Certificate

The experiments entered in this Journal have been satisfactorily performed by

Master/Miss VEDIKA AGARWAL

Studying in R.V. College of Engineering

School/College

Class 4th year Div. C Roll No. _____

During the academic year 2020 - 21

Examiner's signature

Teacher's signature

Principal's signature

Date _____

School/College stamp

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 1: Write a program to generate a line using Bresenham's line drawing technique. Consider slopes greater than one and slopes less than one. User can specify inputs through keyboard/mouse

```
#define BLACK 0
#include <stdio.h>
#include <GL/glut.h>

int x1, x2, y1, y2;

void draw_pixel(int x, int y, int value)
{
    glBegin(GL_POINTS);
    glVertex2i(x,y);
    glEnd();
}

void bres(int x1, int x2, int y1, int y2)
{
    int dx, dy, i, e;

    int incx, incy, inc1, inc2;
    int x, y;
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
dx= x2-x1;
dy= y2-y1;
if (dx<0) dx = -dx;
if (dy<0) dy = -dy;
incx=1;
if(x2<x1) incx=-1;
incy=1;
if(y2<y1) incy=-1;
x= x1;  y=y1;
if(dx>dy)
    {
        draw_pixel (x,y, BLACK);
        e= 2*dy-dx;
        inc1 = 2*(dy-dx);
        inc2 = 2*dy;
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
for(i=0; i<dx; i++)
{
    if(e>=0)
    {
        y+= incy;
        e+= inc1;
    }
    else  e+= inc2;
          x+= incx;
          draw_pixel(x, y, BLACK);
}
}
else
{
    draw_pixel(x, y, BLACK);
    e= 2*dx-dy;
    inc1 = 2*(dx-dy);
    inc2 = 2*dx;
    for(i=0; i<dy; i++)
    {
        if(e>=0)
        {
            x+= incx;
            e+= inc1;
        }
```

Teacher's Signature:

```
        else    et = inc2;
                yt = incy;

                draw_pixel(x, y, BLACK);
            }
        }
    }

Void display()
    {
        glClear(GL_COLOR_BUFFER_BIT);

        bres(x1, y1, x2, y2);
        glFlush();
    }

Void myinit()
    {
        glClearColor(1.0, 1.0, 1.0, 1.0);

        glColor3f(1.0, 0.0, 0.0);
        glPointSize(1.0);

        glMatrixMode(GL_PROJECTION);

        glLoadIdentity();

        glOrtho2D(0.0, 499.0, 0.0, 499.0);
    }
```

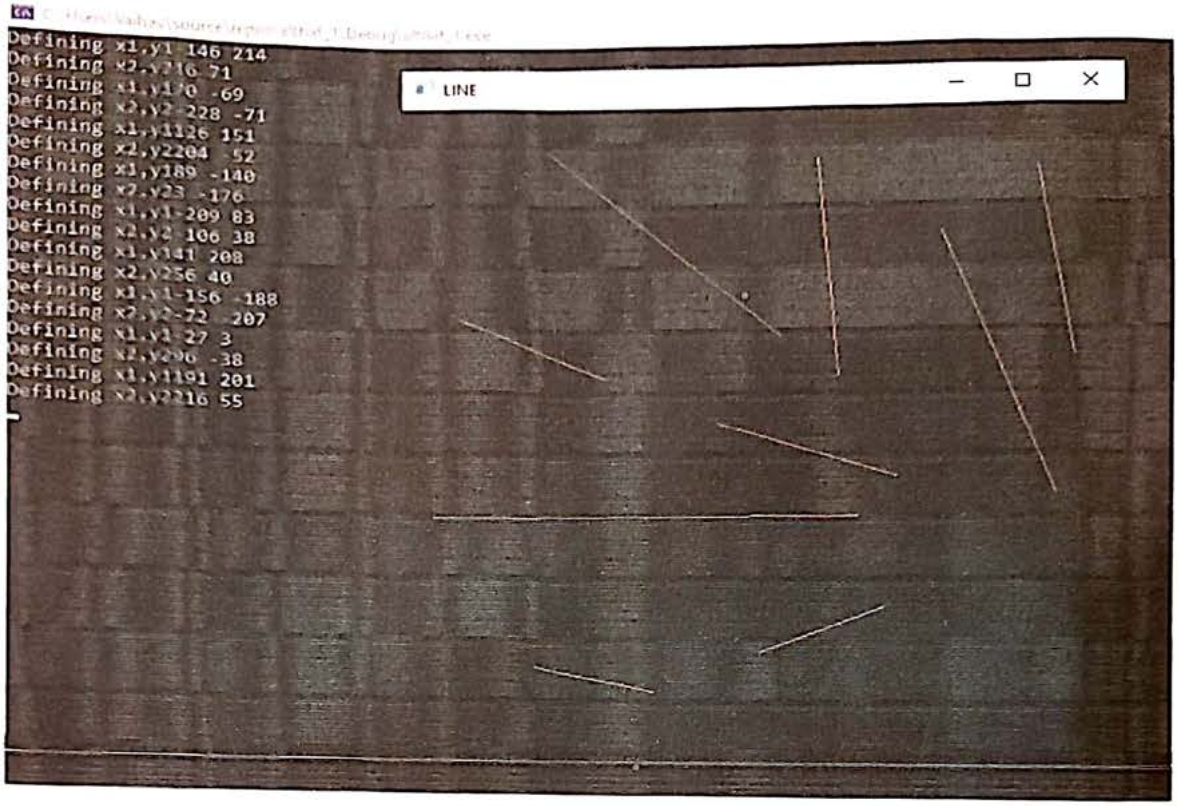
Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

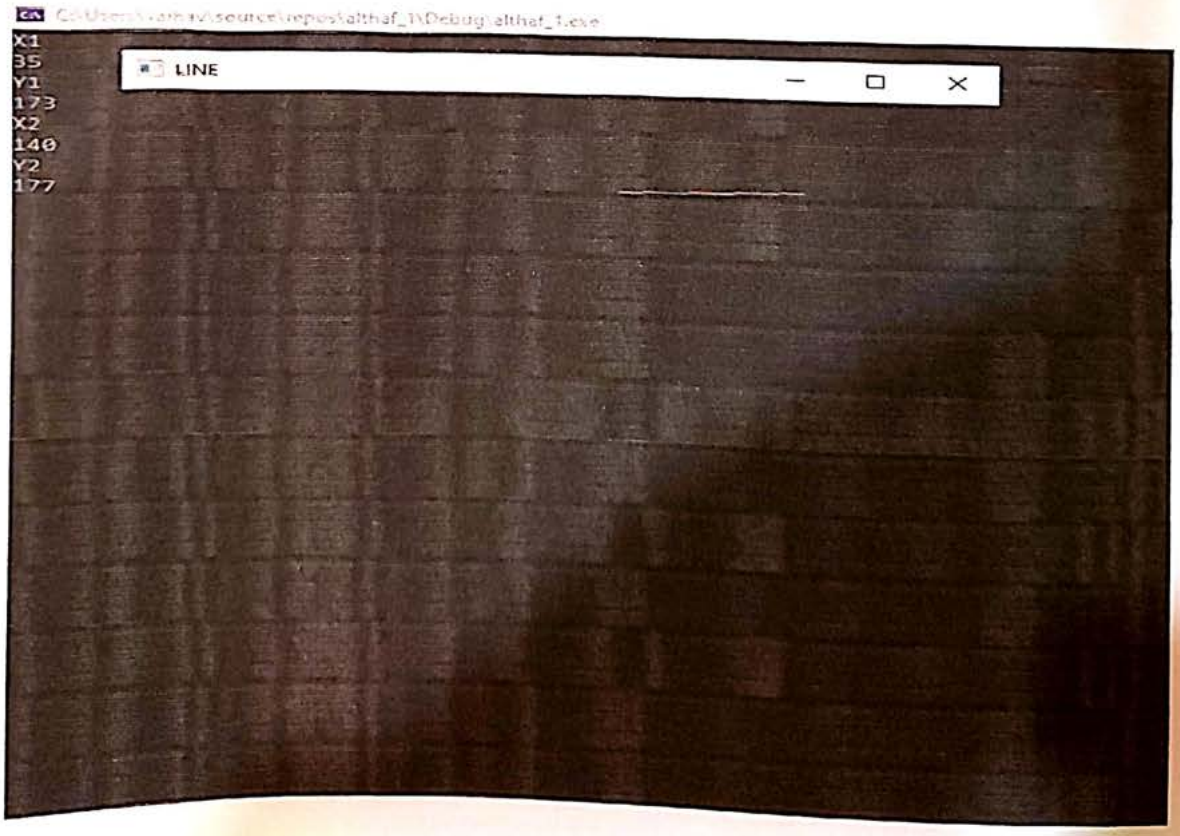
```
void main(int argc, char **argv)
{
    printf("Enter points : x1, y1, x2, y2");
    scanf("%d %d %d %d", &x1, &x2, &y1, &y2);
    Point[1] = y1;
    Point[2] = x2;
    Point[3] = y2;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Bresenham's Algorithm");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

Teacher's Signature:

Input through mouse:



Input through keyboard



EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 2: Write a program to generate circle using Bresenham's circle drawing technique. User can specify inputs through keyboard/mouse.

```
#include <stdio.h>
#include <gl/glut.h>
int xc, yc, r;
void drawCircle(int xc, int yc, int x, int y)
{
    glBegin(GL_POINTS);
    glVertex2i(xc + x, yc + y);
    glVertex2i(xc - x, yc + y);
    glVertex2i(xc + x, yc - y);
    glVertex2i(xc - x, yc - y);
    glVertex2i(xc + y, yc + x);
    glVertex2i(xc - y, yc + x);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
glVertex2i(xc+y, yc-x);
glVertex2i(xc-y, yc-x);
glEnd();
}

Void circleBres(int xc, int yc, int r)
{
    int x=0, y=r;
    int d=3-2*r;
    while(x<y)
    {
        drawCircle(xc, yc, x, y);
        x++;
        if(d<0)
            d=d+4*x+6;
        else
        {
            y--;
            d=d+4*(x-y)+10;
        }
        drawCircle(xc, yc, x, y);
    }
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
void display()
{
    int j;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_
                                  BUFFER_BIT);

    circleBres(xc, yc, r);
    glFlush();
}

void myinit()
{
    glClearColor(1.0, 1.0, 1.0, 0);
    glColor3f(0, 0, 1.0);

    glPointSize(5.0);

    gluOrtho2D(0.0, 500, 0.0, 500);

}

void main(int argc, char *argv[])
{
    int j;

    printf("Enter coord of centre of circle &
            radius");
```

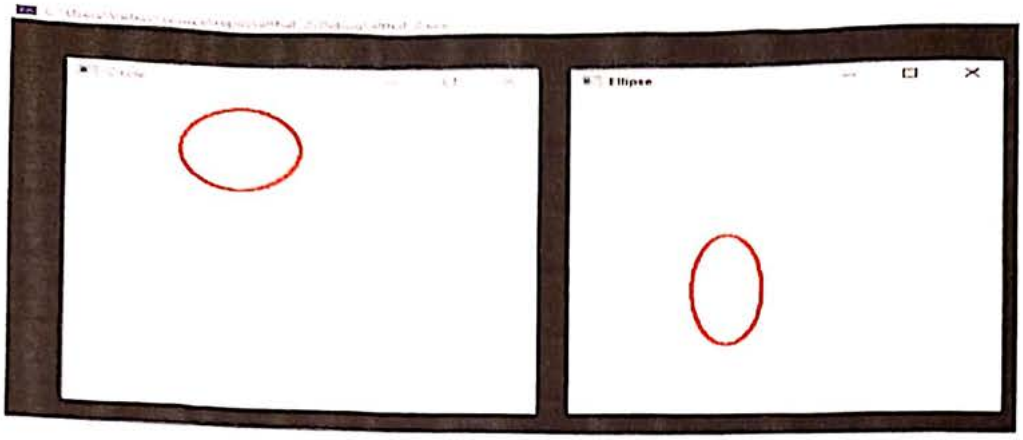
Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

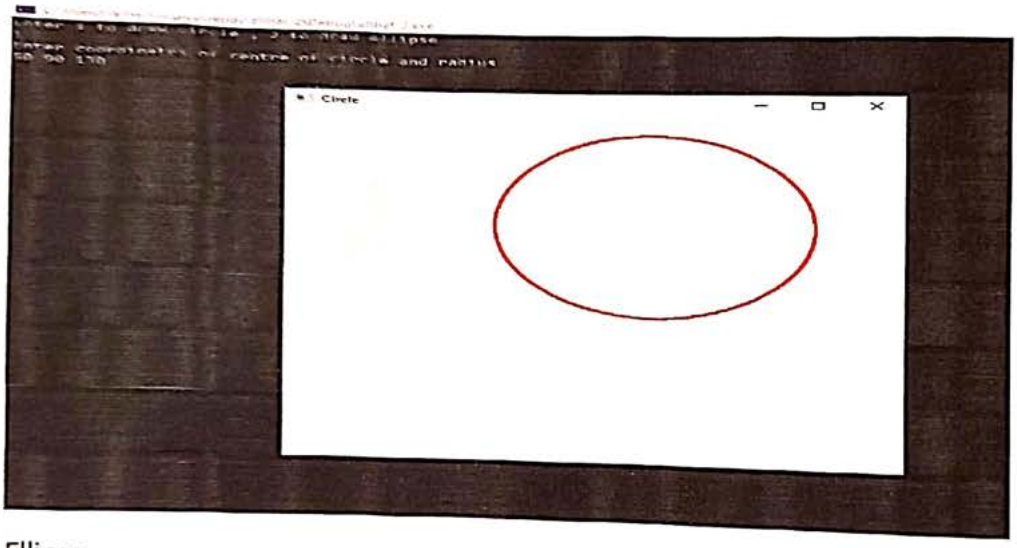
```
scanf("%d %d %d", &xc, &yc, &r);
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB |
                    GLUT_DEPTH);
glutInitWindowSize(500, 500);
glutInitWindowPosition(100, 100);
glutCreateWindow("Bresenham's Circle");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```

Teacher's Signature:

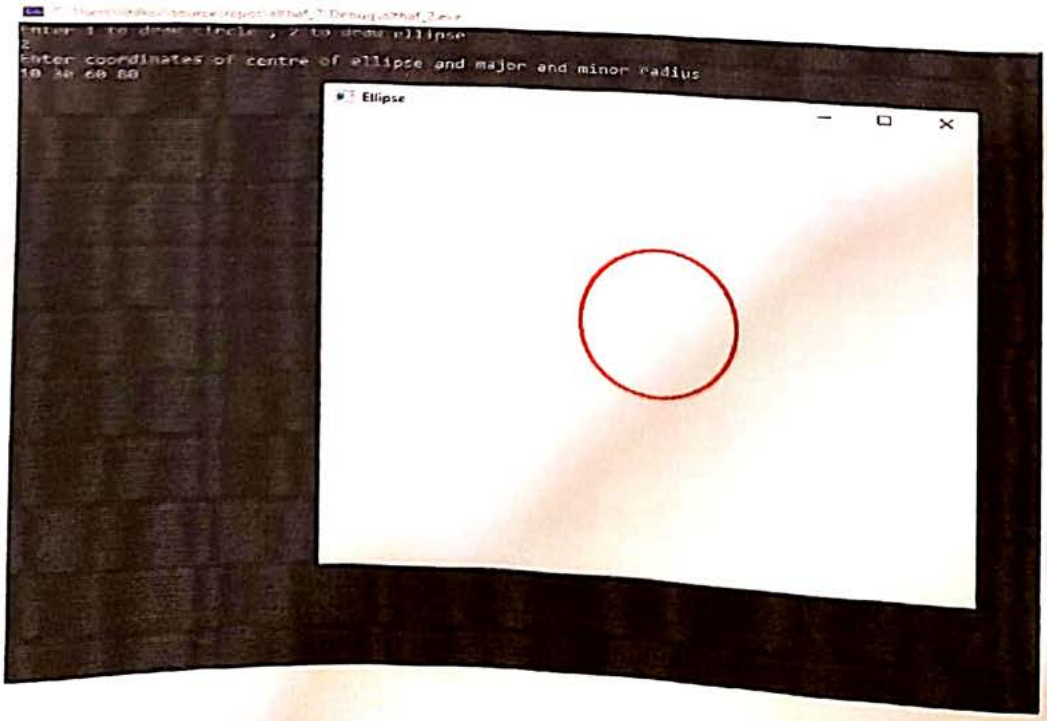
Input through mouse



Input through keyboard :Circle



Ellipse



EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 3: Write a program to recursively subdivide a tetrahedron to form 3D Sierpinski Gasket. The number of recursive steps is to be specified at execution time.

```
#include <GL/glut.h>
typedef GLfloat point[3];
point v[] = {{30.0, 50.0, 100.0}, {0.0, 450.0, -150.0},
             {-350.0, -400.0, -150.0}, {350.0, -400.0,
              -150.0}};
int n;
void triangle(point a, point b, point c)
{
    glBegin(GL_TRIANGLES);
    glVertex3fv(a);
    glVertex3fv(B);
    glVertex3f(c);
```

Teacher's Signature:

```
    glEnd();
}
void divide_triangle(point a, point b, point c, int m)
{
    point v0, v1, v2;
    int j;
    if(m>0)
    {
    for(j=0; j<3; j++) v0[j]=(a[j]+b[j])/2;
    for(j=0; j<3; j++) v1[j]=(a[j]+c[j])/2;
    for(j=0; j<3; j++) v2[j]=(b[j]+c[j])/2;
    divide_triangle(a, v0, v1, m-1);
    divide_triangle(c, v1, v2, m-1);
    divide_triangle(b, v2, v0, m-1);
    }
    else(triangle(a,b,c));
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

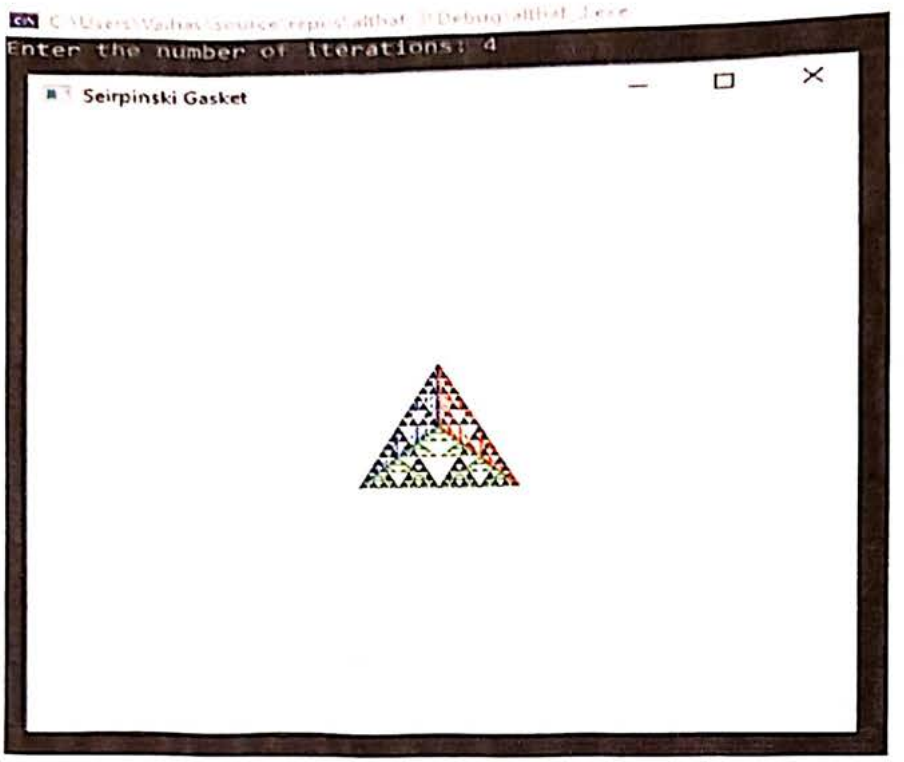
```
void tetra (int m)
{
    glColor3f(1.0, 0.0, 0.0);
    divide_triangle(v[0], v[1], v[2], m);
    glColor3f(0.0, 1.0, 0.0);
    divide_triangle(v[3], v[2], v[1], m);
    glColor3f(0.0, 0.0, 1.0);
    divide_triangle(v[0], v[3], v[1], m);
    glColor3f(0.0, 0.0, 0.0);
    divide_triangle(v[0], v[2], v[3], m);
}
void display()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    tetra(m);
    glFlush();
}
```

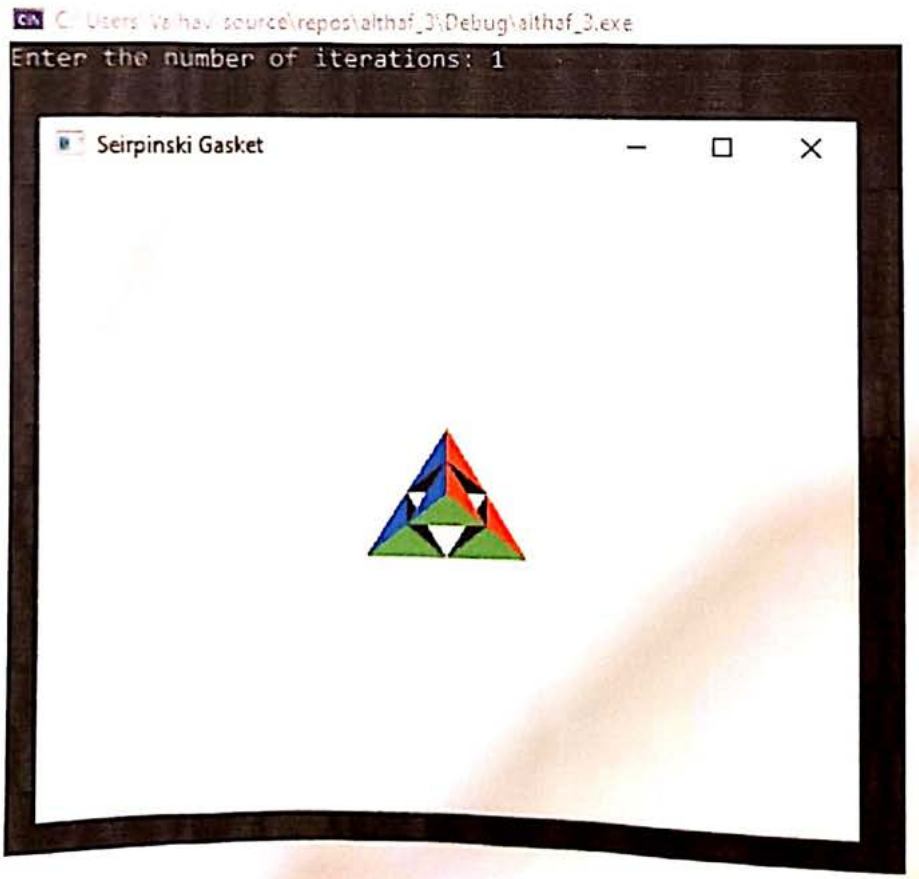
Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
void myinit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-499.0, 499.0, -499.0, 499.0, -499.0, 499.0);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char **argv)
{
    n=5;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("3D Gasket");
    glutDispkyFunc(display);
    myinit();
    glutMainLoop();
}
```

Teacher's Signature:





EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 4: Write a program to fill any given polygon using scan-line area filling algorithm.

```
#include <GL/glut.h>
#include <windows.h>
float x1, x2, x3, x4, y1, y2, y3, y4;
void edgedetect (float x1, float y1, float x2, float y2,
                 int * le, int * re)
{
    float mx, x, temp;
    int i;
    if((y2-y1)<0)
    {
        temp = y1; y1= y2; y2= temp;
        temp= x1;  x1= x2;  x2= temp;
    }
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
if ((y2-y1)!=0)
    mx = (x2-x1)/(y2-y1);
else
    mx = x2-x1;
x = x1;
for(i= y1, i<= y2; i++)
  {
    if(x<(float)le[i])
        le[i] = (int)x;
    if(x>(float)re[i])
        re[i] = (int)x;
    x += mx;
  }
}
void draw_pixel(int x, int y)
  {
    glColor3f(1.0, 0.0, 0.0);
    Sleep(10);
    glBegin(GL_POINTS);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
glVertex2i (x, y);
glEnd();
glFlush();
}

void scanfill(float x1, float y1, float x2, float y2,
              float x3, float y3, float x4,
              float y4)
{
    int le[500], re[500], i, j;
    for (i=0; i<500; i++)
    {
        le[i] = 500;
        re[i] = 0;
    }
edgedetect(x1, y1, x2, y2, le, re);
edgedetect (x2, y2, x3, y3, le, re);
edgedetect (x3, y3, x4, y4, le, re);
edgedetect (x4, y4, x1, y1, le, re);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

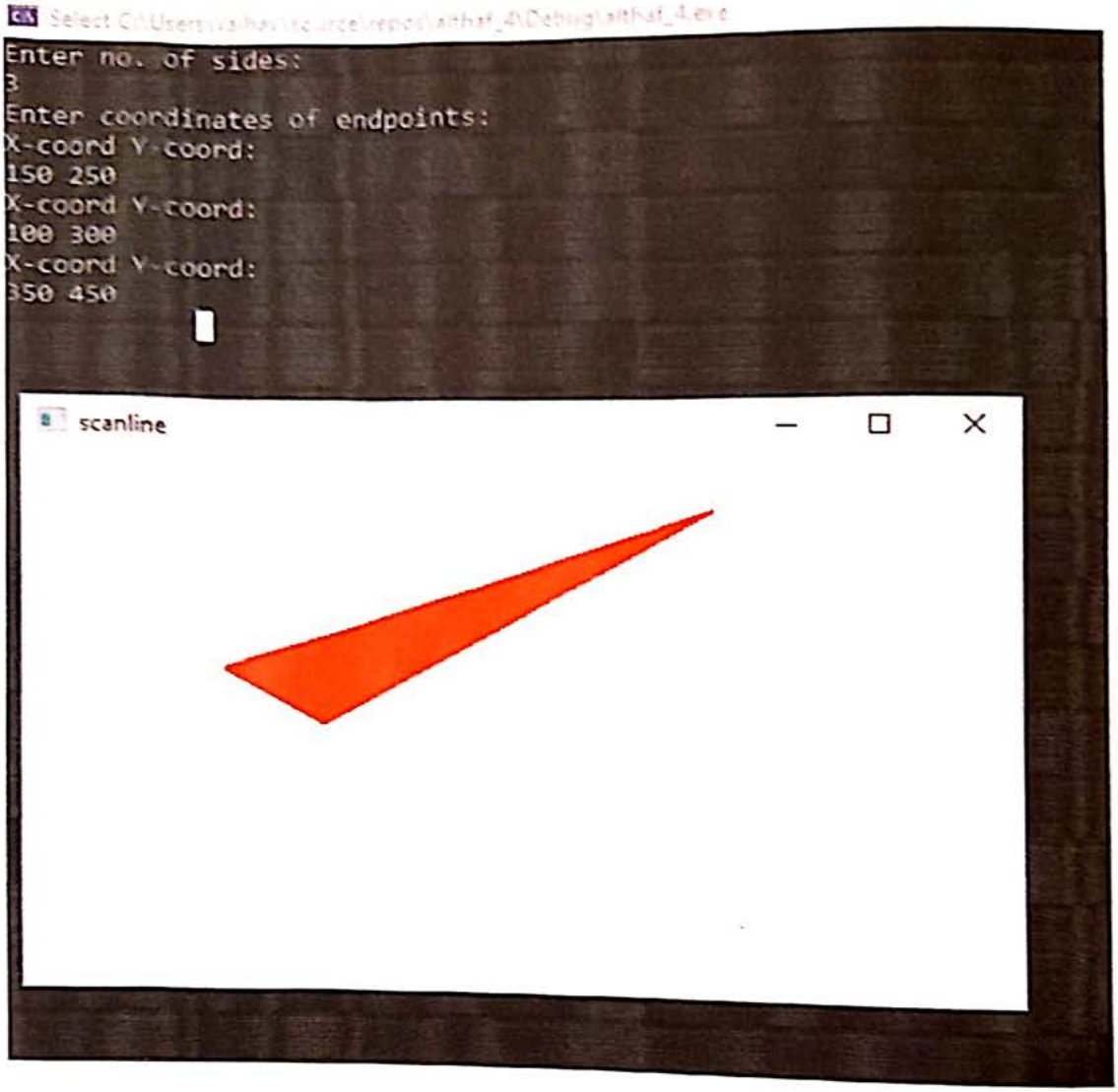
```
for(y = 0; y<500; y++)
{
    if(le[y] <= re[y])
        for(i= (int)le[y]; i<(int)re[y]; i++)
            draw_pixel(i,y);
}
}

Void display()
{
    x1= 200.0; y1= 200.0; x2= 100.0; y2=300.0;
    x3= 200.0; y3= 400.0; x4= 300.0; y4=300.0;

glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
    glVertex2f(x4, y4);
    glEnd();
    scanfill(x1, y1, x2, y2, x3, y3 x4, y4);
    glFlush();
}
void myinit()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 499.0, 0.0, 499.0);
}
void main(int argc, char *argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Filling a Polygon using Scan-line Algorithm");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
```

Teacher's Signature:



EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 5: Write a program to create a house like figure & rotate it about a given fixed point using OpenGL/CUDA transformation functions.

```
#include <gl/glut.h>
#include <math.h>
#include <stdio.h>
float house[11][2] = {{100,200}, {200,250}, {300,200},
                      {100,200}, {100,100}, {175,100},
                      {175,150}, {225,100}, {225,150},
                      {300,100}, {300,200}};
int angle;
float m, c, theta;
void display()
{
    glClearColor(1,1,1,0);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_
        BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho2D(-450, 450, -450, 450);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glColor3f(1, 0, 0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++)
        glVertex2f(house[i]);
glEnd();
glFlush();
glPushMatrix();
glTranslatef(100, 100, 0);
```

Teacher's Signature:

```
glRotatef(angle, 0, 0, 1);
glTranslatef(-100, -100, 0);
glColor3f(1, 1, 0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++)
        glVertex2fv(house[i]);
glEnd();
glPopMatrix();
glFlush();
}

void display2()
{
glClearColor(1, 1, 1, 0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_
                                BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
gluOrtho2D(-450, 450, -450, 450);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glColor3f(1, 0, 0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++)
        glVertex2fv(house[i]);
glEnd();
glFlush();
float x1= 0, x2=500;
float y1= m* x1 +c;
float y2= m* x2 +c;
glColor3f(1, 1, 0);
glBegin(GL_LINES);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
glVertex2f(x1,y1);
glVertex2f(x2,y2);
glEnd();
glFlush();
// Reflected
glPushMatrix();
glTranslatef(0,c,0);
theta = atan(m);
theta= theta * 180/3.14;
glRotatef(theta, 0, 0,1);
glScalef(1,-1,1);
glRotatef(-theta, 0, 0, 1);
glTranslatef(0,-c, 0);
glBegin(GL_LINE_LOOP);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
for (int i=0; i<11; i++)
        glVertex2fv(house[i]);
glEnd();
glPopMatrix();
glFlush();
}

void myInit(){
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);
    glLineWidth(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-450, 450, -450, 450);
}
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

Date:

YOUVA

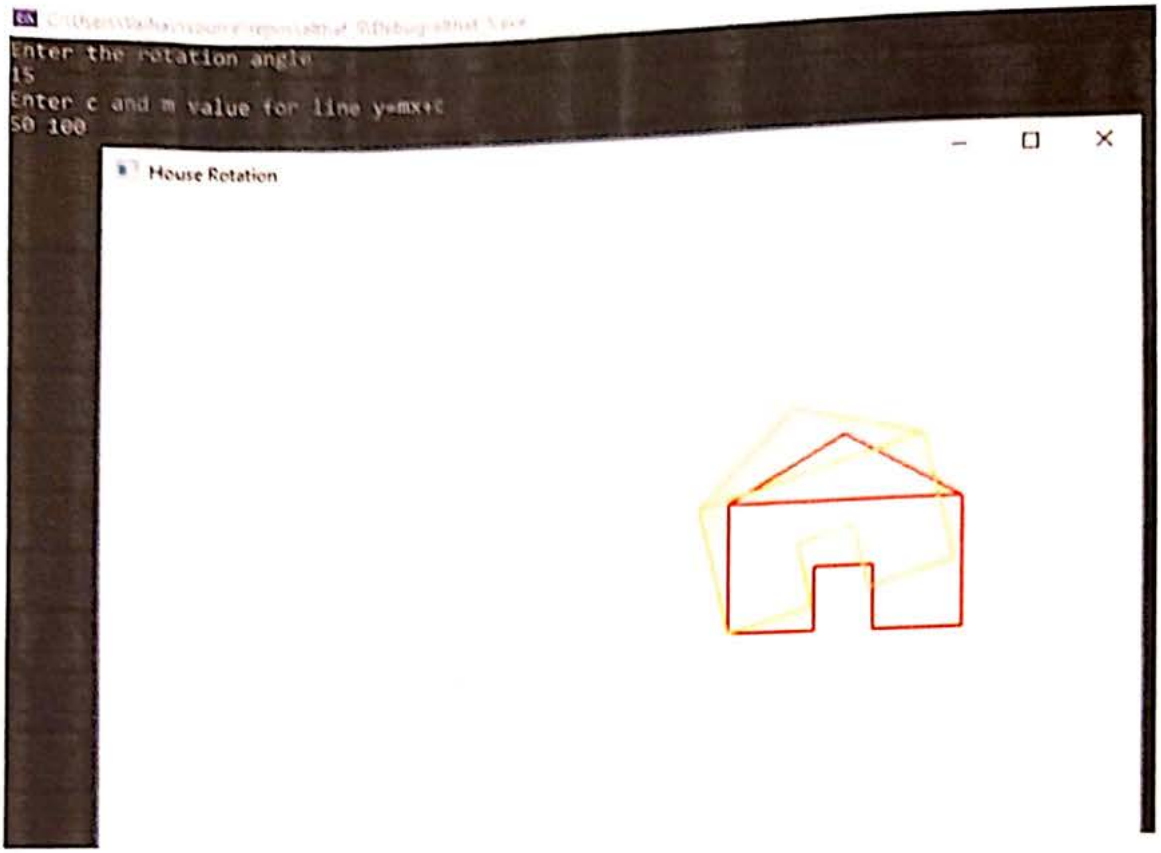
```
void mouse(int btn, int state, int x, int y){
    if(btn == GLUT_LEFT_BUTTON &&
       state == GLUT_DOWN){
        display();
    }
    else if(btn == GLUT_RIGHT_BUTTON &&
            state == GLUT_DOWN){
        display2();
    }}

void main(int argc, char ** argv)
{
    printf("Enter the rotation angle \n");
    scanf("%d", &angle);
    printf("Enter c and m value for
            line y = mx+c \n");
    scanf("%f %f", &c, &m);
    glutInit(&argc, argv);
```

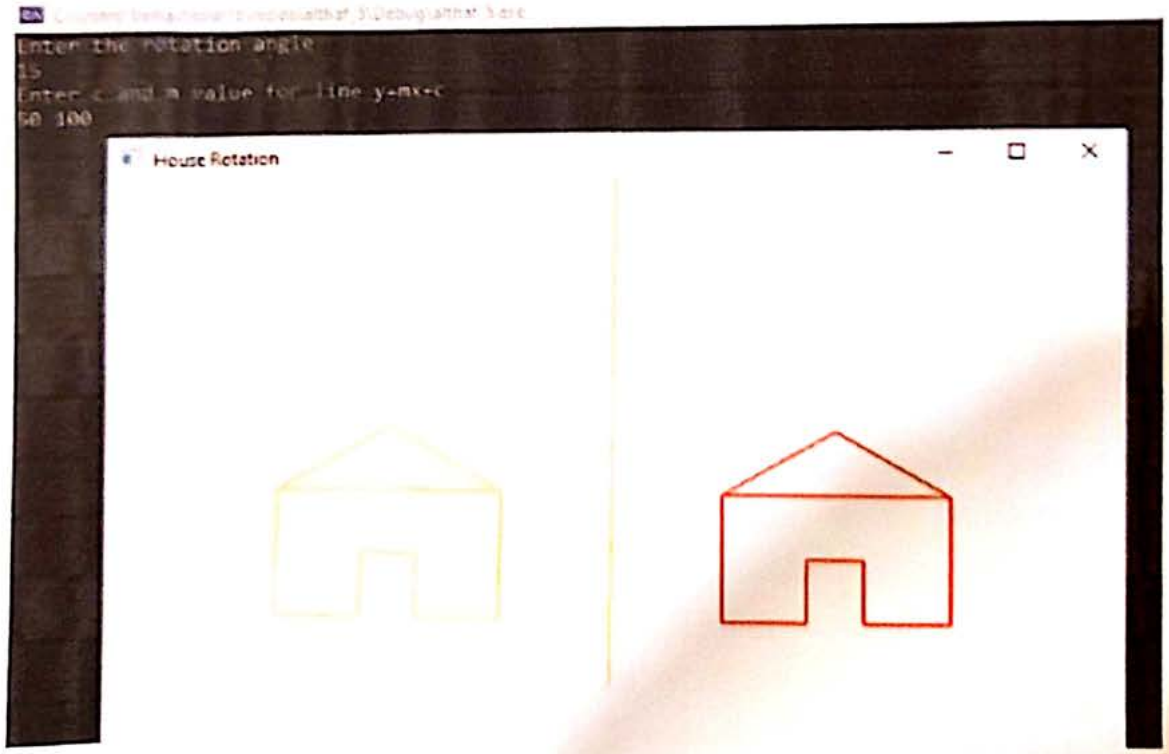
Teacher's Signature:

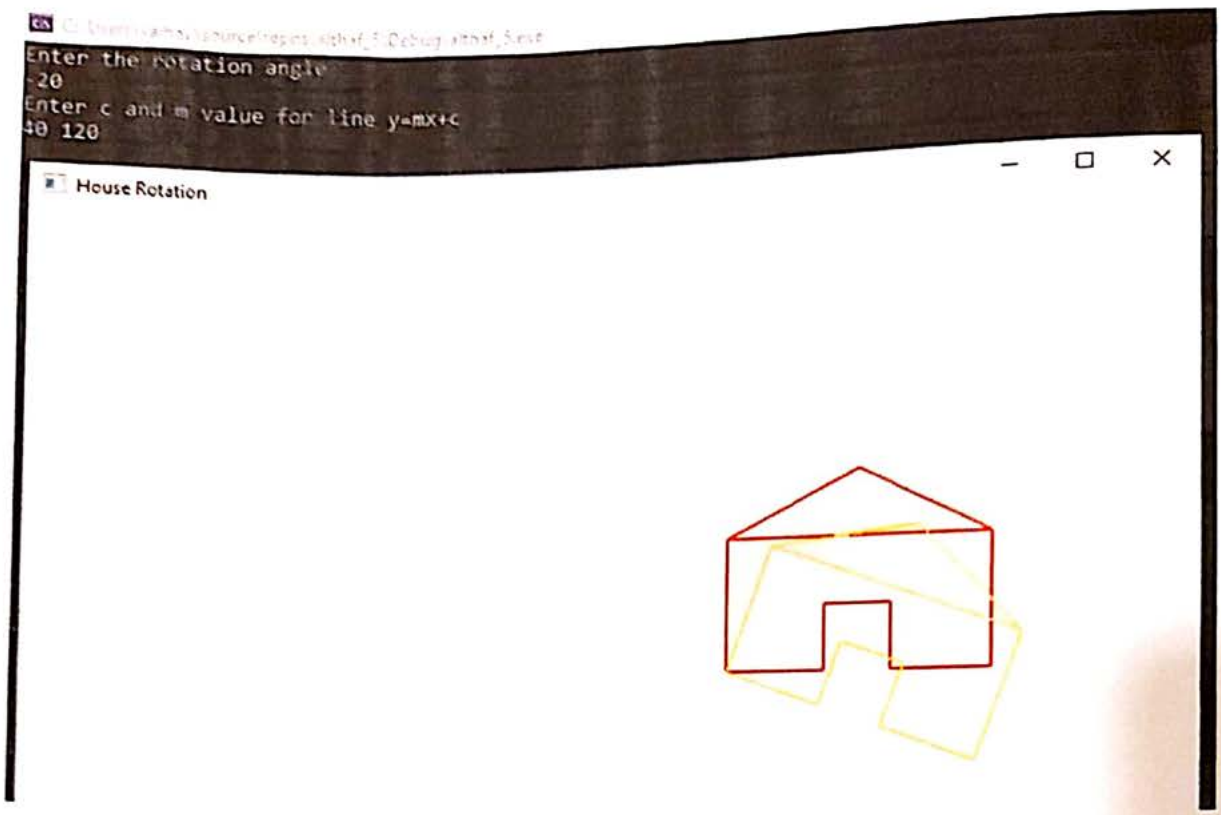
EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

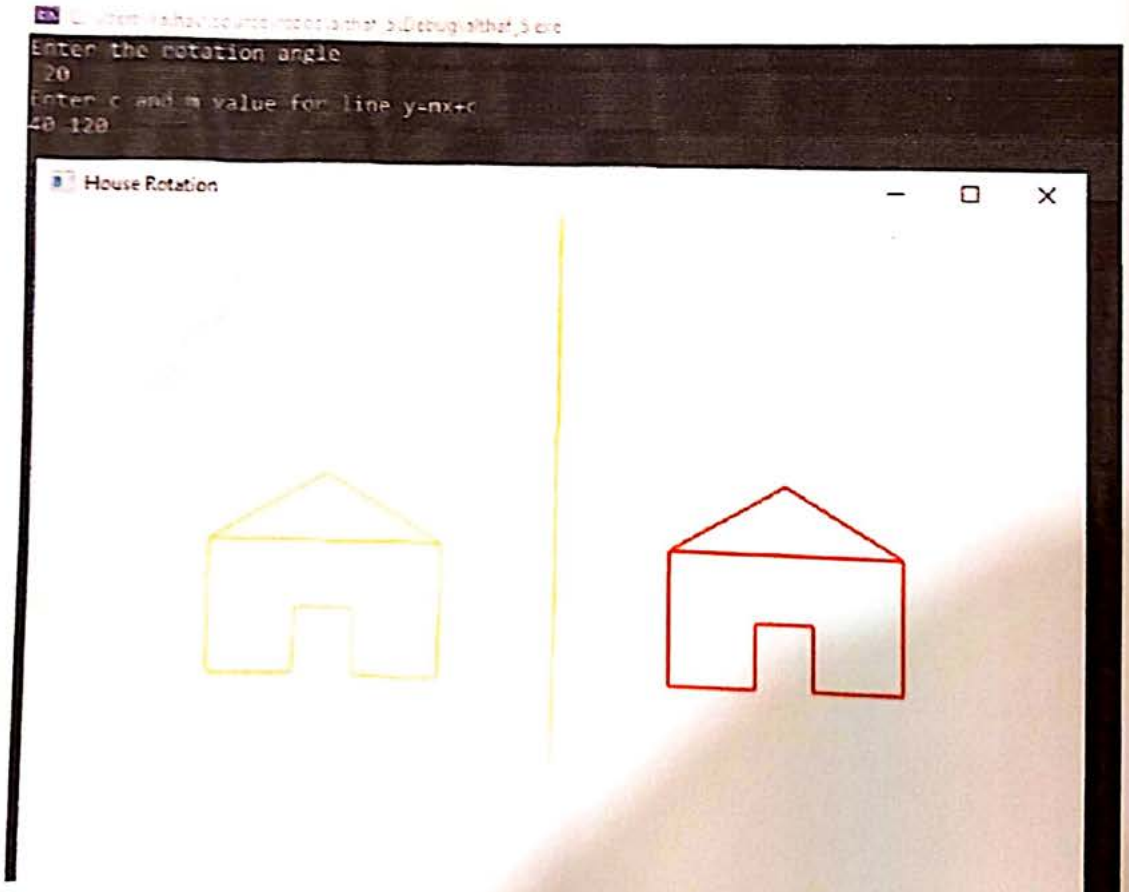
```
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(900, 900);
glutInitWindowPosition(100, 100);
glutCreateWindow("House Rotation");
glutDisplayFunc(display);
glutMouseFunc(mouse);
myInit();
glutMainLoop();
}
```

Teacher's Signature:









EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 6: Write a program to implement the Cohen-Sutherland line clipping algorithm. Make provision to specify the input for multiple lines, window for clipping and viewport for displaying the clipped image.

```
#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);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
void CohenSutherlandLineClipAndDraw(double x0, double y0,
                                    double x1, double y1)
{
    outcode outcode0, outcode1, outcodeOut;
    bool accept = false, done = false;
    outcode0 = ComputeOutCode(x0, y0);
    outcode1 = ComputeOutCode(x1, y1);
    do {
        if (!(outcode0 | outcode1))
        {
            accept = true;
            done = true;
        }
        else if (outcode0 & outcode1)
            done = true;
        else
        {
            double x, y;
            outcodeOut = outcode0 ? outcode0 : outcode1;
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
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;
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
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;
```

Teacher's Signature:

XPT. NO.

NAME:

Page No.:

Date:

YOUVA

```
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, 0.0, 1.0);
glBegin(GL_LINES);
glVertex2d(vx0, vy0);
glVertex2d(vx1, vy1);
glEnd();
}
}
```

Teacher's Signature:

EXPT. NO. NAME:

Page No.:
Date:

YOUVA

```
outcode ComputeOutCode (double x, double y)
{
    outcode code = 0
    if (y > ymax)
        code |= TOP;
    else if (y < ymin)
        code |= BOTTOM;
    if (x > xmax)
        code |= RIGHT;
    else if (x < xmin)
        code |= LEFT;
    return code;
}
void display()
{
    double x0=60, y0=20, x1=80, y1=120;
    double x00=65, y00=65, x11=80, y11=80;
    double x02=25, y02=20, x12=30, y12=30;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2d(x0, y0);
    glVertex2d(x1, y1);
    glEnd();

    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xmin, ymin);
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

Date:

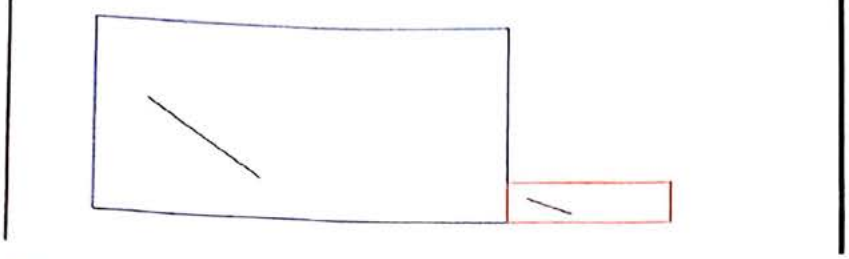
YOUVA

```
    glVertex2f(xmax, ymin);
    glVertex2f(xmax, ymax);
    glVertex2f(xmin, ymax);
  glEnd();

CohenSutherlandLineClipAndDraw(x0,y0, x1,y1);
CohenSutherlandLineClipAndDraw(x00, y00, x11, y11);
CohenSutherlandLineClipAndDraw(x02, y02, x12, y12);

  glFlush();
}

void myinit()
{
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glColor3f(1.0, 0.0, 0.0);
  glPointSize(1.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0.0, 499.0, 0.0, 499.0);
}

void main(int argc, char** argv)
{ glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutInitWindowPosition(0,0);
  glutCreateWindow("Cohen Sutherland Line Clipping Algorithm");
  glutDisplayFunc(display);
  myInit();
  glutMainLoop();
}
```

Teacher's Signature:

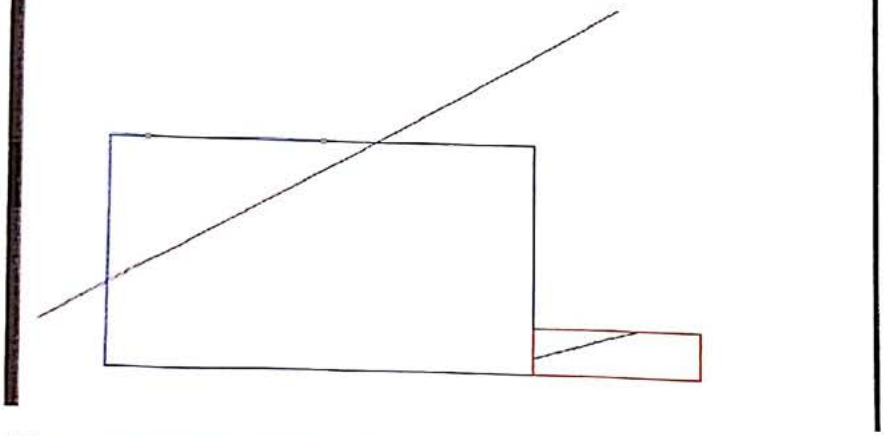
```
Enter window coordinates (xmin ymin xmax ymax):
50 50 300 300
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
300 50 400 100
Enter no. of lines:
1
Enter line endpoints (x1 y1 x2 y2):
150 100 80 200
```

clip



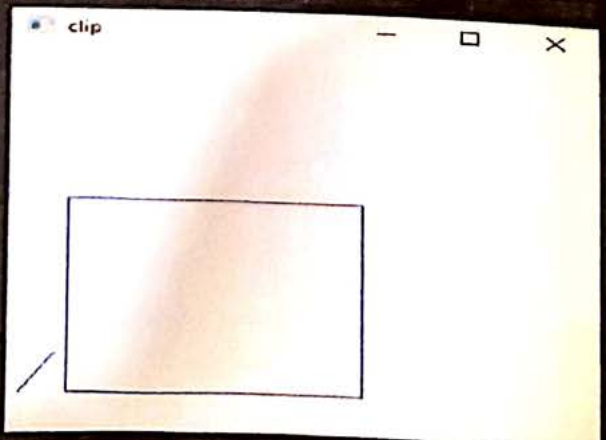
```
Enter window coordinates (xmin ymin xmax ymax):
50 50 300 300
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
300 50 400 100
Enter no. of lines:
1
Enter line endpoints (x1 y1 x2 y2):
10 100 350 150
```

clip



```
Enter window coordinates (xmin ymin xmax ymax):
50 50 300 300
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
300 50 400 100
Enter no. of lines:
1
Enter line endpoints (x1 y1 x2 y2):
10 50 40 100
```

clip



EXPT. NO.

NAME:

Page No.:

Date:

Program 7: Write a program to implement the Liang - Barsky line clipping algorithm. Make provision to specify the window for clipping and viewport for displaying the clipped image.

```
#include <stdio.h>
#include<GL/glut.h>

double xmin=50, ymin=50, xmax=100, ymax=100;
double xvmin=200, yvmin=200, xvmax=300, yvmax=300;

int Clipest(double p, double q, double *t1, double *t2)
{
    double t=q/p;
    if (p<0.0)
    {
        if (t> *t1)  *t1=t;
        if(t> *t2)  return (false);
    }
    else
        if (p>0.0)
    {   if (t<*t2)  *t2=t;
        if (t<*t1)  return(false);
    } else
```

Teacher's Signature:

EXPT. NO. NAME:

Page No.:
Date:

YOUVA

```
if (p == 0.0)
{
  if (q < 0.0)  return (false);
}
return (true);
}

Void LiangBarskyLineClipAndDraw (double x0, double y0,
                                 double x1, double y1)
{
  double dx = x1 - x0, dy = y1 - y0, te = 0.0, t1 = 1.0;
if (cliptest (-dx, x0 - xmin, & te, & t1))
if (cliptest (dx, xmax - x0, & te, & t1))
if (cliptest (-dy, y0 - ymin, & te, & t1))
if (cliptest (dy, ymax - y0, & te, & 1))
{
  if (t1 < 1.0)
  {
    x1 = x0 + t1 * dx;
    y1 = y0 + t1 * dy;
  }
  if (te > 0.0)
  { x0 = x0 + te * dx;
    y0 = y0 + te * dy;
}
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

Date:

YOUVA

```
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, 0.0, 1.0);
glBegin(GL_LINES);
glVertex2f(vx0, vy0);
glVertex2d(vx1, vy1);
glEnd();
}
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

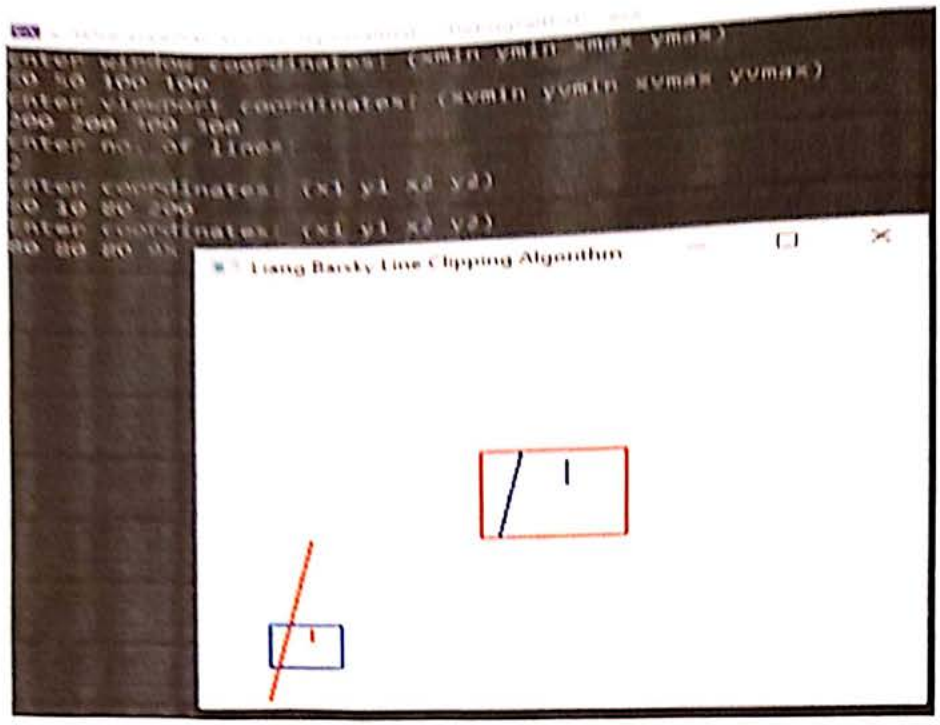
```
void display()
{
    double x0=60, y0=20, x1=80, y1=120;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2d(x0, y0);
    glVertex2d(x1, y1);
    glEnd();

    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xmin, ymin);
    glVertex2f(xmax, ymin);
    glVertex2f(xmax, ymax);
    glVertex2f(xmin, ymax);
    glEnd();
LiangBarskyLineClipAndDraw(x0, y0, x1, y1);
glFlush();
}

void myinit()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 499.0, 0.0, 499.0);
}
```

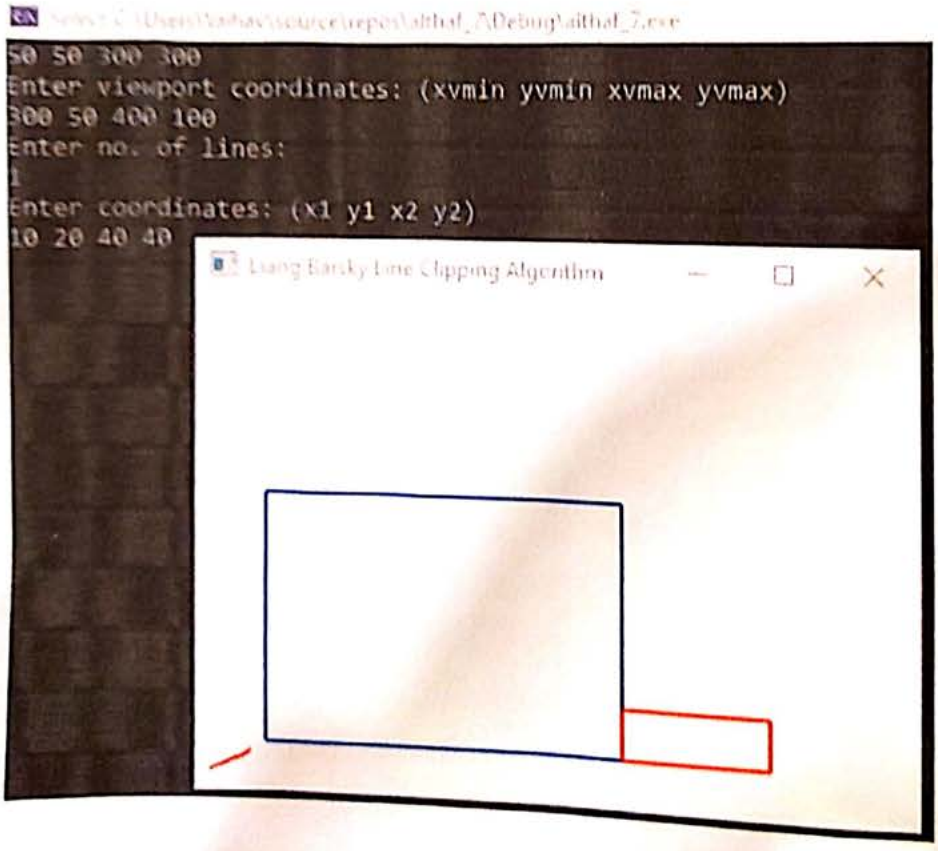
Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
void main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Liang Barsky Line Clipping Algorithm");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

Teacher's Signature:





EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 8: Write a program to implement the Cohen-Hodgeman polygon clipping algorithm. Make provision to specify the input polygon and window for clipping.

```
#include <windows.h>
#include <gl/glut.h>
struct Point {
        float x, y;
} w[4], oVer[4];
int Nont;
void drawPoly(Point pL[], int n) {
        glBegin(GL_POLYGON);
    for (int i=0; i<n; i++)
        glVertex2f(pL[i].x, pL[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;
}
```

Teacher's Signature:

EXPT. NO. NAME:

Page No.:
Date:

YOUVA

```
Void addVer(Point p) {
    oVer[Nout] = p;
    Nout = Nout + 1;
}

PointgetIntersect(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];
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

Date:

```
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, 100.0, 0.0, 100.0);
```

Teacher's Signature:

EXPT. NO.

NAME:

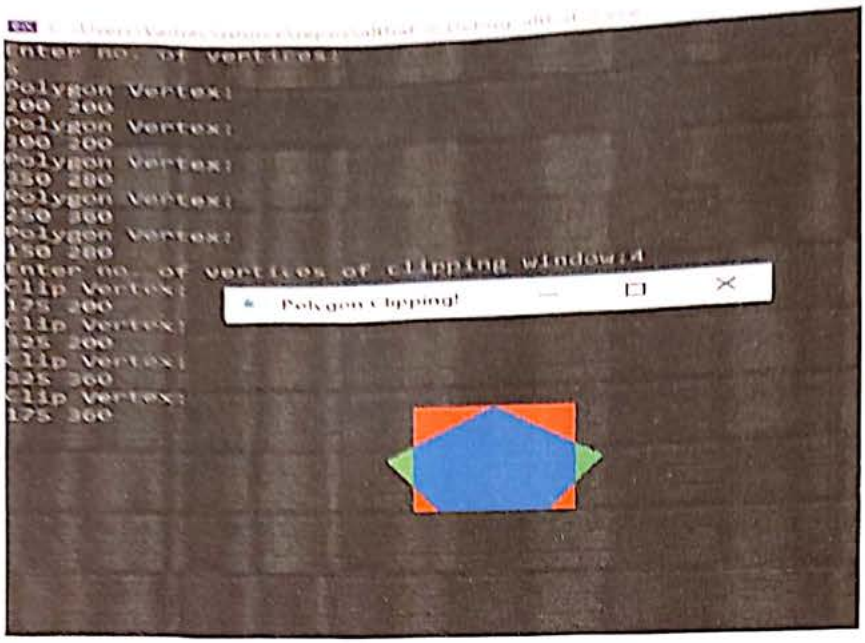
Page No.:

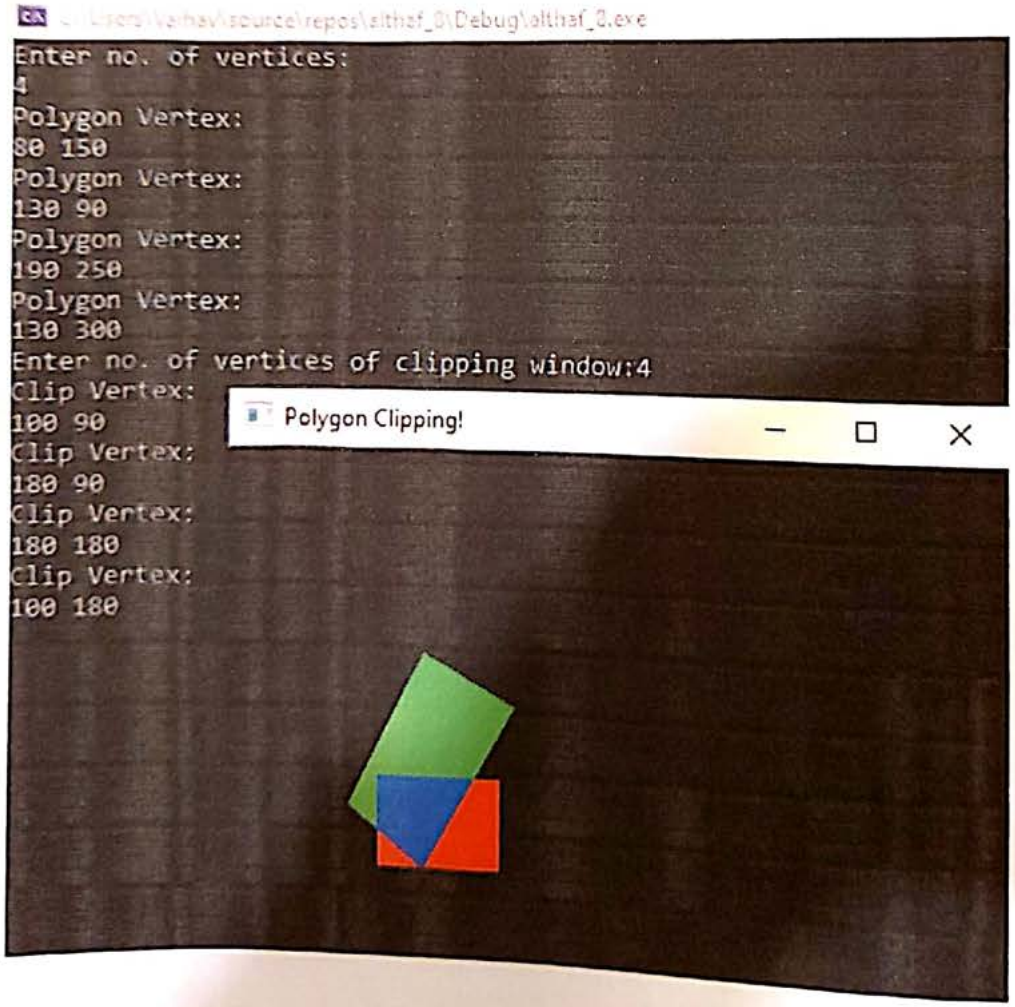
Date:

YOUVA

```
glClear(GL_COLOR_BUFFER_BIT);
w[0].x = 20,  w[0].y = 10;
w[1].x = 20,  w[1].y = 80;
w[2].x = 80,  w[2].y = 80;
w[3].x = 80,  w[3].y = 10;
}
Void display(void){
    Point inVer[4];
    int();
    glColor3f(1.0f, 0.0f, 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 = 60;
    inVer[2].x = 60,  inVer[2].y = 60;
    inVer[3].x = 60,  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;
```

Teacher's Signature:





EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 9: Write a program to model a car like figure using display lists and move a car from one end of the screen to other end. User is able to control the speed with mouse.

```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
#define CAR 1
#define WHEEL 2
float s=1;
void carlist() {
    glNewList(CAR, GL_COMPILE);
    glColor3f(1,1,1);
    glBegin(GL_POLYGON);
    glVertex3f(0,25,0);
    glVertex3f(90,25,0);
    glVertex3f(90,55,0);
    glVertex3f(80,55,0);
    glVertex3f(20,75,0);
    glVertex3f(0,55,0);
    glEnd();
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
    glEndList();
}

void wheelist(){
    glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
    glColor3f(0,1,1);
    glutSolidSphere(10,25,25);
    glEndList();
}

void mykeyboard(unsigned char key, int x, int y){

    switch(key){
    case 't': glutPostRedisplay();
              break;
    case 'q': exit(0);
    default: break;
}
}

Void myInit(){
    glClearColor(0,0,0,0);
    glOrtho(0,600,0,600,0,600);
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
Void draw_wheel(){
    glColor3f(0,1,1);
    glutSolidSphere(10,25,25);
}

Void movecar(float s){
    glTranslatef(s,0.0,0.0);
    glCallList(CAR);
    glPushMatrix();
    glTranslatef(25,25,0.0);
    glCallList(WHEEL);
    glPopMatrix();
    glPushMatrix();
    glTranslatef(75,25,0.0);
    glCallList(WHEEL);
    glPopMatrix();
    glFlush();
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

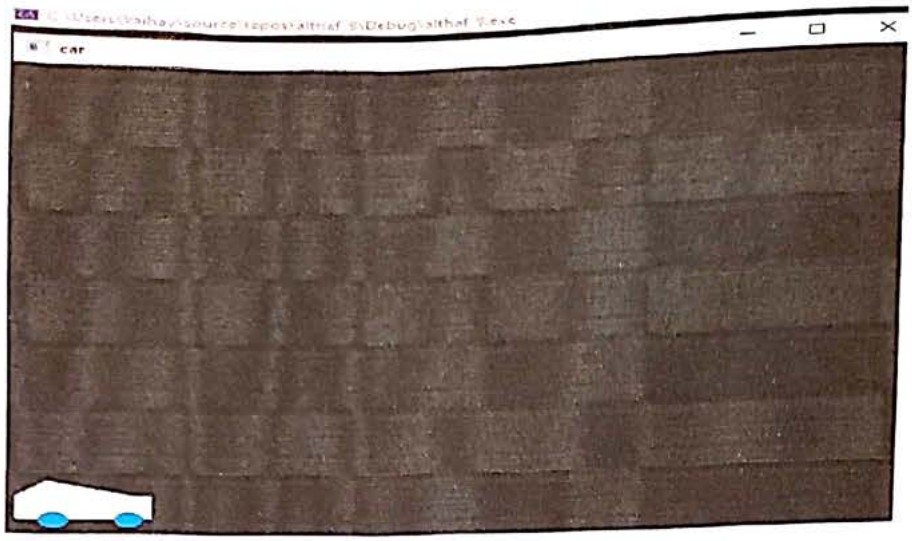
```
void mydisp() {

    glClear(GL_COLOR_BUFFER_BIT);
    carlist();
    moveCar(s);
    wheelist();
}

void mouse(int bth, int state, int x, int y) {

    if(bth == GLUT_LEFT_BUTTON &&
       state == GLUT_DOWN) {

        st=5;
        myDisp();
    }

    else if(bth == GLUT_RIGHT_BUTTON &&
            state == GLUT_DOWN) {

        st=2;
        myDisp();

    }
}
```

Teacher's Signature:

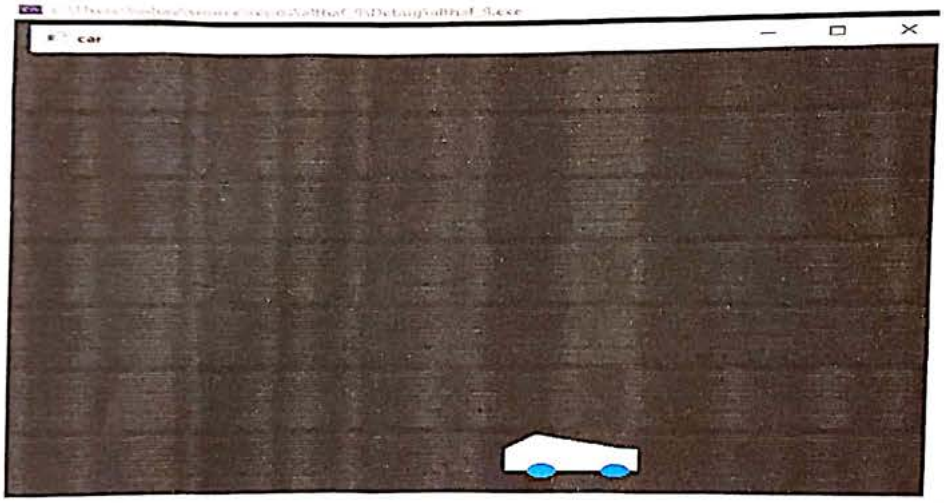
EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
int main(int argc, char * argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(600, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("car");
    myInit();
    glutDisplayFunc(myDisp);
    glutMouseFunc(mouse);
    glutKeyboardFunc(my_keyboard);
    glutMainLoop();
}
```

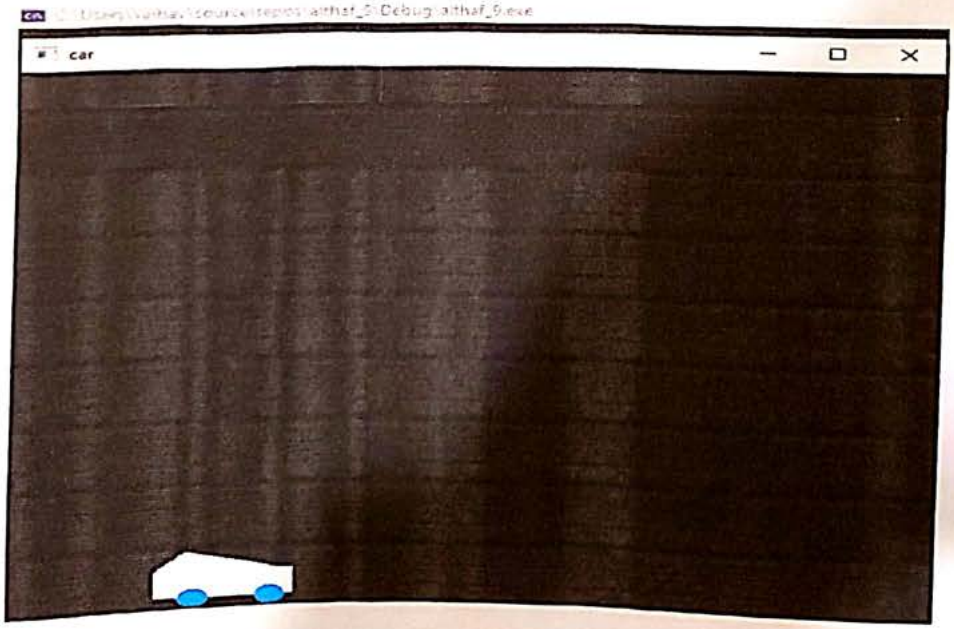
Teacher's Signature:



Click left mouse button to move the car



We can Press t also to move car and q to quit



EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 10: Write a program to create a color cube and spin it using OpenGL transformations.

```
#include <stdlib.h>
#include<GL/glut.h>
#include <gl/GL.h>
#include<gl/GLU.h>
#include <time.h>

GLfloat vertices[] = {-1.0, -1.0, -1.0, 1.0, -1.0, -1.0,
                      1.0, 1.0, -1.0, -1.0, 1.0, -1.0,
                      -1.0, -1.0, 1.0, 1.0, -1.0, 1.0,
                      1.0, 1.0, 1.0, -1.0, 1.0, 1.0};

GLfloat normals[] = {-1.0, -1.0, -1.0, 1.0, -1.0, -1.0,
                     1.0, 1.0, -1.0, -1.0, 1.0, -1.0,
                     -1.0, -1.0, 1.0, 1.0, -1.0, 1.0,
                     1.0, 1.0, 1.0, -1.0, 1.0, 1.0};

GLfloat colors[] = {0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
                    1.0, 1.0, 0.0, 0.0, 1.0, 0.0,
                    0.0, 0.0, 1.0, 1.0, 0.0, 1.0,
                    1.0, 1.0, 1.0, 0.0, 1.0, 1.0};

GLubyte cubeIndices[] = {0, 3, 2, 1, 2, 3, 7, 6, 0, 4, 7, 3, 1, 2,
                         6, 5, 4, 5, 6, 7, 0, 1, 5, 4};
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
static GLfloat theta[] = {0.0, 0.0, 0.0};
static GLfloat beta[] = {0.0, 0.0, 0.0};
static GLint axis = 2;

void delay(float secs)
{
    float end = clock() / CLOCKS_PER_SEC + secs;
    while((clock() / CLOCKS_PER_SEC) < end);
}

void displaySingle(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();
    glRotatef(theta[0], 1.0, 0.0, 0.0);
    glRotatef(theta[1], 0.0, 1.0, 0.0);
    glRotatef(theta[2], 0.0, 0.0, 1.0);

    glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);

    glBegin(GL_LINES);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(1.0, 1.0, 1.0);
glEnd();

glFlush();
}

void spincube()
{
    delay(0.01);
    theta[axis] += 2.0;
    if(theta[axis] > 360.0) theta[axis] -= 360.0;

    glutPostRedisplay();
}

void mouse(int btn, int state, int x, int y)
{
    if(btn == GLUT_LEFT_BUTTON && state == GLUT_
                                DOWN) axis=0;

if(btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
                                axis = 1;

if(btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
                                axis = 2;

}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
void myReshape(int w, int h)
{
  glViewport(0, 0, w, h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if(w<=h)
    glOrtho(-2.0, 2.0, -2.0 * (GLfloat)h / (GLfloat)w,
            2.0 * (GLfloat)h / (GLfloat)w,
            -10.0, 10.0);
  else
    glOrtho(-2.0 * (GLfloat)w / (GLfloat)h,
            2.0 * (GLfloat)w / (GLfloat)h, -2.0, 2.0, -10.0,
            10.0);
  glMatrixMode(GL_MODELVIEW);
}
```

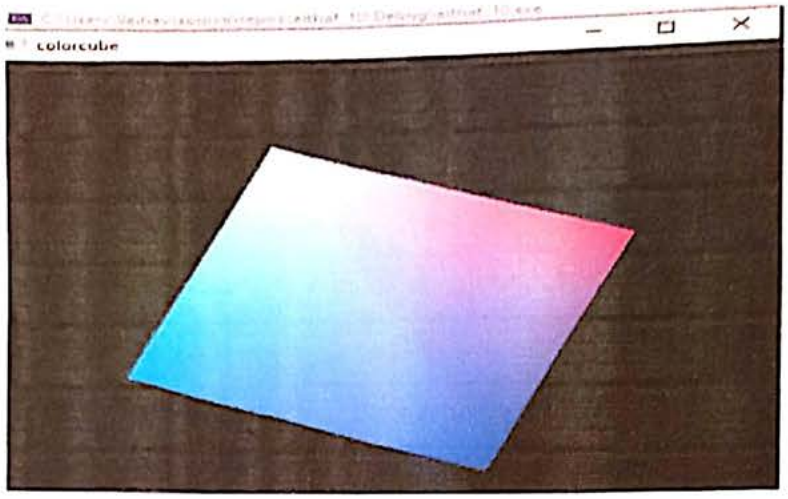
Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

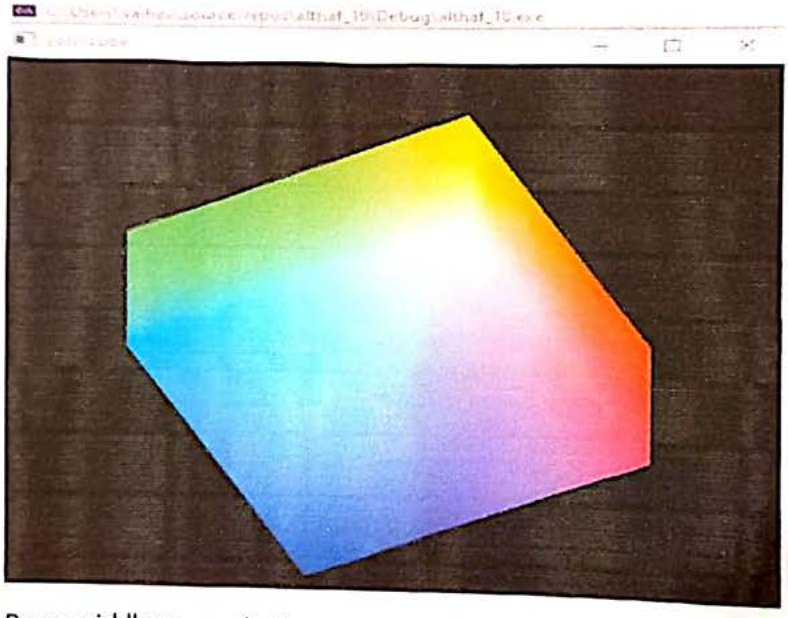
```
void main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(500, 500);
    glutCreateWindow("colorcube");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(displaySingle);
    glutIdleFunc(spinCube);
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glEnableClientState(GL_COLOR_ARRAY);
    glEnableClientState(GL_NORMAL_ARRAY);
    glEnableClientState(GL_VERTEX_ARRAY);
    glVertexPointer(3, GL_FLOAT, 0, vertices);
    glColorPointer(3, GL_FLOAT, 0, colors);
    glNormalPointer(GL_FLOAT, 0, normals);
    glColor3f(1.0, 1.0, 1.0);
    glutMainLoop();
}
```

Teacher's Signature:

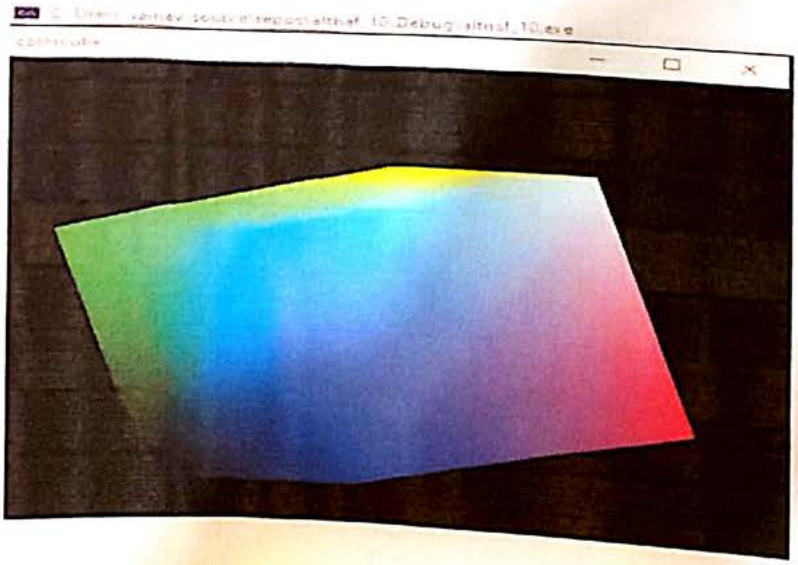
Press left mouse button to rotate about x-axis



Press right mouse button to rotate about y-axis



Press middle mouse button to rotate about z-axis



EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

Program 11: Create a menu with three entries named curves, colors and quit. The entry curves has a sub menu which has four entries namely Limacon, Cardiod, Three-Leaf, and Spiral. The color menu has sub menu with all eight colors of RGB color model. Write a program to create the above hierarchial menu and attach appropriate services to each menu entries with mouse buttons.

```
#include <gl|glut.h>
#include <math.h>
#include <stdio.h>
struct screenPt {
        int x;
        int y;
    };
typedef enum {limacon=1, cardiod=2, threeLeaf=3, spiral=4}
                curveName;

int w=600, h=500;
int curve = 1;
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

Date:

YOUVA

```
int red=0, green=0, blue=0;
void myinit(void){
    glClearColor(1.0,1.0,1.0,1.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
void lineSegment(screenPt p1, screenPt p2){
    glBegin(GL_LINES);
    glVertex2i(p1.x, p1.y);
    glVertex2i(p2.x, p2.y);
    glEnd();
    glFlush();
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
void drawCurve(int curveNum){

    const double twoPi = 6.283185;
    const int a = 175, b = 60;
    float x, theta, dtheta = 1.0/float(x);

    int x0 = 200, y0 = 250;
    screenPt curvePt[2];
    curve = curveNum;

    glColor3f(red, green, blue);

    curvePt[0].x = x0;
    curvePt[0].y = y0;

    glClear(GL_COLOR_BUFFER_BIT);

    switch(curveNum){

    case limacon: curvePt[0].x += a+b; break;
    case cardioid: curvePt[0].x += a+a; break;
    case threeleaf: curvePt[0].x += a; break;
    case spiral: break;
    default: break;

}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
theta = dtheta;
while (theta < twoPi) {
    switch (curveNum) {
        case limacon: r = a * cos(theta) + b; break;
        case cardiod: r = a * (1 + cos(theta)); break;
        case threeleaf: r = a * cos(3 * theta); break;
        case spiral: r = (a / 4.0) * theta; break;
        default: break;
    }
    curvePt[1].x = x0 + r * cos(theta);
    curvePt[1].y = y0 + r * sin(theta);
    lineSegment (curvePt[0], curvePt[1]);
    curvePt[0].x = curvePt[1].x;
    curvePt[0].y = curvePt[1].y;
    theta += dtheta;
}
}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
Void colorMenu(int id){
  switch(id){
  case 0: break;
  case 1: red=0;
          green=0;
          blue=1;
          break;
  case 2: red=0;
          green=1;
          blue=0;
          break;
  case 3: red=0;
          green=1;
          blue=1;
          break;
  case 4: red=1;
          green=0;
          blue=0;
          break;
```

Teacher's Signature:

```
case 5: red=1;
        green=0;
        blue=1;
        break;

case 6: red=1;
        green=1;
        blue=0;
        break;

case 7: red=1;
        green=1;
        blue=1;
        break;

default: break;

}
drawCurve(curve);
}

void myDisplay() {

glClear(GL_COLOR_BUFFER_BIT);

~~print~~ }
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

Date:

YOUVA

```
void myReshape(int nw, int nh){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (double)nw, 0.0, (double)nh);
    glClear(GL_COLOR_BUFFER_BIT);
}

void main(int argc, char ** argv) {
    glutInit(&argc, argv)
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(w,h);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Drawing curves");
    int curveId = glutCreateMenu(colorMenu);
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
int curveId = glutCreateMenu(drawCurve);
glutAddMenuEntry("Limacon", 1);
glutAddMenuEntry("Cardioid", 2);
glutAddMenuEntry("Threeleaf", 3);
glutAddMenuEntry("Spiral", 4);
glutAttachMenu(GLUT_LEFT_BUTTON);
int colorId = glutCreateMenu(colorMenu);
glutAddMenuEntry("Red", 4);
glutAddMenuEntry("Green", 2);
glutAddMenuEntry("Blue", 1);
glutAddMenuEntry("Black", 0);
glutAddMenuEntry("Yellow", 6);
glutAddMenuEntry("Cyan", 3);
glutAddMenuEntry("Magenta", 5);
```

Teacher's Signature:

EXPT. NO.

NAME:

Page No.:

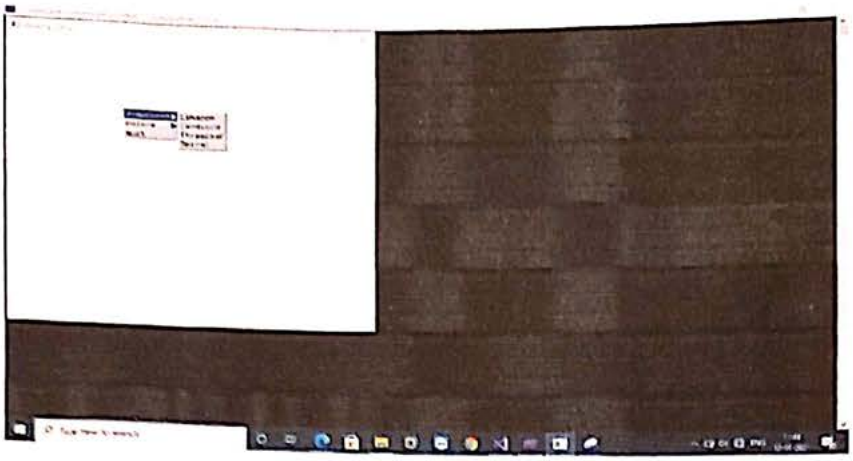
Date:

YOUVA

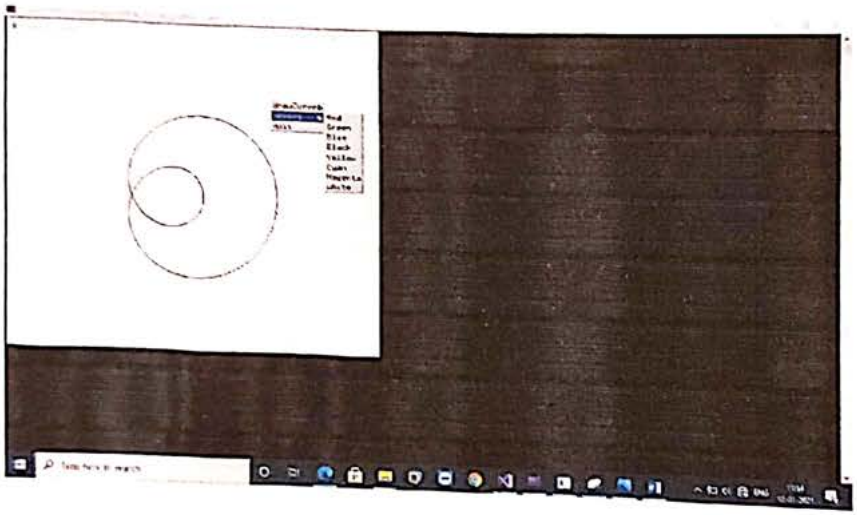
```
glutAddMenuEntry("white", 7);
glutAttachMenu(GLUT_LEFT_BUTTON);
glutCreateMenu(main_menu);
glutAddSubMenu("drawCurve", curveId);
glutAddSubMenu("colors", colorId);
glutAddMenuEntry("quit", 3);
glutAttachMenu(GLUT_LEFT_BUTTON);
myinit();
glutDisplayFunc(mydisplay);
glutReshapeFunc(myreshape);
glutMainLoop();
}
```

Teacher's Signature:

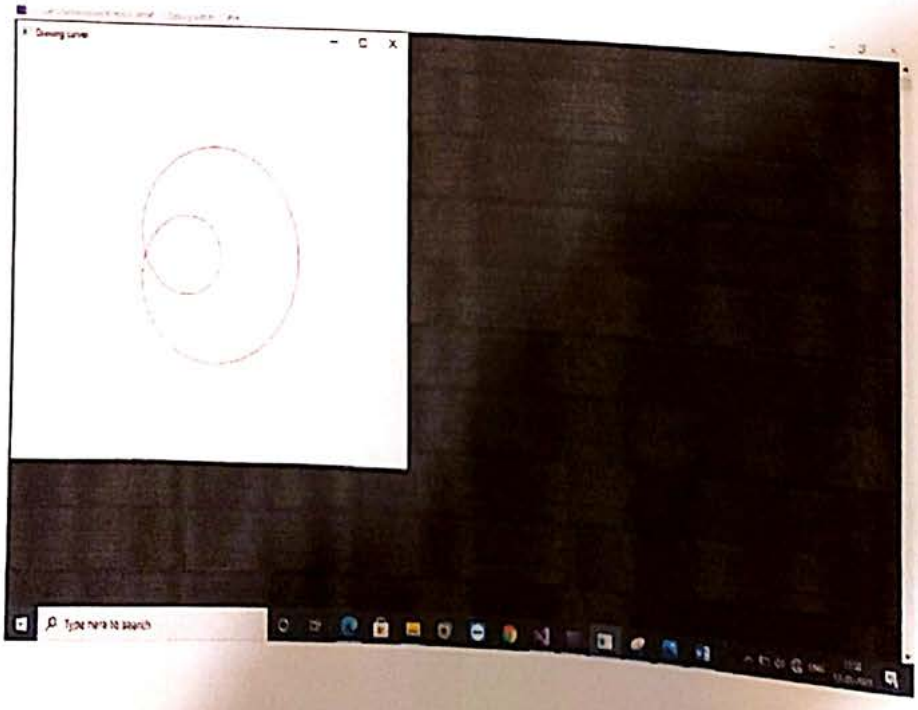
Click left mouse button to show and select the menu



We can choose colour or quit



Choose colour



EXPT NO: NAME:

Page No.: YOUVA
Date:

Program 12: Write a program to construct Bezier curve. Control points are supplied through keyboard/mouse

```
#include <iostream>
#include <math.h>
#include<gl/glut.h>

using namespace std;
float f, g, x, xc[4], yc[4];
int flag=0;

void myInit() {

    glClearColor(1,1,1,1);
    glColor3f(1,1,1);
    glPointSize(5);

    gluOrtho2D(0,500,0,500);

}
```

Teacher's Signature:

EXPT. NO. | NAME: | Page No.: | Date: | YOUVA

```
Void drawPixel(float x, float y) {
    glBegin(GL_POINTS);
    glVertex2f(x,y);
    glEnd();
}

Void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    int i;
    double t;
    glColor3f(0,0,0);

    glBegin(GL_POINTS);

for(t=0; t<1; t=t+0.005) {

    double xt = pow(1-t,3) * x1[10] + 3*t * pow(1-t,2)
                x[1]
                x1[1] + 3* pow(t,2) * (1-t) * x1[2] +
                pow(t,3) * x1[3];

double yt = pow(1-t,3) * yc[0] + 3*t * pow(1-t,2) *
            yc[1] + 3 * pow(t,2) * (1-t) * yc[2] + pow(t,3) *
                                                   yc[3];
    glVertex2f(xt, yt);
}
```

Teacher's Signature:

EXPT. NO | NAME: | Page No: | Date: | Yours

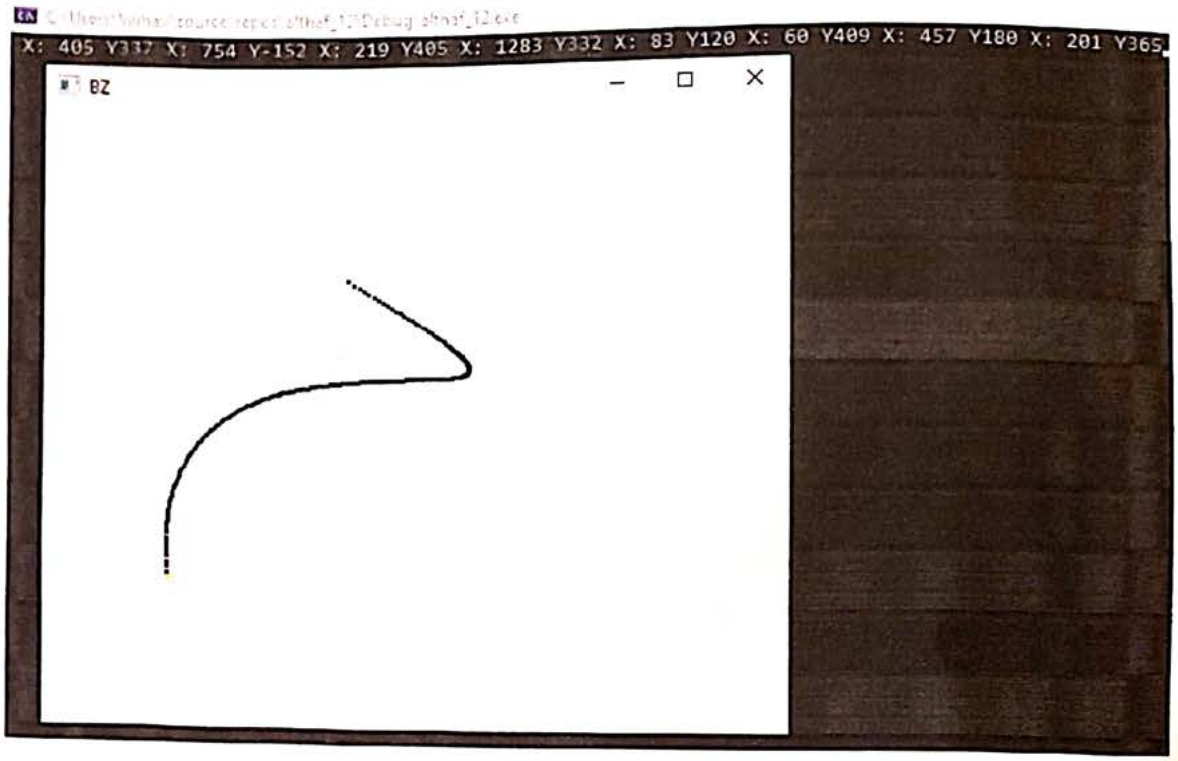
```
glColor3f(1,1,0);
for(i=0; i<4; i++){
    glVertex2f(x1[i], yc[i]);
    glEnd()
    glFlush();
}
}

Void mymouse(int btn, int stat, int x, int y)
{
    if(btn== GLUT_LEFT_BUTTON &&
        state == GLUT_DOWN && flag<4)
{
    x1[flag] = x;
    yc[flag] = 500-y;

    cout<<" X :"<<x<<"Y"<<500-y;

    glPointSize(3);
    glColor3f(1,1,0);
    glBegin(GL_POINTS);
    glVertex2f
    glVertex2i(x,500-y);
    glEnd();
    glFlush();
    flag++;
}
```

Teacher's Signature:

```
if (flag >= 4 && btn == GLUT_LEFT_BUTTON)
{
    glClearColor(0,0,1);
    display();
    flag=0;
}
}

int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("BZ");
    glutDisplayFunc(display);
    glutMouseFunc(mymouse);
    myInit();
    glutMainLoop();
}
```

Teacher's Signature:

Draw BZ curve through mouse



Enter co-ordinates via keyboard