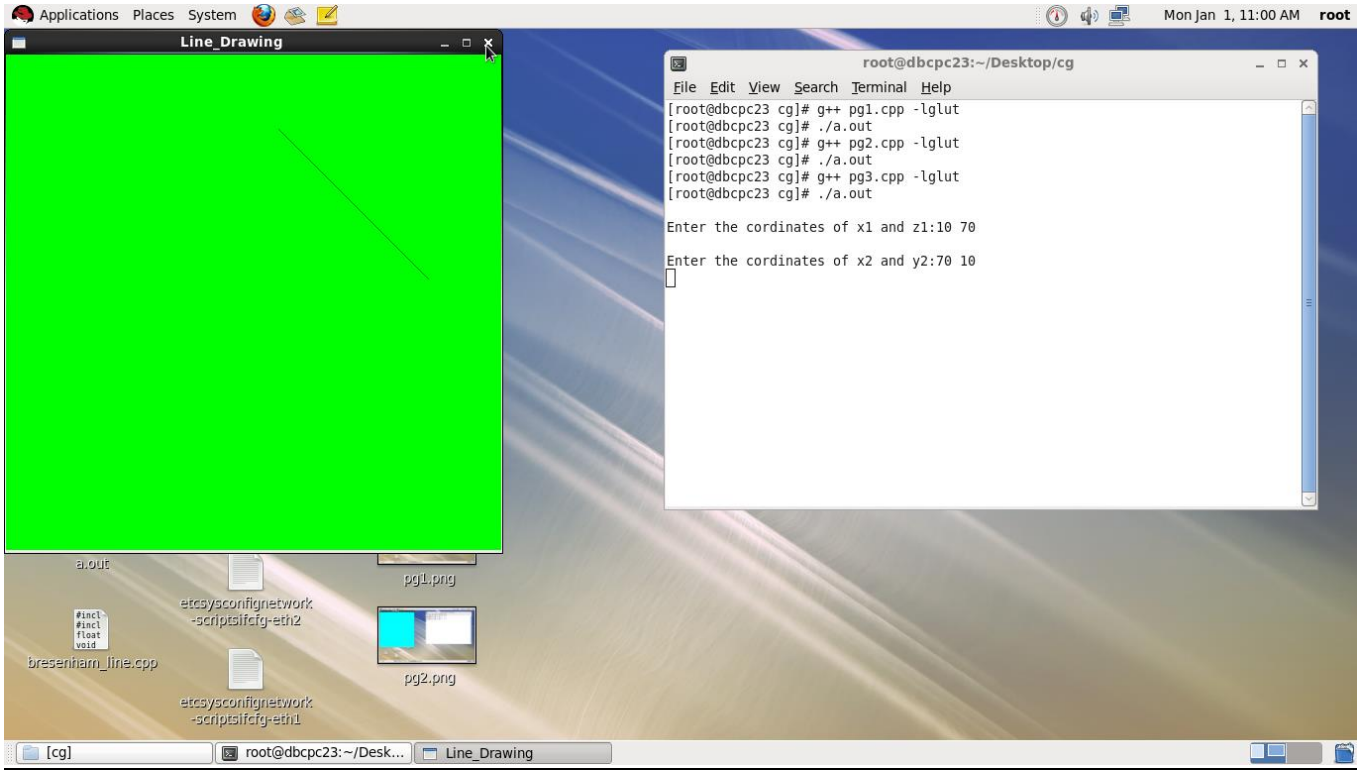
```
#include<stdio.h>
#include<GL/glut.h>
float x,y,x1,z1,x2,y2,dx,dy,step;
void dda()
{
int xinc,yinc,k;
glClear(GL_COLOR_BUFFER_BIT);
glPointSize(9.0);
glColor3f(0.0,0.5,0.0);
dx=x2-x1;
dy=y2-z1;
if(abs(dx)>abs(dy))
step=abs(dx);
else
step=abs(dy);
xinc=dx/(float)step;
yinc=dy/(float)step;
x=x1;
y=z1;
for(k=0;k<=step;k++)
{
glBegin(GL_LINES);
glVertex2f(x,y);
x=x+xinc;
y=y+yinc;
glVertex2f(x,y);
}
glEnd();
glFlush();
}
main(int argc,char **argv)
{
printf("\nEnter the cordinates of x1 and z1:");
scanf("%f%f",&x1,&z1);
printf("\nEnter the cordinates of x2 and y2:");
scanf("%f%f",&x2,&y2);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
```

```
glutCreateWindow("Line_Drawing");
glClearColor(0.0,1.0,0.0,0.0);
glOrtho(-100,100,-100,100,-10,10);
glutDisplayFunc(dda);
glutMainLoop();
return 0;
}
```

OUTPUT :

Enter the cordinates of x1 and z1:20 80

Enter the cordinates of x2 and y2:30 90



PROGRAM NO 4

AIM: Bresenhams line drawing algorithm.

PROGRAM CODE :

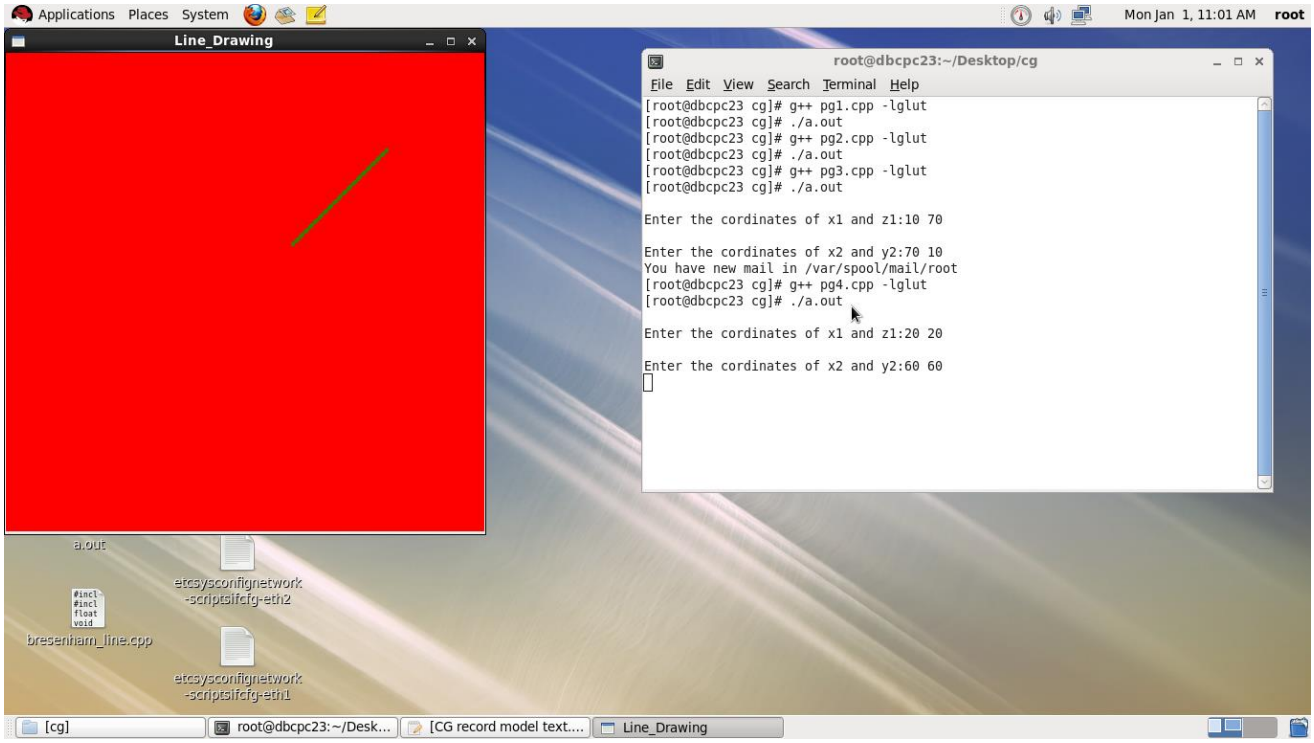
```
#include<stdio.h>
#include<GL/glut.h>
float x1,x2,z1,y2,x,y,step,p,dx,dy;
void bresenham()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glPointSize(4.0);
	glColor3f(0.2,0.5,0.0);
	int k;
	dx=x2-x1;
	dy=y2-z1;
	step=dx-1;
	p=2*(dy-dx);
	x=x1;
	y=z1;
	for(k=0;k<step;k++)
	{
		glBegin(GL_POINTS);
		glVertex2f(x,y);
		if(p<0)
		{
			x=x+1;
			p=p+(2*dy);
		}
		else
		{
			x=x+1;
			y=y+1;
			p=p+2*(dy-dx);
		}
		glVertex2f(x,y);
	}
	glEnd();
	glFlush();
}
main(int argc,char **argv)
```

```
{
    printf("\nEnter the cordinates of x1 and z1:");
    scanf("%f%f",&x1,&z1);
    printf("\nEnter the cordinates of x2 and y2:");
    scanf("%f%f",&x2,&y2);
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Line_Drawing");
    glClearColor(1.0,0.0,0.0,0.0);
    glOrtho(-100,100,-100,100,-10,10);
    glutDisplayFunc(bresenham);
    glutMainLoop();
    return 0;
}
```

OUTPUT :

Enter the coordinates of x1 and z1:20 20

Enter the cordinates of x2 and y2:60 60



PROGRAM NO 5

AIM: Program for circle drawing.

PROGRAM CODE:

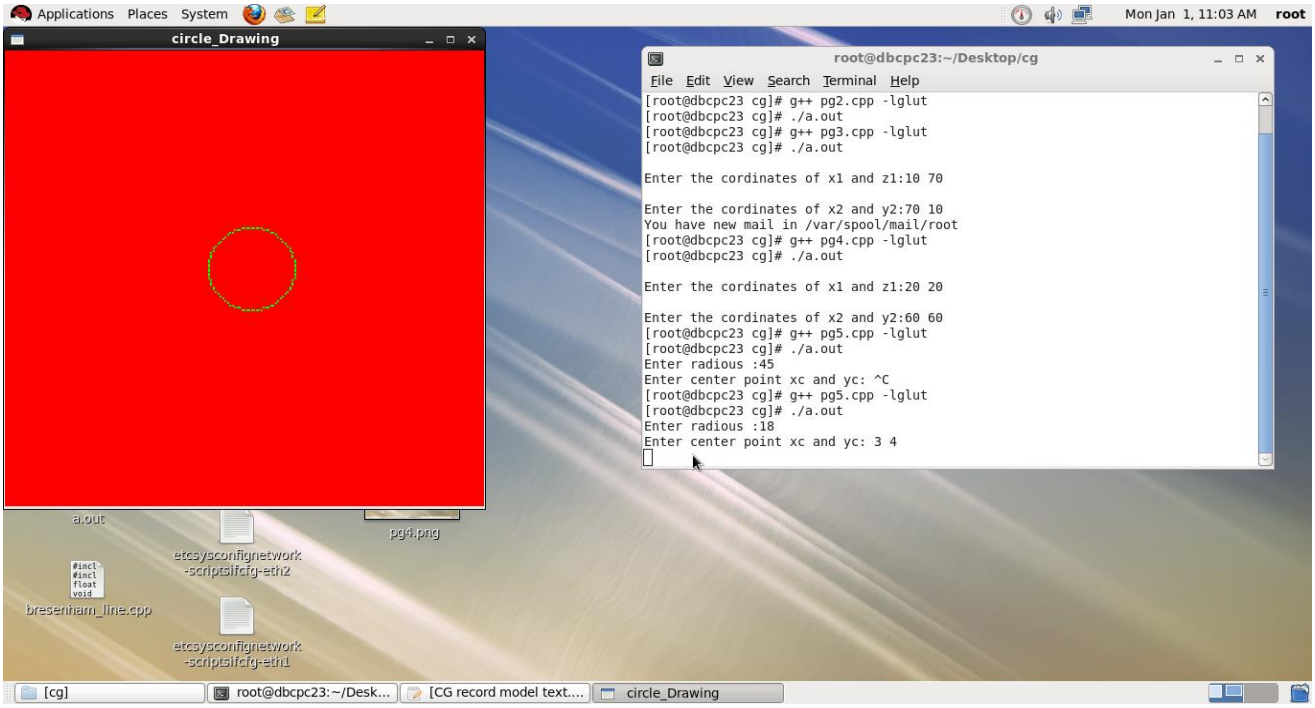
```
#include<stdio.h>
#include<GL/glut.h>
void circle_plotting(float xx,float yy);
float r,xc,yc,x,y,p;
void circle_algorithm()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0,1.0,0.0);
    glPointSize(2.0);
    x=0;
    y=r;
    p=1-r;
    glBegin(GL_POINTS);
    circle_plotting(x,y);
    while(x<=y)
    {
        if(p<0)
        {
            x=x+1;
            p=p+(2*x)+1;
        }
        else
        {
            x=x+1;
            y=y-1;
            p=p+(2*(x-y))+1;
        }
        circle_plotting(x,y);
    }
    glEnd();
    glFlush();
}


void circle_plotting(float xx,float yy)
{
    glVertex2f(xc+xx,yc+yy);
    glVertex2f(xc-xx,yc-yy);
    glVertex2f(xc-xx,yc+yy);
    glVertex2f(xc+xx,yc-yy);
```

```
        glVertex2f(xc+yy,yc+xx);
        glVertex2f(xc-yy,yc+xx);
        glVertex2f(xc+yy,yc-xx);
        glVertex2f(xc-yy,yc-xx);
}
main(int argc,char **argv)
{
        printf("Enter radious :");
        scanf("%f",&r);
        printf("Enter center point xc and yc: ");
        scanf("%f%f",&xc,&yc);
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(0,0);
        glutCreateWindow(" circle_Drawing");
        glClearColor(1.0,0.0,0.0,0.0);
        glOrtho(-100,100,-100,100,-10,10);
        glutDisplayFunc(circle_algorithm);
        glutMainLoop();
        return 0;
}
```

OUTPUT :

Enter radius :18
Enter center point xc and yc: 3 4



PROGRAM NO 6

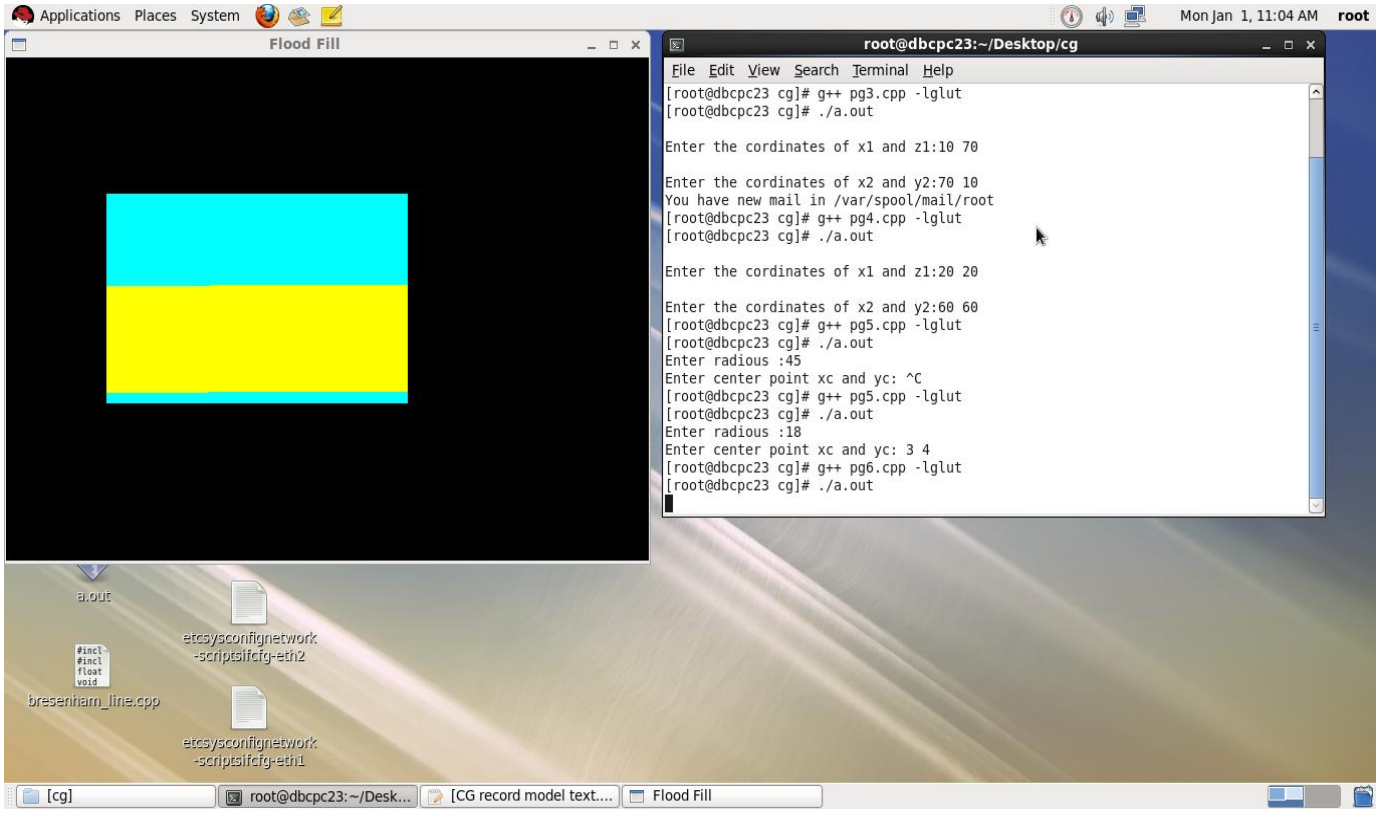
AIM: Program for Food fill algorithm.

PROGRAM CODE :

```
#include<GL/glut.h>
void floodfill(float,float,float[],float[]);
float fill[3]={1.0,1.0,0.0},old[3]={0.0,1.0,1.0};
void display()
{
	glClearColor(0,0,0,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3fv(old);
	glBegin(GL_POLYGON);
	glVertex2i(100,150);
	glVertex2i(400,150);
	glVertex2i(400,350);
	glVertex2i(100,350);
	glEnd();
	glFlush();
	floodfill(200.0,160.0,fill,old);
	glFlush();
}
void floodfill(float x,float y,float fill[3],float old[3])
{
	float pix[3];
	glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,pix);
	if(pix[0]==old[0]&&pix[1]==old[1]&&pix[2]==old[2])
	{
		glBegin(GL_POINTS);
		glColor3fv(fill);
		glVertex2f(x,y);
		glEnd();
		glFlush();
		floodfill(x-1,y,fill,old);
		floodfill(x+1,y,fill,old);
		floodfill(x,y+1,fill,old);
		floodfill(x,y-1,fill,old);
	}
}
int main(int argc,char *argv[])
{
	glutInit(&argc,argv);
	glutInitWindowSize(640,480);
	glutCreateWindow("Flood Fill");
	glutDisplayFunc(display);
```

```
    glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);
    glutMainLoop();
    return 0;
}
```

OUTPUT:



PROGRAM NO 7

AIM: Line clipping using Cohen Sutherland algorithm.

PROGRAM CODE :

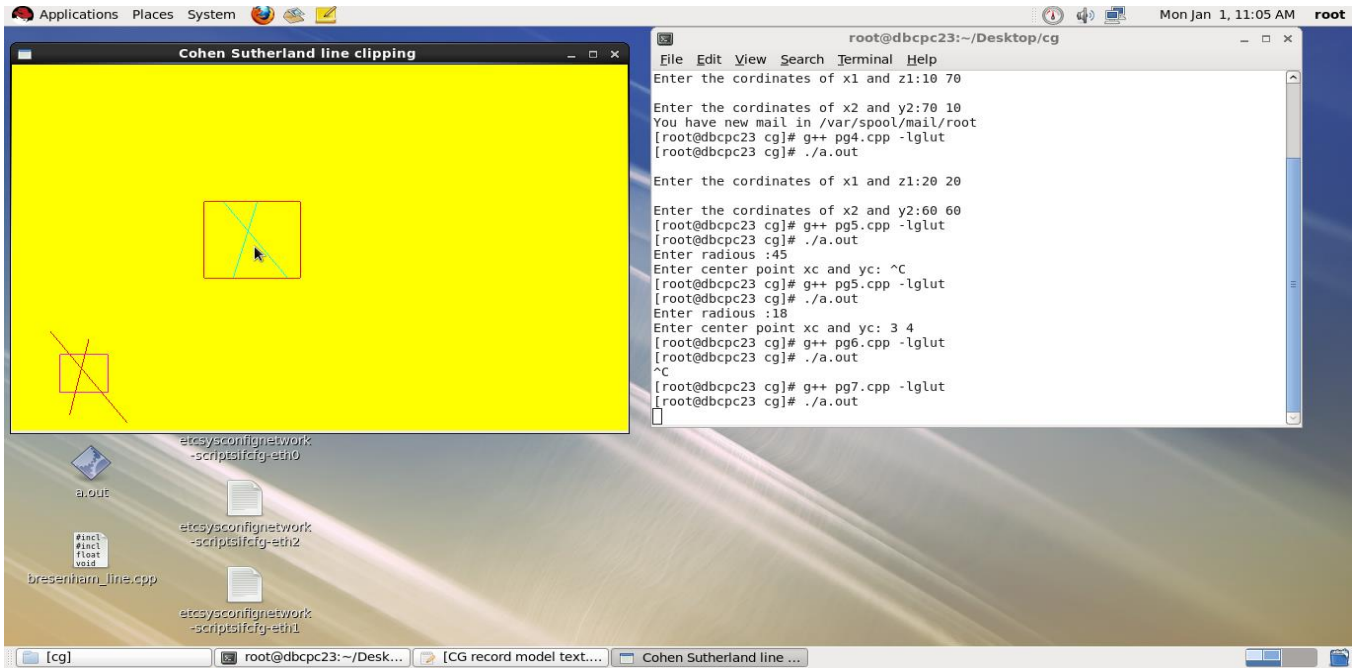
```
#include<stdio.h>
#include<GL/glut.h>
#define outcode int
double xmin=50,ymin=50,xmax=100,ymax=100;
double xvmin=200,yvmin=200,xvmax=300,yvmax=300;
const int RIGHT=8;
const int LEFT=2;
const int TOP=4;
const int BOTTOM=1;
outcode computeoutcode(double x,double y);
void cohen_clipping(double x0,double y0,double x1,double y1)
{
    outcode outcode0,outcode1,outcodeout;
    int accept=0,done=0;
    outcode0=computeoutcode(x0,y0);
    outcode1=computeoutcode(x1,y1);
    do
    {
        if(!(outcode0|outcode1))
        {
            accept=1;
            done=1;
        }
        else if(outcode0 & outcode1)
            done=1;
        else
        {
            double x,y;
            outcodeout=outcode0?outcode0:outcode1;
            if(outcodeout & TOP)
            {
                x=x0+(x1-x0)*(ymax-y0)/(y1-y0);
                y=ymax;
            }
            else if(outcodeout & BOTTOM)
```

```
                {
                        x=x0+(x1-x0)*(ymin-y0)/(y1-y0);
                        y=ymin;
                }
                else if(outcodeout & RIGHT)
                {
                        y=y0+(y1-y0)*(xmax-x0)/(x1-x0);
                        x=xmax;
                }
                else
                {
                        y=y0+(y1-y0)+(xmin-x0)/(x1-x0);
                        x=xmin;
                }
                if(outcodeout==outcode0)
                {
                        x0=x;
                        y0=y;
                        outcode0=computeoutcode(x0,y0);
                }
                else
                {
                        x1=x;
                        y1=y;
                        outcode1=computeoutcode(x1,y1);
                }
        }
}while(!done);
if(accept)
{
        double sx=(xvmax-xvmin)/(xmax-xmin);
        double sy=(yvmax-yvmin)/(ymax-ymin);
        double vx0=xvmin+(x0-xmin)*sx;
        double vy0=yvmin+(y0-ymin)*sy;
        double vx1=xvmin+(x1-xmin)*sx;
        double vy1=yvmin+(y1-ymin)*sy;
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xvmin,yvmin);
        glVertex2f(xvmax,yvmin);
        glVertex2f(xvmax,yvmax);
```

```
                glVertex2f(xvmin,yvmax);
                glEnd();
                glColor3f(0.0,1.0,1.0);
                glBegin(GL_LINES);
                glVertex2d(vx0,vy0);
                glVertex2d(vx1,vy1);
                glEnd();
        }
}
outcode computeoutcode(double x,double y)
{
        outcode code=0;
        if(y>ymax)
                code |=TOP;
        if(y<ymin)
                code |=BOTTOM;
        if(x>xmax)
                code |=RIGHT;
        if(x<xmin)
                code |=TOP;
        return code;
}
void display()
{
        double x0=120,y0=10,x1=40,y1=130;
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_LINES);
        glVertex2d(x0,y0);
        glVertex2d(x1,y1);
        glVertex2d(60,20);
        glVertex2d(80,120);
        glEnd();
        glColor3f(1.0,0.0,1.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xmin,ymin);
        glVertex2f(xmax,ymin);
        glVertex2f(xmax,ymax);
        glVertex2f(xmin,ymax);
        glEnd();
        cohen_clipping(x0,y0,x1,y1);
```

```
        cohen_clipping(60,30,80,110);
        glFlush();
}
void init()
{
        glClearColor(1.0,1.0,0.0,1.0);
        glColor3f(0.0,1.0,0.0);
        glPointSize(3.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);
}
int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutCreateWindow("Cohen Sutherland line clipping");
        glutInitWindowSize(300,900);
        glutInitWindowPosition(0,0);
        glutDisplayFunc(display);
        init();
        glutMainLoop();
        return 0;
}
```

OUTPUT:



PROGRAM NO 8

AIM: Program for Polygon clipping.

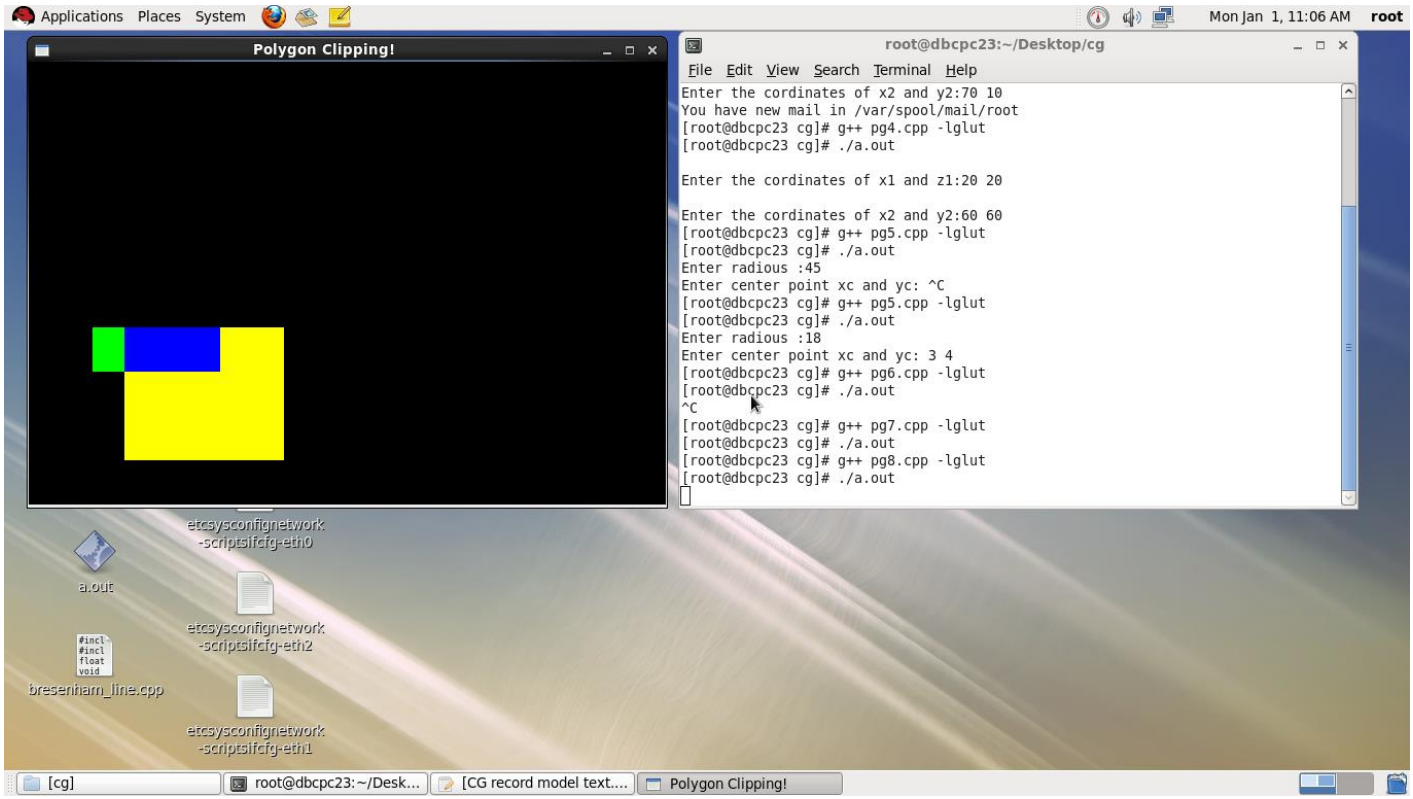
PROGRAM CODE :

```
#include <GL/glut.h>
struct Point
{
float x,y;
}
w[4],oVer[4];
 int Nout;
 void drawPoly(Point p[],int n)
 {
 glBegin(GL_POLYGON);
 for(int i=0;i<n;i++)
  glVertex2f(p[i].x,p[i].y);
 glEnd();
 }
bool insideVer(Point p)
{
if((p.x>=w[0].x)&&(p.x<=w[2].x))
if((p.y>=w[0].y)&&(p.y<=w[2].y))
return true;
return false;
}
void addVer(Point p)
{
oVer[Nout]=p;
Nout=Nout+1;
}
Point getInterSect(Point s,Point p,int edge)
{
Point in;
float m;
if(w[edge].x==w[(edge+1)%4].x)
{
m=(p.y-s.y)/(p.x-s.x);
in.x=w[edge].x;
in.y=in.x*m+s.y;
}
else
{
m=(p.y-s.y)/(p.x-s.x);
in.y=w[edge].y;
in.x=(in.y-s.y)/m;
```

```
}
return in;
}
void clipAndDraw(Point inVer[],int Nin)
{
Point s,p,interSec;
for(int i=0;i<4;i++)
 {
Nout=0;
s=inVer[Nin-1];
for(int j=0;j<Nin;j++)
{
    p=inVer[j];
        if(insideVer(p)==true){
                if(insideVer(s)==true){
                        addVer(p);
                                }
                                else{
                                interSec=getInterSect(s,p,i);
                                addVer(interSec);
                                addVer(p);
                                }
                        }
                else
                {
                if(insideVer(s)==true)
                {
                        interSec=getInterSect(s,p,i);
                        addVer(interSec);
                }
                }
                s=p;
        }
        inVer=oVer;
        Nin=Nout;
        }
        drawPoly(oVer,4);
        }
        void init()
        {
        glClearColor(0.0f,0.0f,0.0f,0.0f);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0,100.0,0.0,100.0,0.0,100.0);
        glClear(GL_COLOR_BUFFER_BIT);
        w[0].x =15,w[0].y=10;
        w[1].x =15,w[1].y=40;
```

```
w[2].x =40,w[2].y=40;
w[3].x =40,w[3].y=10;
}
void display(void)
{
Point inVer[4];
init();
glColor3f(1.0f,1.0f,0.0f);
drawPoly(w,4);
glColor3f(0.0f,1.0f,0.0f);
inVer[0].x =10,inVer[0].y=40;
inVer[1].x =10,inVer[1].y=30;
inVer[2].x =30,inVer[2].y=30;
inVer[3].x =30,inVer[3].y=40;
drawPoly(inVer,4);
glColor3f(0.0f,0.0f,1.0f);
clipAndDraw(inVer,4);
glFlush();
}
int main(int argc,char *argv[])
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(400,400);
glutInitWindowPosition(100,100);
glutCreateWindow("Polygon Clipping!");
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

OUTPUT:


Applications Places System
Mon Jan 1, 11:06 AM root
Polygon Clipping!
root@dbcpc23:~/Desktop/cg
File Edit View Search Terminal Help
Enter the cordinates of x2 and y2:70 10
You have new mail in /var/spool/mail/root
[root@dbcpc23 cg]# g++ pg4.cpp -lglut
[root@dbcpc23 cg]# ./a.out
Enter the cordinates of x1 and z1:20 20
Enter the cordinates of x2 and y2:60 60
[root@dbcpc23 cg]# g++ pg5.cpp -lglut
[root@dbcpc23 cg]# ./a.out
Enter radious :45
Enter center point xc and yc: ^C
[root@dbcpc23 cg]# g++ pg5.cpp -lglut
[root@dbcpc23 cg]# ./a.out
Enter radious :18
Enter center point xc and yc: 3 4
[root@dbcpc23 cg]# g++ pg6.cpp -lglut
[root@dbcpc23 cg]# ./a.out
^C
[root@dbcpc23 cg]# g++ pg7.cpp -lglut
[root@dbcpc23 cg]# ./a.out
[root@dbcpc23 cg]# g++ pg8.cpp -lglut
[root@dbcpc23 cg]# ./a.out
etcsysconfignetwork-scriptsifcfg-eth0
a.out
etcsysconfignetwork-scriptsifcfg-eth2
bresenham_line.cpp
etcsysconfignetwork-scriptsifcfg-eth1
[cg]
root@dbcpc23:~/Desk...
[CG record model text....
Polygon Clipping!

PROGRAM NO 9

AIM: 2D Transformation Program.

PROGRAM CODE :

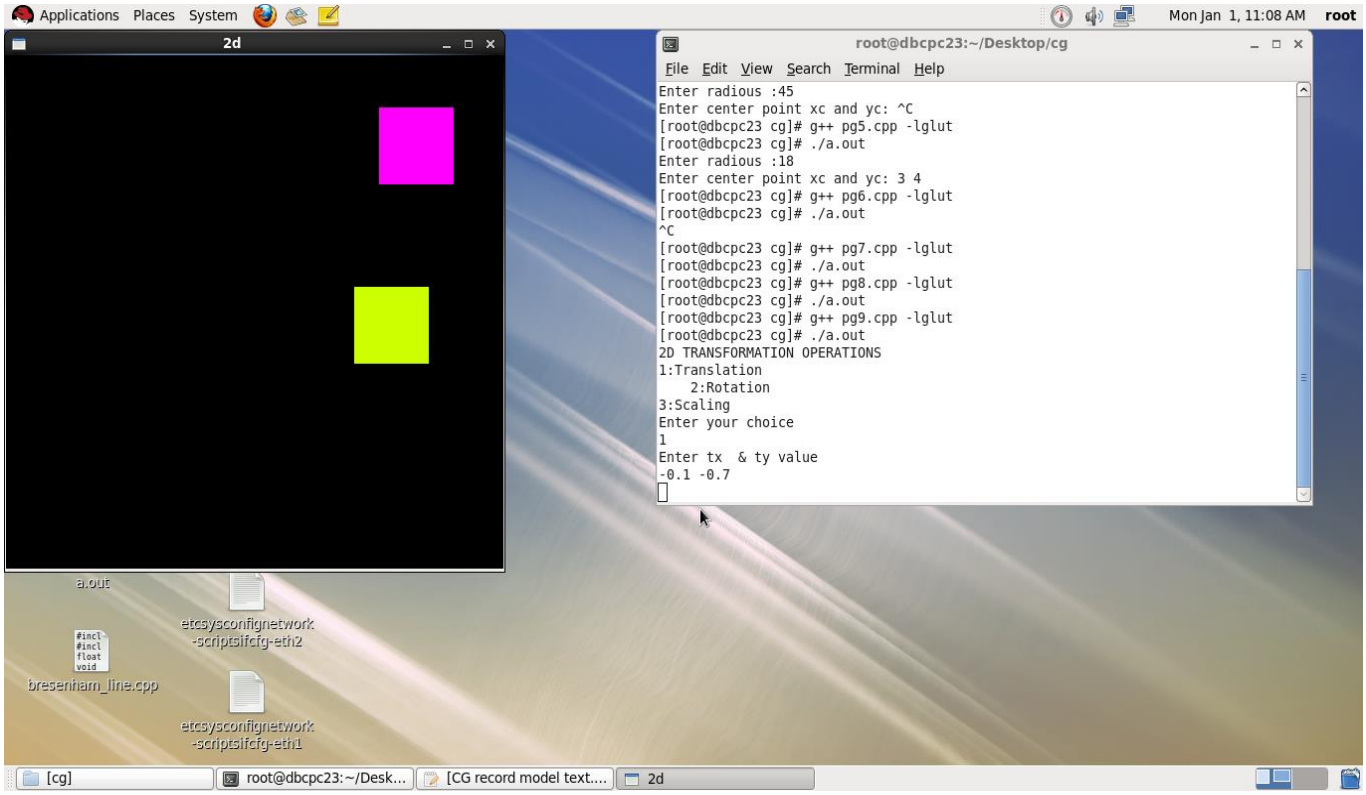
```
#include<stdio.h>
#include<math.h>
#include<GL/glut.h>
int ch;
float x1=0.5,x2=0.8,x3=0.8,x4=0.5,y=0.5,y2=0.5,y3=0.8,y4=0.8;
float X1,X2,X3,X4,Y,Y2,Y3,Y4;
void display(void)
{
      float tx,ty;
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1.0,0.0,1.0);
      glPointSize(10.0);
      glBegin(GL_POLYGON);
      glVertex2f(x1,y);
      glVertex2f(x2,y2);
      glVertex2f(x3,y3);
      glVertex2f(x4,y4);
      glEnd();
      glColor3f(0.8,1.0,0.0);
      glBegin(GL_POLYGON);
      glVertex2f(X1,Y);
      glVertex2f(X2,Y2);
      glVertex2f(X3,Y3);
      glVertex2f(X4,Y4);
      glEnd();
      glFlush();
      }
      void translate()
      {
      float tx,ty;
      printf("Enter tx  & ty value\n");
      scanf("%f%f",&tx,&ty);
      X1=x1+tx;
      X2=x2+tx;
      X3=x3+tx;
      Y=y+ty;
      Y2=y2+ty;
      Y3=y3+ty;
      X4=x4+tx;
      Y4=y4+ty;
      }
```

```
void rotate()
{
int theta;
printf("Enter an angle\n");
scanf("%d",&theta);
X1=x1*cos(theta)-y*sin(theta);
X2=x2*cos(theta)-y2*sin(theta);
X3=x3*cos(theta)-y3*sin(theta);
X4=x4*cos(theta)-y4*sin(theta);
Y=x1*sin(theta)+y*cos(theta);
Y2=x2*sin(theta)+y2*cos(theta);
Y3=x3*sin(theta)+y3*cos(theta);
Y4=x4*sin(theta)+y4*cos(theta);
}
void scale()
{
float sx,sy;
printf("enter sx & sy value\n");
scanf("%f%f",&sx,&sy);
X1=x1*sx;
X2=x2*sx;
X3=x3*sx;
X4=x4*sx;
Y=y*sy;
Y2=y2*sy;
Y3=y3*sy;
Y4=y4*sy;
}
main(int argc,char **argv)
{
printf("2D TRANSFORMATION OPERATIONS \n1:Translation\n
2:Rotation\n3:Scaling\n");
printf("Enter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1: translate();
        break;
case 2: rotate();
        break;
case 3: scale();
        break;
}
glutInit(&argc,argv);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("2d");
```

```
glClearColor(0.0,0.0,0.0,0.0);
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

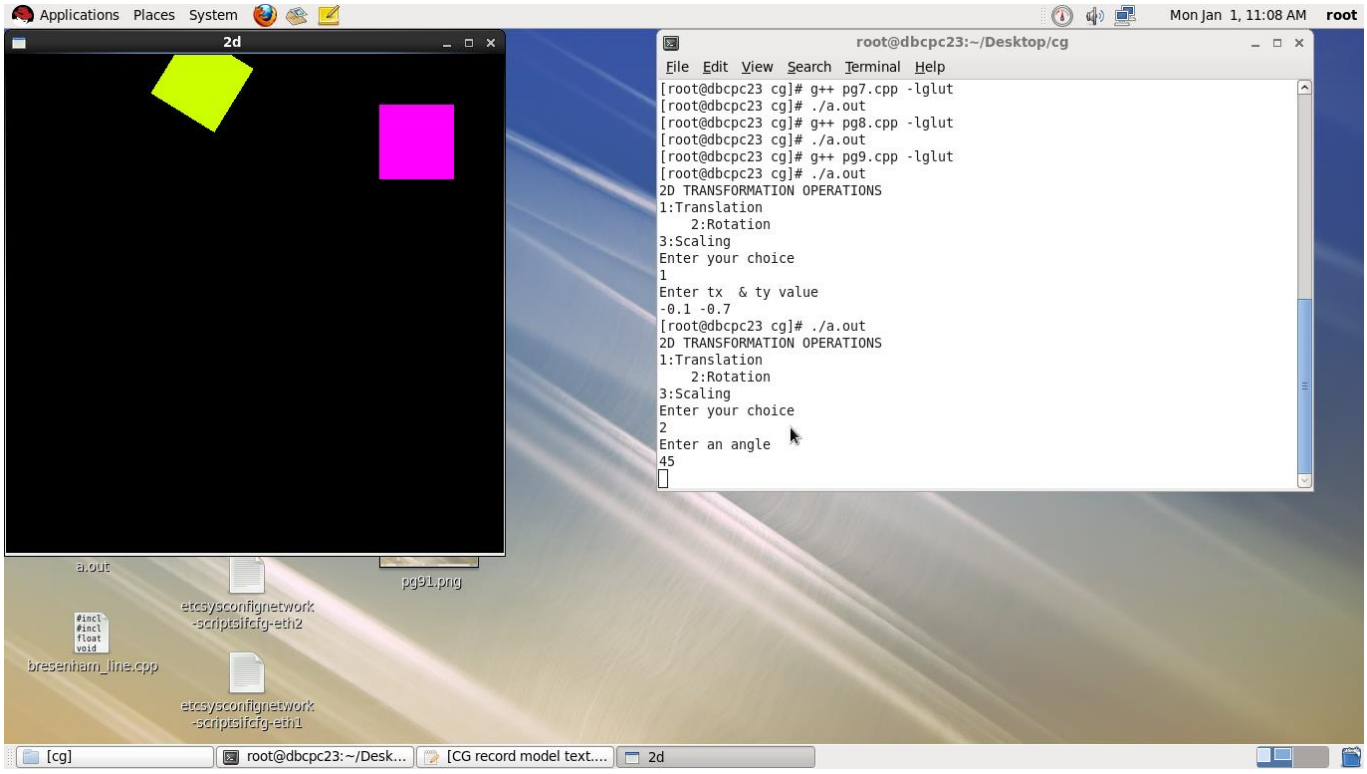
OUTPUT :

2D TRANSFORMATION OPERATIONS
1:Translation
2:Rotation
3:Scaling
Enter your choice
1
Enter tx & ty value
-0.1 -0.7



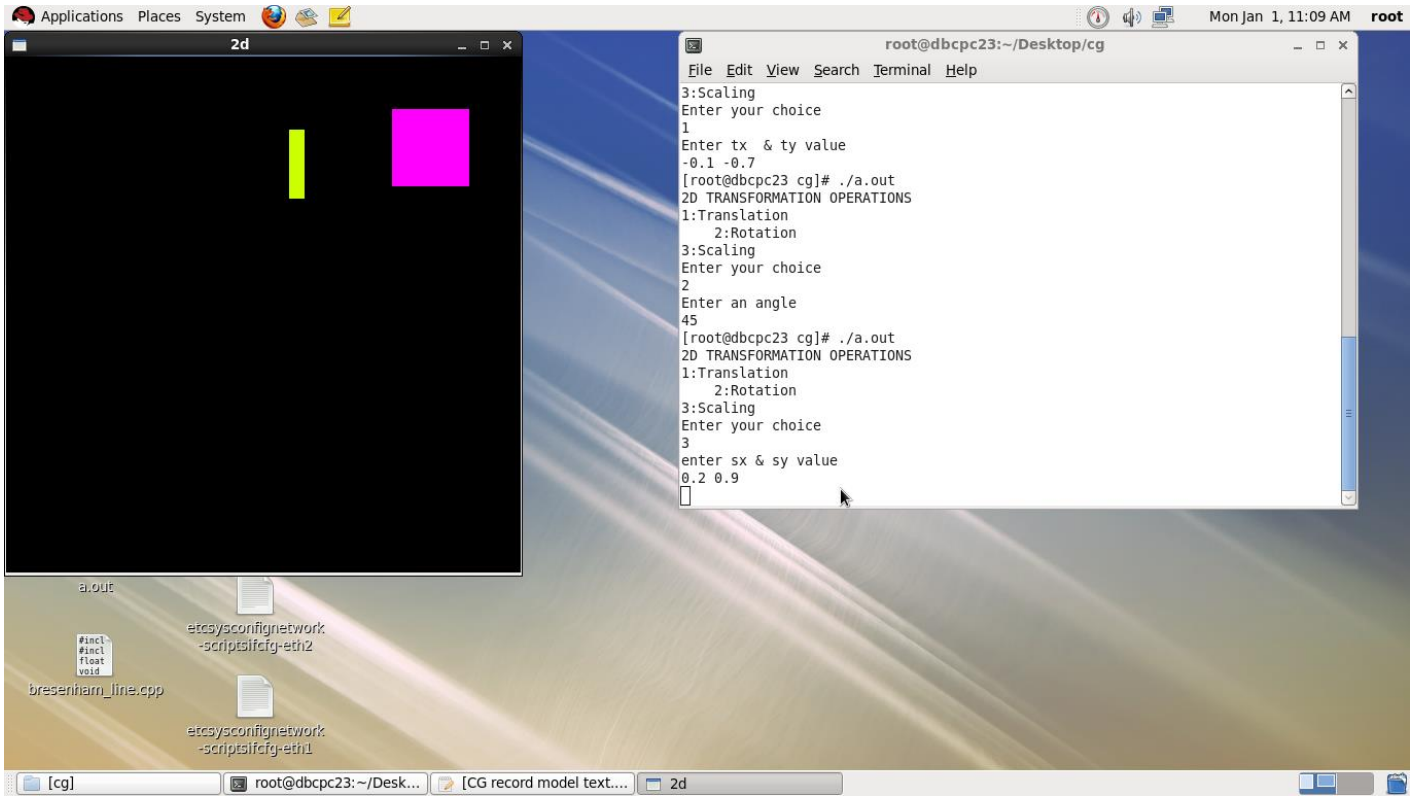
2D TRANSFORMATION OPERATIONS

1:Translation
2:Rotation
3:Scaling
Enter your choice
2
Enter an angle
45



2D TRANSFORMATION OPERATIONS

1:Translation
2:Rotation
3:Scaling
Enter your choice
3
enter sx & sy value
0.2 0.9



PROGRAM NO 10

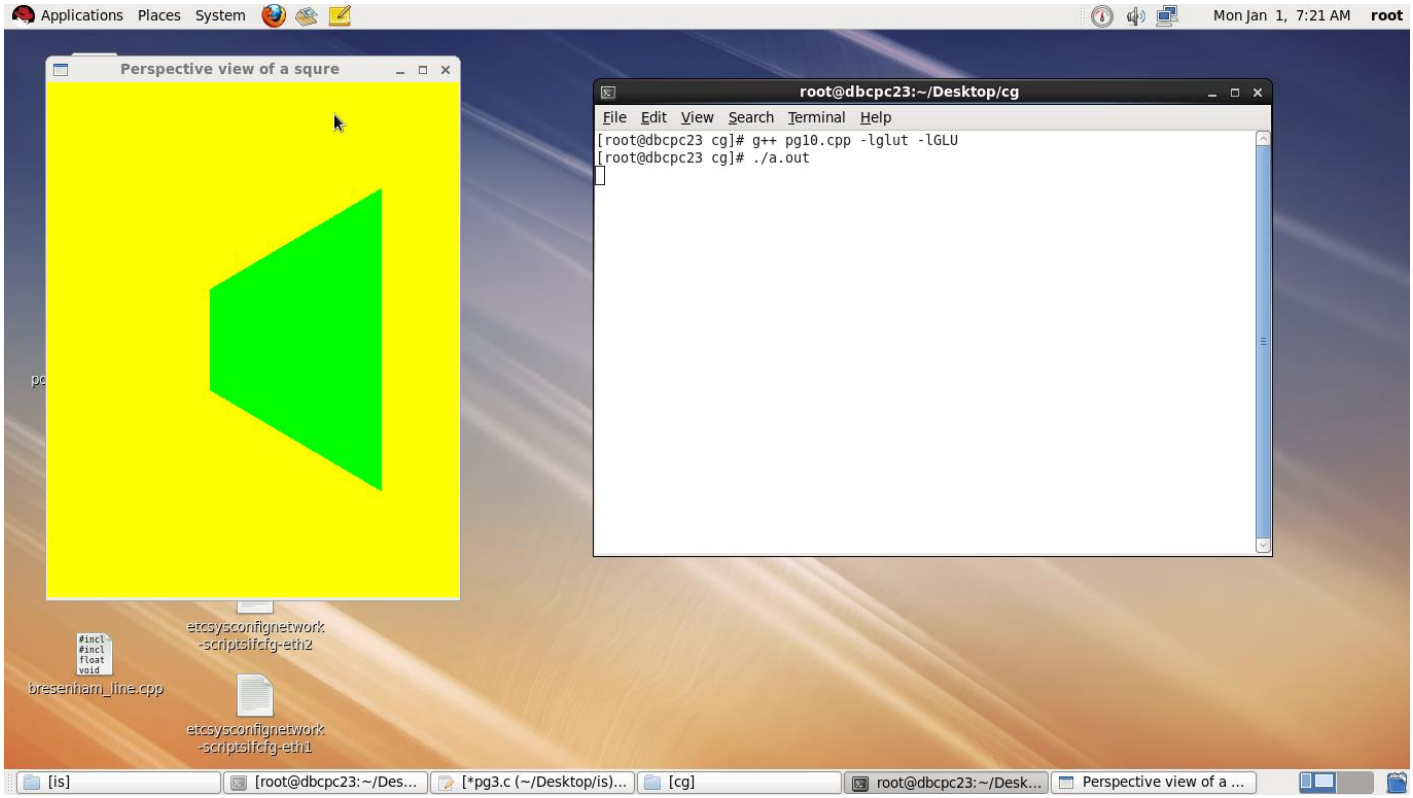
AIM: Perspective view of a square based on 3d viewing.

PROGRAM CODE :

```
#include<GL/glut.h>
GLint winWidth=600,winHeight=600;
GLfloat x0=100.0,y0=50.0,z0=50.0;
GLfloat xref=50.0,yref=50.0,zref=0.0;
GLfloat Vx=0.0,Vy=1.0,Vz=0.0;
GLfloat xwMin=-40.0,ywMin=-60.0,xwMax=40.0,ywMax=60.0;
GLfloat dnear=25.0,dfar=125.0;
void init(void)
{
glClearColor(1.0,1.0,0.0,0.0);
glMatrixMode(GL_MODELVIEW);
gluLookAt(x0,y0,z0,xref,yref,zref,Vx,Vy,Vz);
glMatrixMode(GL_PROJECTION);
glFrustum(xwMin,xwMax,ywMin,ywMax,dnear,dfar);
}
void displayFun(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0,1.0,0.0);
glPolygonMode(GL_FRONT,GL_FILL);
glPolygonMode(GL_BACK,GL_LINE);
glBegin(GL_QUADS);
glVertex3f(0.0,0.0,0.0);
glVertex3f(100.0,0.0,0.0);
glVertex3f(100.0,100.0,0.0);
glVertex3f(0.0,100.0,0.0);
glEnd();
glFlush();
}
void reshapeFun(GLint newWidth,GLint newHeight)
{
glViewport(0,0,newWidth,newHeight);
winWidth=newWidth;
winHeight=newHeight;
}
int main(int argc,char **argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(400,500);
glutInitWindowPosition(40,50);
```

```
glutCreateWindow("Perspective view of a squre");
init();
glutDisplayFunc(displayFun);
glutReshapeFunc(reshapeFun);
glutMainLoop();
return 0;
}
```

OUTPUT:



PROGRAM NO 11

AIM: Three dimensional transformations.

PROGRAM CODE:

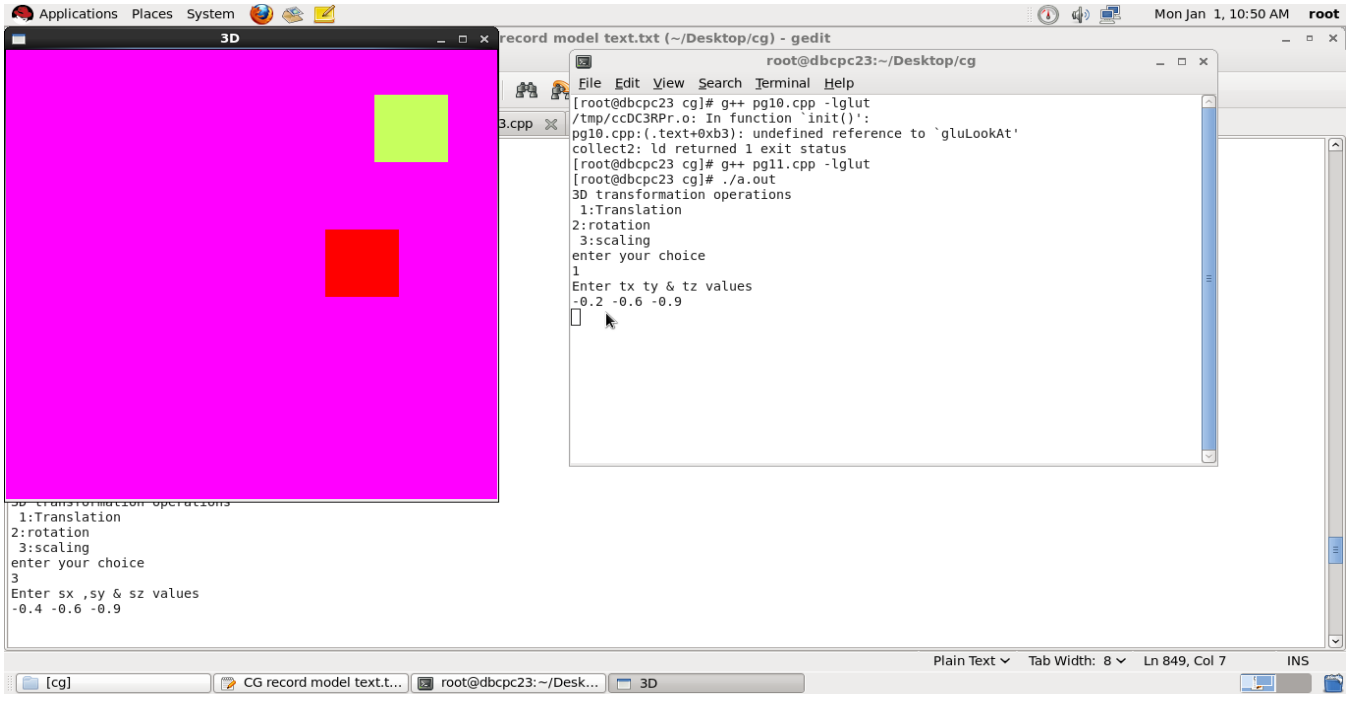
```
#include<stdio.h>
#include<math.h>
#include<GL/glut.h>
int ch;
float
x1=0.5,x2=0.8,x3=0.8,x4=0.5,y=0.5,y2=0.5,y3=0.8,y4=0.8,z1=0.6,z2=0.4,z3=0.7,z4=0.2;
float X1,X2,X3,X4,Y,Y2,Y3,Y4,Z1,Z2,Z3,Z4;
void display(void)
{
      float tx,ty;
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(0.78,9.33,0.37);
      glPointSize(10.0);
      glBegin(GL_POLYGON);
      glVertex3f(x1,y,z1);
      glVertex3f(x2,y2,z2);
      glVertex3f(x3,y3,z3);
      glVertex3f(x4,y4,z4);
      glEnd();
      glColor3f(8080,0.0,0.0);
      glBegin(GL_POLYGON);
      glVertex3f(X1,Y,Z1);
      glVertex3f(X2,Y2,Z2);
      glVertex3f(X3,Y3,Z3);
      glVertex3f(X4,Y4,Z4);
      glEnd();
      glFlush();
}
void translate()
{
      float tx,ty,tz;
      printf("Enter tx ty & tz values\n");
      scanf("%f%f%f",&tx,&ty,&tz);
      X1=x1+tx;X2=x2+tx;X3=x3+tx;X4=x4+tx;
      Y=y+ty;Y2=y2+ty;Y3=y3+ty;Y4=y4+ty;
```

```
        Z1=z1+tz;Z2=z2+tz;Z3=z3+tz;Z4=z4+tz;
}
void rotate()
{
        int theta;
        printf("Enter an angle\n");
        scanf("%d",&theta);
        X1=x1*cos(theta)-y*sin(theta);
        X2=x2*cos(theta)-y2*sin(theta);
        X3=x3*cos(theta)-y3*sin(theta);
        X4=x4*cos(theta)-y4*sin(theta);
        Y=x1*sin(theta)+y*cos(theta);
        Y2=x2*sin(theta)+y2*cos(theta);
        Y3=x3*sin(theta)+y3*cos(theta);
        Y4=x4*sin(theta)+y4*cos(theta);
        Z1=z1*cos(theta)-z1*sin(theta);
        Z2=z2*cos(theta)-z2*sin(theta);
        Z3=z3*cos(theta)-z3*sin(theta);
        Z4=z4*cos(theta)-z4*sin(theta);
}
void scale()
{
        float sx,sy,sz;
        printf("Enter sx ,sy & sz values\n");
        scanf("%f%f%f",&sx,&sy,&sz);
        X1=x1*sx;X2=x2*sx;X3=x3*sx;X4=x4*sx;
        Y=y*sy;Y2=y2*sy;Y3=y3*sy;Y4=y4*sy;
        Z1=z1*sz;Z2=z2*sz;Z3=z3*sz;Z4=z4*sz;
}
void init(void)
{
        glClearColor(1.0,0.0,1.0,0.0);
}
int main(int argc,char **argv)
{
        printf("3D transformation operations\n 1:Translation\n2:rotation\n 3:scaling\n");
        printf("enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1: translate();
```

```
                break;
        case 2: rotate();
                break;
        case 3: scale();
                break;
    }
    glutInit(&argc,argv);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("3D");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

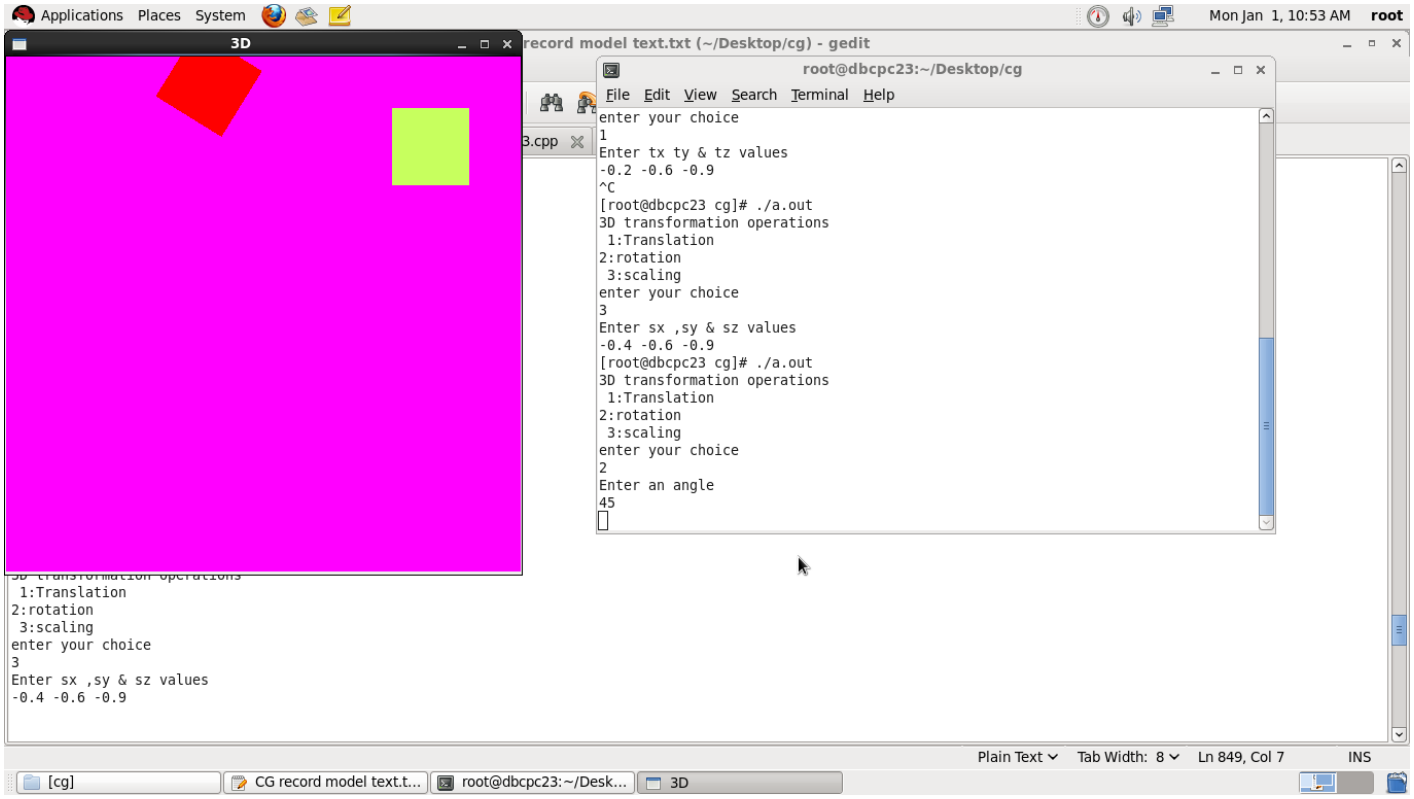
OUTPUT

3D TRANSFORMATION OPERATIONS
1:Translation
2:rotation
3:scaling
enter your choice
1
Enter tx ty & tz values
-0.2 -0.6 -0.9



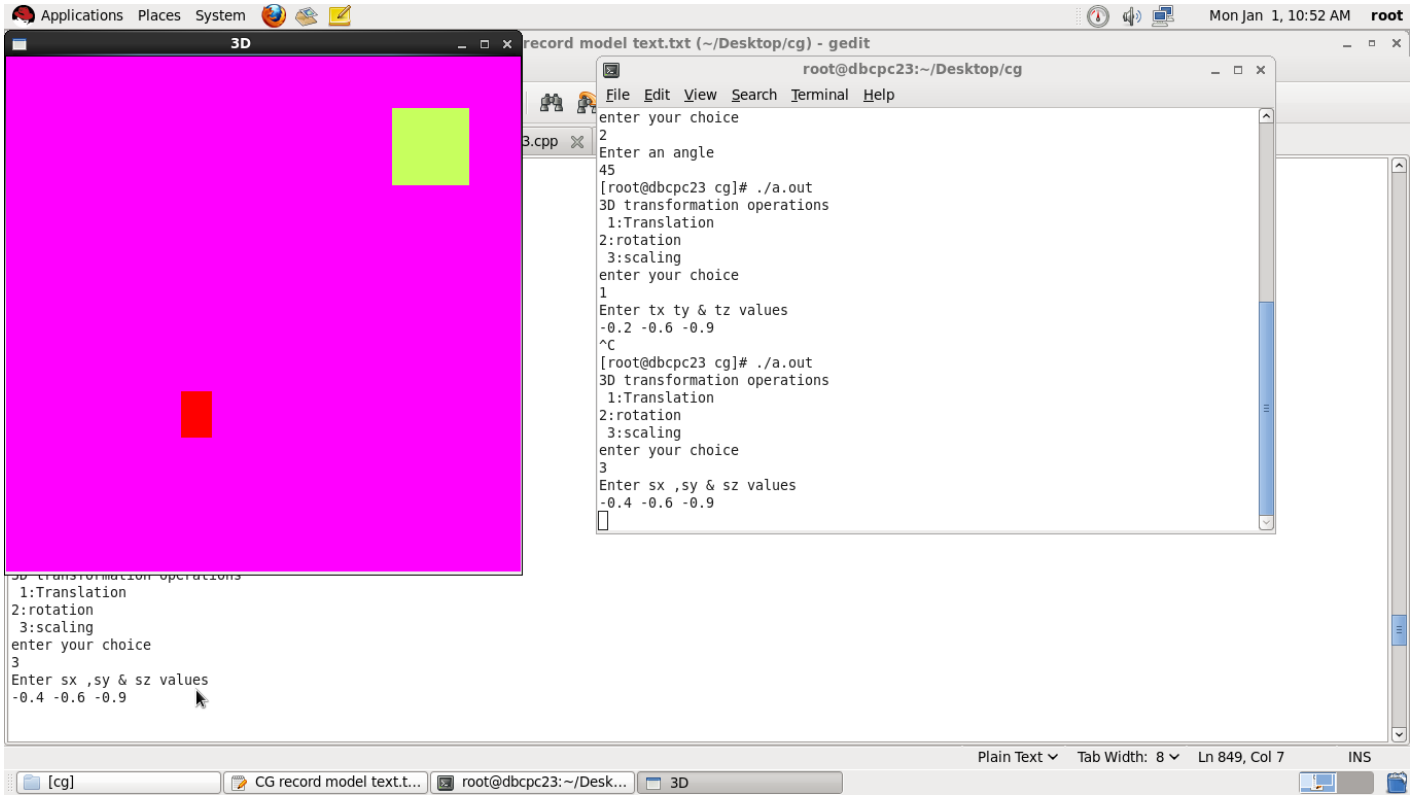
3D TRANSFORMATION OPERATIONS

1:Translation
2:rotation
3:scaling
enter your choice
2
Enter an angle
45



3D TRANSFORMATION OPERATIONS

1:Translation
2:rotation
3:scaling
enter your choice
3
Enter sx ,sy & sz values
-0.4 -0.6 -0.9



PROGRAM NO 12

AIM: Program to implement octahedron.

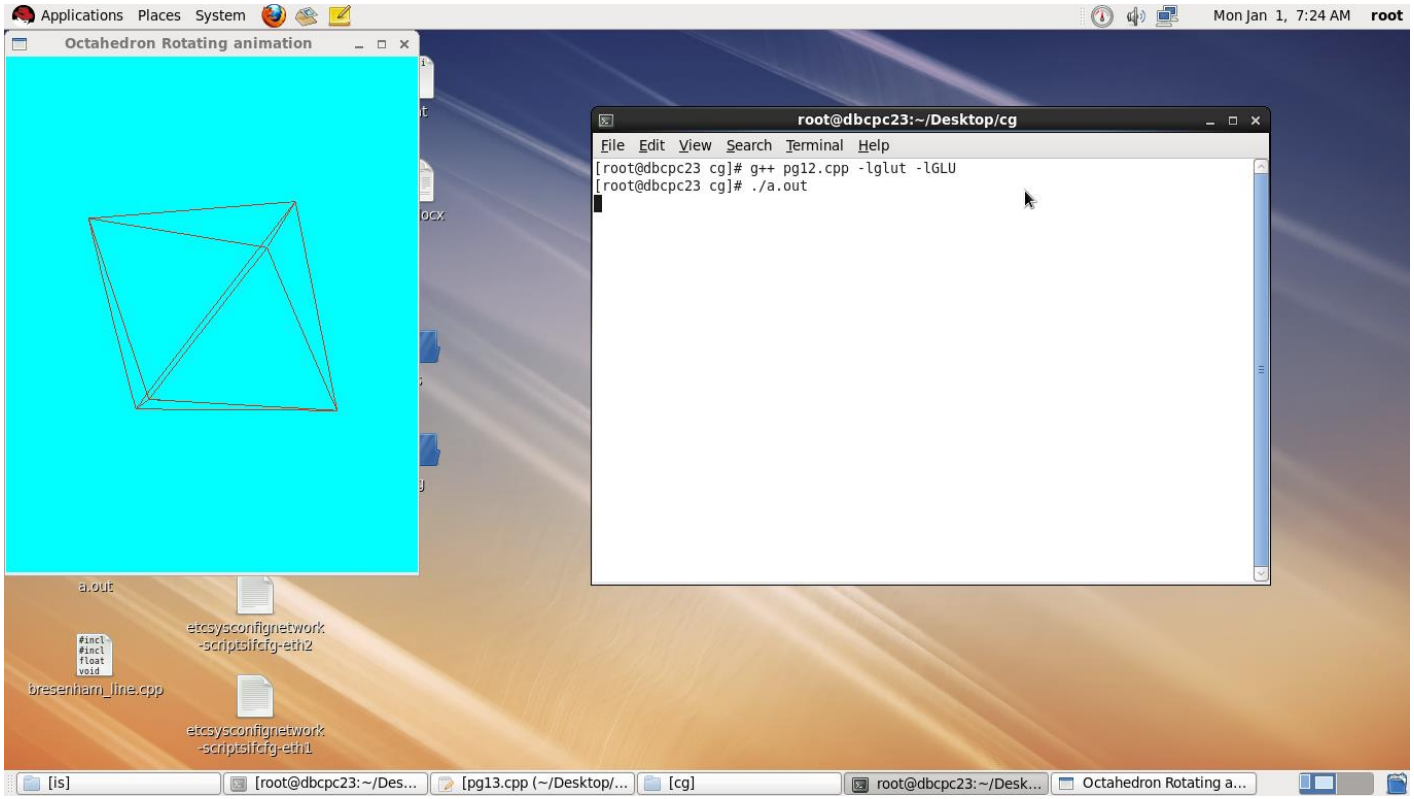
PROGRAM CODE :

```
#include<GL/glu.h>
#include<GL/glut.h>
GLfloat xRotated,yRotated,zRotated;
void displayOctahedron(void)
{
        glMatrixMode(GL_MODELVIEW);
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        glTranslatef(0.0,0.0,-4.5);
        glColor3f(0.8,0.2,0.1);
        glRotatef(xRotated,1.0,0.0,0.0);
        glRotatef(yRotated,0.0,1.0,0.0);
        glRotatef(zRotated,0.0,0.0,1.0);
        glScalef(1.0,1.0,1.0);
        glutSolidOctahedron();
        glFlush();
}
void reshapeOctahedron(int x,int y)
{
        if(y==0||x==0)
                return;
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
        glViewport(0,0,x,y);
}
void idleOctahedron(void)
{
        yRotated+=0.03;
        displayOctahedron();
}
int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(400,500);
        glutCreateWindow("Octahedron Rotating animation");
        glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
```

```
        xRotated=yRotated=zRotated=30.0;
        xRotated=33;
        yRotated=40;
        glClearColor(0.0,1.0,1.0,0.0);
        glutDisplayFunc(displayOctahedron);
        glutReshapeFunc(reshapeOctahedron);
        glutIdleFunc (idleOctahedron);

        glutMainLoop();
        return 0;
}
```

OUTPUT:



PROGRAM NO 13

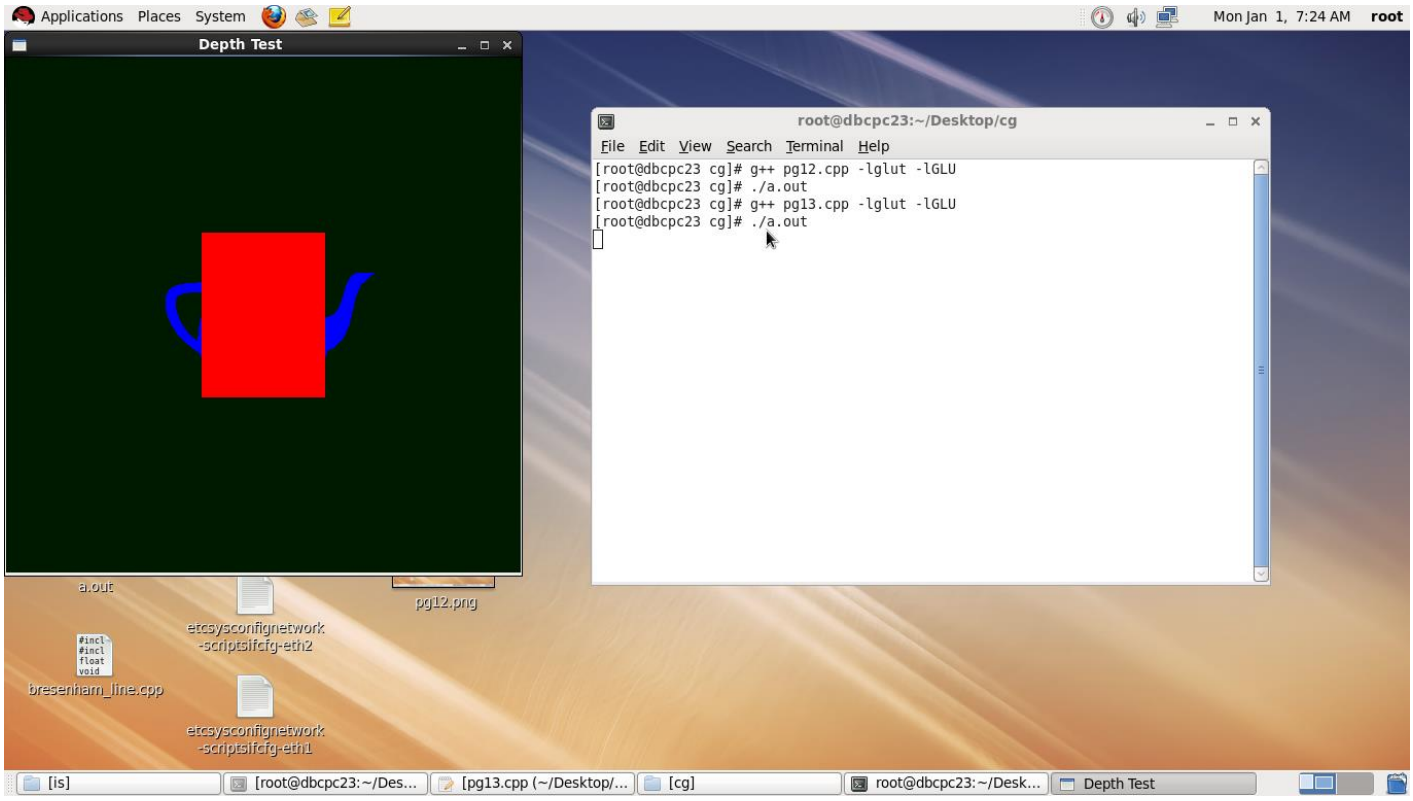
AIM: Program to implement visible surface detection.

PROGRAM CODE :

```
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
void draw()
{
	glClearColor(0.0,0.1,0.0,1.0);
	glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
	glColor3f(1.0,0.0,0.0);
	glutSolidCube(7.0);
	glTranslatef(0.0,0.0,-20.0);
	glColor3f(0.0,0.0,1.0);
	glutSolidTeapot(7.0);
	glutSwapBuffers();
}
int main(int argc,char **argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
	glutInitWindowSize(500,500);
	glutCreateWindow("Depth Test");
	glClearColor(0.0,1.0,0.0,0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(45.0,1.333,0.01,10000.0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glTranslatef(0,0,-30);
	glEnable(GL_DEPTH_TEST);
	glutDisplayFunc(draw);
	glutMainLoop();
	return 0;
```

}

OUTPUT:


Applications Places System
Mon Jan 1, 7:24 AM root
Depth Test
root@dbcpc23:~/Desktop/cg
File Edit View Search Terminal Help
[root@dbcpc23 cg]# g++ pg12.cpp -lglut -lGLU
[root@dbcpc23 cg]# ./a.out
[root@dbcpc23 cg]# g++ pg13.cpp -lglut -lGLU
[root@dbcpc23 cg]# ./a.out
a.out
pg12.png
etcsysconfignetwork-scriptsifcfg-eth2
bresenham_line.cpp
etcsysconfignetwork-scriptsifcfg-eth1
[is]
[root@dbcpc23:~/Des...
[pg13.cpp (~/Desktop/...
[cg]
root@dbcpc23:~/Desk...
Depth Test

PROGRAM NO 14

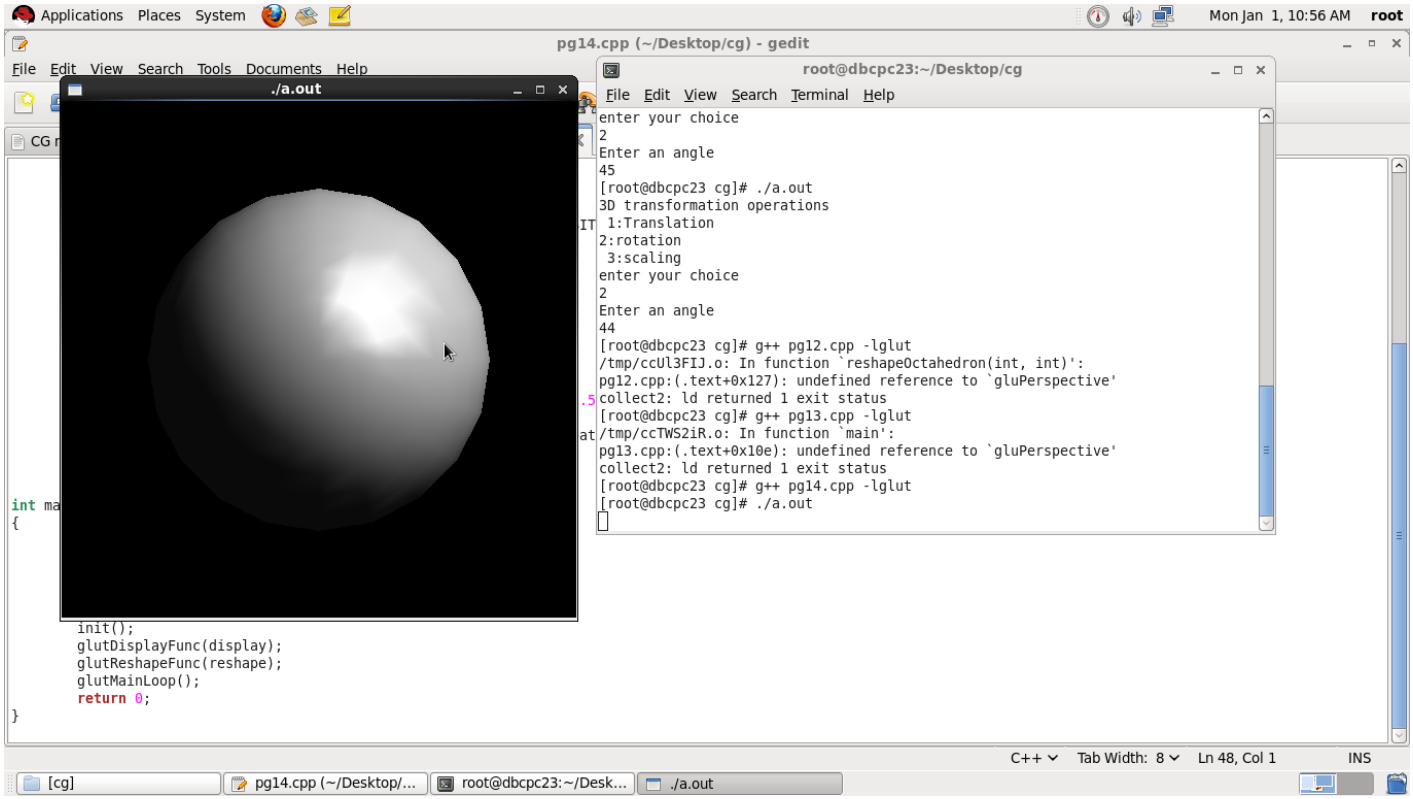
AIM: Program to implement rendering.

PROGRAM CODE :

```
#include<GL/glu.h>
#include<GL/glut.h>
void init(void)
 {
 GLfloat mat_specular[]={1.0,1.0,1.0,1.0};
 GLfloat mat_shininess[]={50.0};
 GLfloat light_position[]={1.0,1.0,1.0,0.0};
 glClearColor(0.0,0.0,0.0,0.0);
 glShadeModel(GL_SMOOTH);
 glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
 glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);
 glLightfv(GL_LIGHT0,GL_POSITION,light_position);
 glEnable(GL_LIGHTING);
 glEnable(GL_LIGHT0);
 glEnable(GL_DEPTH_TEST);
 }
 void display(void)
 {
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
 glutSolidSphere(1.0,20,17);
  glFlush();
 }
 void reshape(int w,int h)
 {
 glViewport(0,0,(GLsizei)w,(GLsizei)h);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 if(w<=h)
 glOrtho(-1.5,1.5,-1.5*(GLfloat)h/(GLfloat)w,1.5*(GLfloat)h/(GLfloat)w,-10.0,10.0);
 else
 glOrtho(-1.5*(GLfloat)w/(GLfloat)h,1.5*(GLfloat)h/(GLfloat)w,-1.5,1.5,-10.0,10.0);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 }
 int main(int argc,char **argv)
 {
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
 glutInitWindowSize(500,500);
 glutInitWindowPosition(100,200);
 glutCreateWindow(argv[0]);
```

```
init();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

OUTPUT :



PRINCIPLES OF INTELLIGENT SYSTEMS

PROGRAM NO 1

AIM: Program to implement perceptron network.

PROGRAM CODE :

```
#include<stdio.h>
main()
{
signed int x[4] [2],tar[4];
float w[2],wc[2],out=0;
int i,j,k=0,h=0;
float s=0,b=0,bc=0,alpha=0;
float theta;
printf("Enter the value of theta & alpha");
scanf ("%f%f",&theta, &alpha);
for(i=0;i<=3;i++)
{
printf("Enter the value of %d Inputrow & Target",i);
for(j=0;j<=1;j++)
{
scanf ("%d",&x[i][j]);
}
scanf ("%d",&tar[i]);
w[i]=0;
wc[i]=0;
}
printf("\tNet\t Target\tWeight changes\tNew weights\t Bias changes\tBias\n");
printf("-----------------------------------------------------\n");
mew:
printf ("ITERATION %d\n", h);
printf("-----------------------------------------------------\n");
for(i=0;i<=3;i++)
{
for(j=0;j<=1;j++)
{
s+=(float)x[i][j]*w[j];
}
s+=b;
printf("%.2f\t",s);
if (s>theta)
out=1;
else if(s<-theta)
out=-1;
printf ( "%d\t", tar[i]);
s=0;
```

```
if (out==tar[i])
{
for(j=0;j<=1;j++)
{
wc[j]=0;
bc=0;
printf ( "%.2f\t" ,wc[j]);
}
for(j=0;j<=1;j++)
printf("%.2f\t",w[j]);
k+=1;
b+=bc;
printf( "%.2f\t\t" ,bc);
printf ( "%.2f\t",b);
}
else
{
for ( j=0; j<=1; j++)
{
wc[j]=x[i] [j]*tar[i]*alpha;
w[j]+=wc[j];
printf("%.2f\t" ,wc[j]);
wc[j]=0;
}
for(j=0;j<=1;j++)
printf ( "% .2f\t" ,w[j]);
bc=tar[i]*alpha;
b+=bc;
printf ( "%.2f\t\t", bc);
printf ( "%.2f\t", b);
}
printf ( "\n");
}
if(k==4)
{
printf ( "\nFinal weights\n" );
for(j=0;j<=1;j++)
{
printf ( "w[%d] =%.2f\t", j ,w[j]);
}
printf("Bias b=%.2f", b);
}
else
{
k=0;
h=h+1;
```

```
goto mew;
} return ;
getch();
}
```

OUTPUT:

```
Enter the value of theta & alpha1 2
Enter the value of 0 Inputrow & Target1 2 1
Enter the value of 1 Inputrow & Target1 2 1
Enter the value of 2 Inputrow & Target1 2 1
Enter the value of 3 Inputrow & Target1 2 1
Net      Target Weight changes        New weights    Bias changes  Bias
-----------------------------------------------------
ITERATION 0
-----------------------------------------------------
0.00     1      2.00    4.00    2.00    4.00   2.00          2.00
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00
ITERATION 1
-----------------------------------------------------
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00
12.00    1      0.00    0.00    2.00    4.00   0.00          2.00

Final weights
w[0] =2.00      w[1] =4.00      Bias b=2.00(base)
```

PROGRAM NO 2

AIM: Program to implement kohonen self organizing feature map.

PROGRAM CODE :

```
#include<stdio.h>
void main()
{
signed int x[4][4];
float w[4][2],d1,d2,o,m=0;
int i,j,k,J;
float alp=0.6;
printf("enter the input");
for(i=0;i<=3;i++)
{
for(j=0;j<=3;j++)
{
scanf("%d",&x[i][j]);
}
}
printf("enter the weight matrix:");
for(i=0;i<=3;i++)
{
for(j=0;j<=1;j++)
{
scanf("%f", &o);
w[i][j]=o;
}
}
mew:
for(i=0;i<=3;i++)
{
for(k=0;k<=1;k++)
{
for(j=0;j<=3;j++)
{
if(k==0)
d1+=(w[j][k]-x[i][j])*(w[j][k]-x[i][j]);
else
d2+=(w[j][k]-x[i][j])*(w[j][k]-x[i][j]);
}
}
if(d1>d2)
J=1;
else
J=0;
```

```
d1=d2=0;
for(j=0;j<=3;j++)
{
w[j][J]+=alp*(x[i][j]-w[j][J]);
}
}
alp=alp/1.014;
if(m>=100)
{
for(i=0;i<=3;i++)
{
for(j=0;j<=1;j++)
{
printf("\n%f\t",w[i][j]);
}
}
}
else
{
m=m+1;
for(i=0;i<=3;i++)
{
for(j=0;j<=1;j++)
{
printf("\n%f\t",w[i][j]);
}
printf("\n");
}
goto mew;
}
}
```

OUTPUT:

```
enter the input0
1
0
0
1
0
0
0
1
0
1
0
1
```

1
0
0
enter the weight matrix:0.1
0.1
0.2
0.1
0.1
0.2
0.3
0.1

0.616000
0.856000

0.872000
0.016000

0.016000
0.632000

0.048000
0.016000

0.694401
0.975996

0.978663
0.002667

0.002667
0.697068

0.008001
0.002667

0.703979
0.995837

0.996299
0.000463

0.000463
0.704441

0.001388
0.000463
0.702353

PROGRAM NO 3

AIM: Program to implement fuzzy logic implementation.

PROGRAM CODE :

```
#include<stdio.h>
#define SIZE 50
int S1[SIZE],S2[SIZE],S3[SIZE],S4[SIZE],S5[SIZE];
void accept(int S[],int n)
{
   int i;
   for(i=1;i<=n;i++)
      scanf("%d",&S[i]);
   S[0]=n;
}

void display(int S[])
{
   int n,i;
   n=S[0];
   printf("{ ");
   for(i=1;i<=n;i++)
      printf("%d ",S[i]);
   printf("}");
}

int ele_pre(int S[],int x)
{
   int n,i;
   n=S[0];
   for(i=1;i<=n;i++)
      if(S[i]==x)
         return 1;
   return 0;
}

void set_union(int S1[],int S2[])
{
   int n,i,m;
   n=S1[0];
   for(i=1;i<=n;i++)
      S3[i]=S1[i];
   m=S2[0];
   for(i=1;i<=m;i++)
      if(!ele_pre(S1,S2[i]))
         S3[++n]=S2[i];
```

```
    S3[0]=n;
}

void intersection(int S1[],int S2[])
{
    int n,i,j=0;
    n=S1[0];
    for(i=1;i<=n;i++)
        if(ele_pre(S2,S1[i]))
            S3[++j]=S1[i];
    S3[0]=j;
}
void complementS1(int S1[])
{
   int i,n;
    n=S1[0];
    S3[0]=0;
    for(i=1;i<=n;i++)
       if(!ele_pre(S2,S1[i]))
         S3[++S3[0]]=S1[i];
}
void complementS2(int S2[])
{
   int i,n;
    n=S2[0];
    S3[0]=0;
    for(i=1;i<=n;i++)
       if(!ele_pre(S2,S1[i]))
         S3[++S3[0]]=S2[i];
}

int main()
{
    int n,m;
    printf("————————————————————————————————————————————————————————————————————————————————");
    printf("\nProgram to perform the Set operations");
    printf("\n————————————————————————————————————————————————————————————————————————————————");
    printf("\nHow many elements for Set-1 ");
    scanf("%d",&n);
    printf("\nEnter elements\n");
    accept(S1,n);

    printf("\nHow many elements for Set-2 ");
    scanf("%d",&m);
```

```
    printf("\nEnter elements\n");
    accept(S2,m);

    set_union(S1,S2);
    printf("\nS1 U S2 = S3 = ");
    display(S3);

    intersection(S1,S2);
    printf("\nS1 ^ S2 = S3 = ");
    display(S3);

    complementS1(S1);
    printf("\nS1' = S3 =");
    display(S3);

    complementS2(S2);
    printf("\nS2' = S3 =");
    display(S3);

    printf("\n----------\n");
}
```

OUTPUT:

```
_____  _____
Program to perform the Set operations
_____ _____
How many elements for Set-1 3

Enter elements
12 13 14

How many elements for Set-2 4

Enter elements
12 13 16 18

S1 U S2 = S3 = { 12 13 14 16 18 }
S1 ^ S2 = S3 = { 12 13 }
S1' = S3 ={ 14 }
S2' = S3 ={ 16 18 }
_____ _____ ____
```

PROGRAM NO 4

AIM: Program to perform cartesian product over two given fuzzy sets.

PROGRAM CODE :

```
#include<stdio.h>
#include<limits.h>
#define min(x,y) (x<y?x:y)
struct SET
{
float nr[5];
float dr[5];
int n;
};
typedef struct SET fuzzy;
void printval(fuzzy *m,char *x)
{
int i;
printf("\n %s={",x);
for(i=0;i<m->n;i++)
{
printf("%5.2f/%5.2f",m->nr[i],m->dr[i]);
if(i!=m->n-1)
putchar('+');
}
printf(" }");
}
void main()
{
fuzzy V,I;
int i,j;
float P[6][6];
V.n=I.n=5;
V.nr[0]=0.2;
V.nr[1]=0.8;
V.nr[2]=1;
V.nr[3]=0.9;
V.nr[4]=0.7;
V.dr[0]=30;
V.dr[1]=45;
V.dr[2]=60;
V.dr[3]=75;
V.dr[4]=90;
```

```
I.nr[0]=0.4;
I.nr[1]=0.7;
I.nr[2]=1;
I.nr[3]=0.8;
I.nr[4]=0.6;
I.dr[0]=0.8;
I.dr[1]=0.9;
I.dr[2]=1;
I.dr[3]=1.1;
I.dr[4]=1.2;
printval(&V,"V");
printval(&I,"I");
printf("\n");
for(i=0;i<=V.n;i++)
for(j=0;j<=I.n;j++)
{
if(i==0 && j>0)
P[i][j] =I.dr[j-1];
else if(j==0 && i>0)
P[i][j]=V.dr[i-1];
else if(i>0 && j>0)
P[i][j]=min(V.nr[i-1],I.nr[j-1]);
}
for(i=0;i<=V.n;i++)
{
for(j=0;j<=I.n;j++)
{
if(i==0 && j==0)
printf(" ");
else
printf("%6.2f",P[i][j]);
}
printf("\n");
}
}
```

OUTPUT:

```
V={ 0.20/30.00+ 0.80/45.00+ 1.00/60.00+ 0.90/75.00+ 0.70/90.00 }
 I={ 0.40/ 0.80+ 0.70/ 0.90+ 1.00/ 1.00+ 0.80/ 1.10+ 0.60/ 1.20 }
   0.80  0.90  1.00  1.10  1.20
 30.00  0.20  0.20  0.20  0.20  0.20
 45.00  0.40  0.70  0.80  0.80  0.60
 60.00  0.40  0.70  1.00  0.80  0.60
 75.00  0.40  0.70  0.90  0.80  0.60
 90.00  0.40  0.70  0.70  0.70  0.60
```