EXPT. NO | NAME | PAGE NO | DATE | SIGN

Program 1: Write a program to generate a line using Bresenham's line drawing technique. Consider slopes greater than one and less than one. User must be able to draw as many lines and specify input through keyboard/mouse.

```
#include<iostream>
#include<GL/glut.h>
using namespace std;

int x1, y1, x2, y2, flag=0;

void draw_pixel(int x, int y){
    glColor3f(1,0,0);
    glBegin(GL_POINTS);
        glVertex2i(x,y);
    glEnd();
    glFlush();
}

void draw_line(){
    int dx, dy, i, e;
    int incx, incy, inc1, inc2;
    int x, y;
    dx = x2 - x1;
    dy = y2 - y1;
    if(dx<0) dx = -dx;
    if(dy<0) dy = -dy;
```

Teacher's Signature:

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

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

Teacher's Signature:

EXPT NO | NAME | Page No | Date

```
        if (e>0) {
            x+= inc(x);
            e+= inc(1);
        } else
            e+= inc(2);
        y+= inc(y);
        draw_pixel(x,y);
    }
}
  glFlush();
}

void myinit()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1,1,1,1);
    gluOrtho2D(-250,250,-250,250);
}

void MyMouse(int button, int state, int x, int y) {
    switch(button)
    {
        case GLUT_LEFT_BUTTON:
            if(state == GLUT_DOWN) {
                if(flag==0) {
                    printf("Defining x1,y1");
                    x1 = x-250;
                    y1 = 250-y;
```

Teacher's Signature:

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

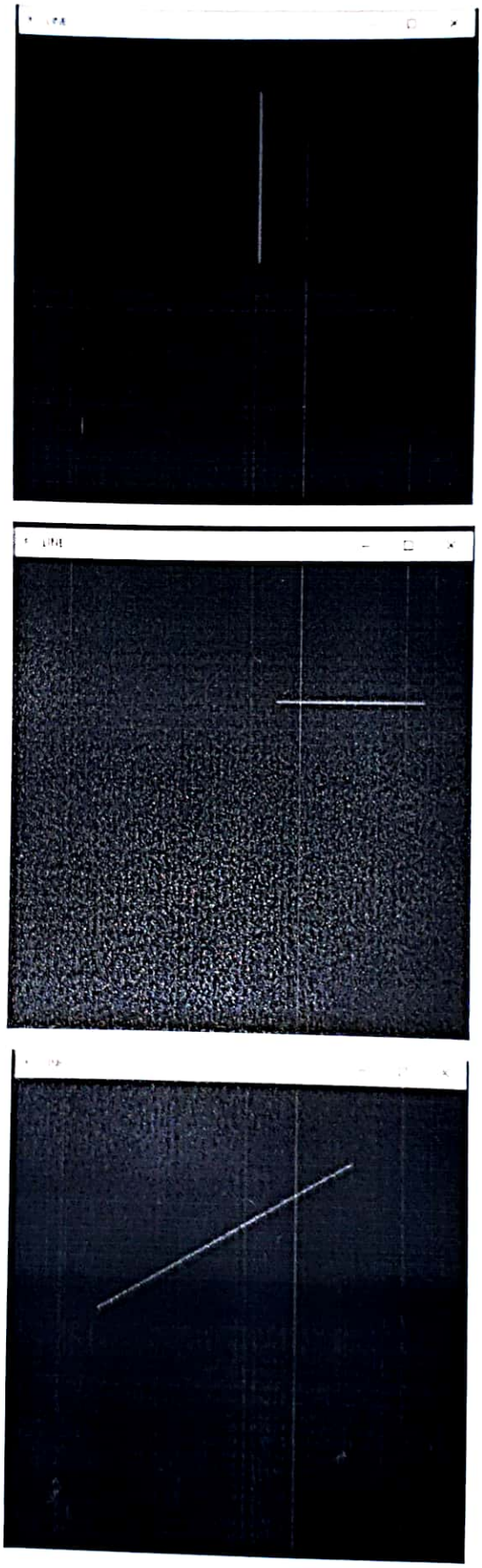
```
        flag++;
        cout << x1 << " " << y1 << "\n";
    }
    else {
        printf("Defining x2, y2 ");
        x2 = x-250;
        y2 = 250-y;
        flag=0;
        cout << x2 << " " << y2 << "\n";
        draw_line();
    } break;
  }
}
void display() { }

int main (int argc, char **argv) {
    //FOR KEYBOARD
    cout << "x1 \n" << "y1 \n";
    cin >> x1 >> y1;
    cout << "x2 \n y2 \n";
    cin >> x2 >> y2;

    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutCreateWindow ("LINE");
    myinit();
    // glutMouseFunc(MyMouse);
```

Teacher's Signature

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

```
    draw_line();  // For keyboard, remove for mouse
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Teacher's Signature



Program1: Line

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

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

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

int xc, yc, r;
int rx, ry, xce, yce;

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

void circlebres() {
    glClear(GL_COLOR_BUFFER_BIT);
```

Teacher's Signature:

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

```
int x=0, y=r;
int d=3-2*r;
while (x<=y)
{
    draw_circle (xc, yc, x, y);
    x++;
    if (d<0)
        d = d+4*x+6;
    else {
        y--;
        d = d+4*(x-y)+10;
    }
    draw_circle (xc, yc, x, y);
}
    glFlush();
}
int p1_x, p2_x, p1_y, p2_y;
int point1_done = 0;

void myMouseFunction(int button, int state, int x, int y){
  if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN
    && point1_done == 0) {
        p1_x = x-250;
        p1_y = 250-y;
        point1_done = 1;
  }
  else if (button == GLUT_LEFT_BUTTON && state == GLUT-
        DOWN) {
```

Teacher's Signature:

```
        p2_x = x - 250;
        p2_y = 250 - y;
        xc = p1_x;
        yc = p1_y;
        float exp = (p2_x - p1_x) * (p2_x - p1_x) + (p2_y - p1_y) *
                    (p2_y - p1_y);
        r = (int)(sqrt(exp));
        drawbres();
        point1_done = 0;
    }
}

void draw_ellipse (int xce, int yce, int x, int y) {
    glBegin (GL_POINTS);
    glVertex2i (x + xce, y + yce);
    glVertex2i (-x + xce, y + yce);
    glVertex2i (x + xce, -y + yce);
    glVertex2i (-x + xce, -y + yce);
    glEnd();
}

void midptellipse () {
    glClear (GL_COLOR_BUFFER_BIT);
    float dx, dy, d1, d2, x, y;
    x = 0;
    y = ry;
    d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
    dx = 2 * ry * ry * x;
```

Teacher's Signature:

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

```
dy = 2*rx*rx*y;
while (dx < dy) {
    draw_ellipse (xce, yce, x, y);
    if (d1 < 0) {
        x++;
        dx = dx + (2*ry*ry);
        d1 = d1 + dx + (ry*ry);
    } else {
        x++;
        y--;
        dx = dx + (2*ry*ry);
        dy = dy - (2*rx*rx);
        d1 = d1 + dx - dy + (ry*ry);
    }
}
d2 = ((ry*ry)*((x+0.5)*(x+0.5))) + ((rx*rx)*
     ((y-1)*(y-1))) - (rx*rx*ry*ry);
while (y >= 0)
{
    draw_ellipse (xce, yce, x, y);
    if (d2 > 0) {
        y--;
        dy = dy - (2*rx*rx);
        d2 = d2 + (rx*rx) - dy;
    } else {
        y--;
        x++;
        dx = dx + (2*ry*ry);
```

Teacher's Signature:

```
            dy = dy - (2*rx*rx);
            d2 = d2 + dx - dy + (rx*rx);
        }
    }
    glFlush();
}

int p1e_x, p2e_x, p1e_y, p2e_y, p3e_x, p3e_y;
int pointle_done = 0;

void MyMousefunction (int button, int state, int x, int y){
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN
        && pointle_done == 0){
            p1e_x = x-250;
            p1e_y = 250-y;
            xce = p1e_x;
            yce = p1e_y;
            pointle_done = 1;
    } else if (button == GLUT_LEFT_BUTTON && state == GLUT_
        DOWN && pointle_done==1) {
            p2e_x = x-250;
            p2e_y = 250-y;
            float exp = (p2e_x - p1e_x)*(p2e_x - p1e_x)
            + (p2e_y - p1e_y)*(p2e_y - p1e_y);
            rx = (int)(sqrt(exp));
            pointle_done = 2;
    }
else if (button == GLUT_LEFT_BUTTON && pointle_done
    == 2) {
```

Teacher's Signature:

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

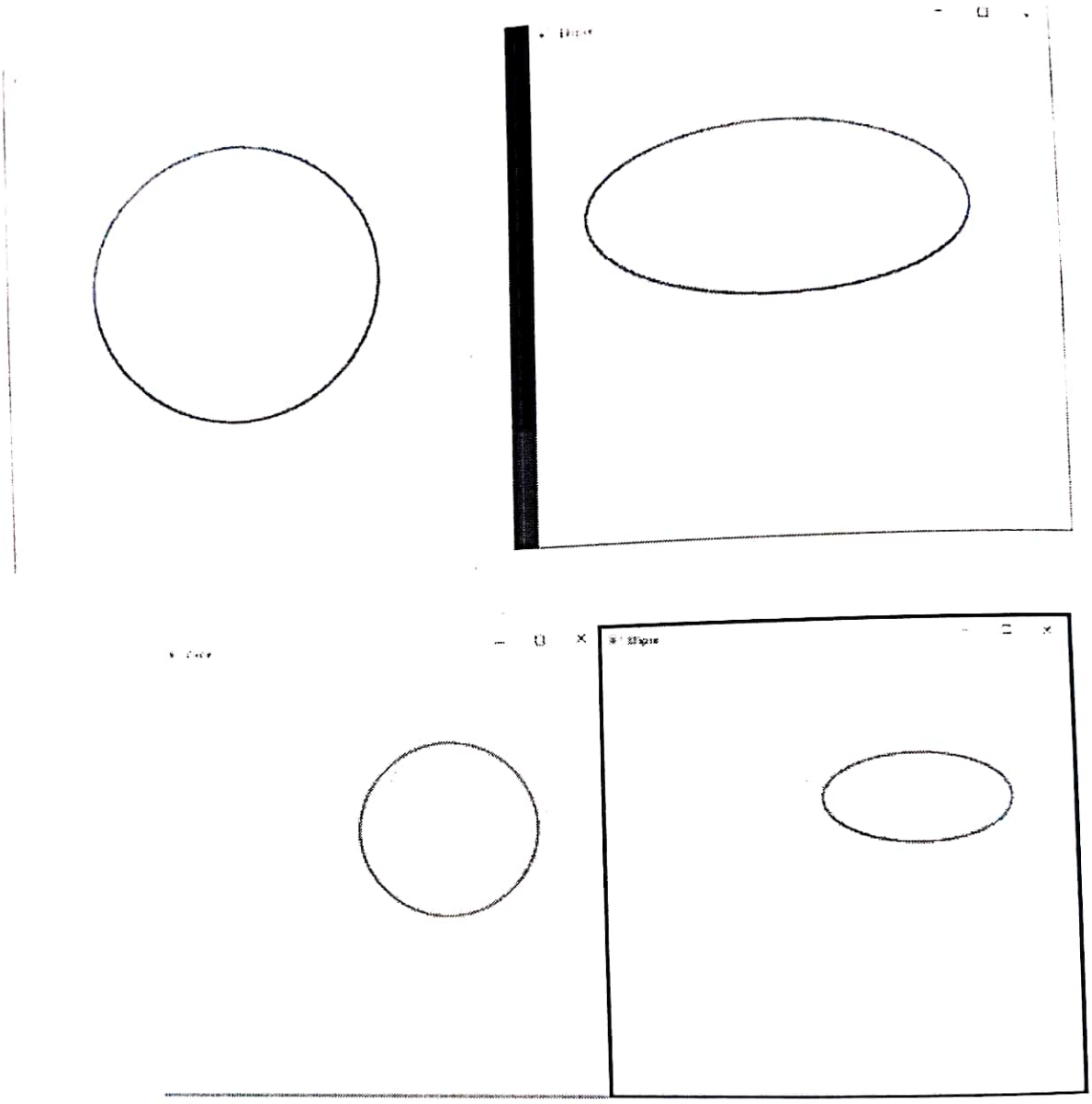
```
        p3e_x = x-250;
        p3e_y = 250-y;
        float exp = (p3e_x - p1e_x) * (p3e_x - p1e_x) +
        (p3e_y - p1e_y) * (p3e_y - p1e_y);
        ry = (int).(sqrt(exp));
        midptellipse ();
        pointle_done = 0;
    }
}

void myinit () {
    glClear (1, 1, 1, 1);
        glColor3f (1.0, 0.0, 0.0);
        glPointSize (3.0);
    gluOrtho2D (-250, 250, -250, 250);
}

int main (int argc, char **argv)
{
    glut Init (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (0, 0);
    // FOR MOUSE
     int id1 = glutCreateWindow ("Circle");
     glut set Window (id1);
     glutMouseFunc (myMouseFunction(circle));
     glutDisplayFunc (myDrawing());
```

Teacher's Signature:

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

```
minit();
glutInitWindowSize(500,500);
glutInitWindowPosition(600,100);
int id2 = glutCreateWindow("Ellipse");
glutSetWindow(id2);
glutMouseFunc(myMouseFunc);
glutDisplayFunc(myDrawing);

printf("Enter 1 to draw circle, 2 to draw ellipse \n");
int ch;
scanf("%d", &ch);
switch(ch){
    case 1: printf("Enter coordinates of centre of
                    circle & radius \n");
            scanf("%d %d %d", &xc, &xy, &r);
            glutCreateWindow("Circle");
            glutDisplayFunc(circleres);
            break;
    case 2: printf("Enter coordinates of centre of
            ellipse & major & minor radius \n");
            scanf("%d %d %d %d", &xce, &yce, &rx, &ry);
            glutCreateWindow("Ellipse");
            glutDisplayFunc(midptellipse);
            break;
}
minit();
glutMainLoop();
}
```

Teacher's Signature:



Program2: Circle and Ellipse

EXPT NO | NAME

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>
#include <stdio.h>

int n;
typedef float point[3];

point tetra[4] = {{0,100,-100},{0,0,100},{100,-100,-100},
                  {-100,-100,-100}};
void tetrahedron(void);
void myinit(void);
void divide_triangle(point a, point b, point c, int m);
void draw_triangle(point p1, point p2, point p3);

int main(int argc, char **argv){
    printf("Enter the number of iterations: ");
    scanf("%d", &n);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_
                        DEPTH);
    glutInitWindowSize(500,500);
    glutCreateWindow("Seirpinski Gasket");
    glutDisplayFunc(tetrahedron);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
}
```

Teacher's Signature:

EXPT NO | NAME | Page No | Date | YOUVA

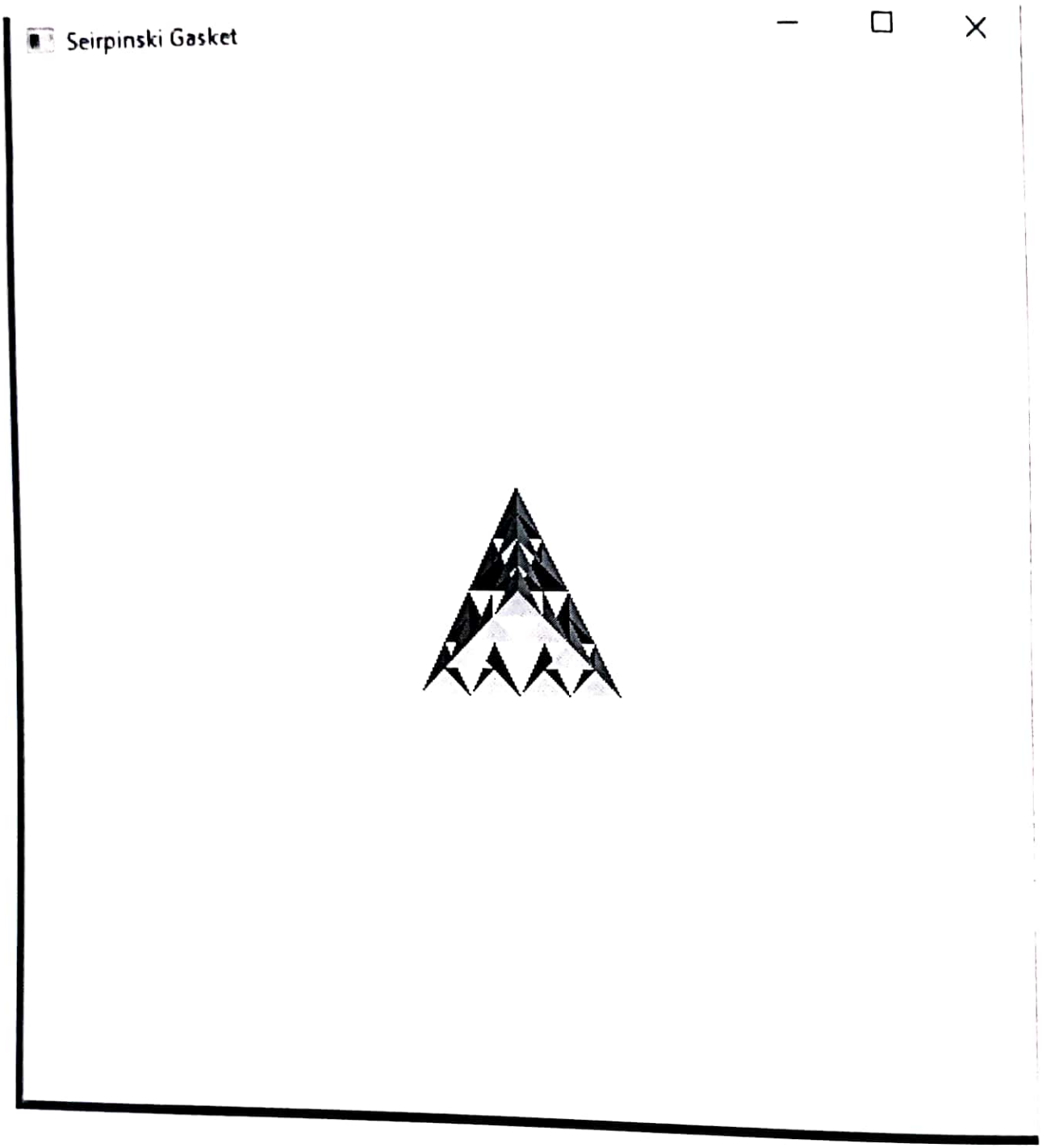
```
void divide_triangle(point a, point b, point c, int m){
    point v1, v2, v3;
    int j;
    if(m>0){
        for(j=0; j<3; j++){
            v1[j]=(a[j]+b[j])/2;
            v2[j]=(a[j]+c[j])/2;
            v3[j]=(b[j]+c[j])/2;
        }
        divide_triangle(a, v1, v2, m-1);
        divide_triangle(b, v3, v1, m-1);
        divide_triangle(c, v3, v1, m-1);
    }else
        draw_triangle(a,b,c);
}
void myinit(){
    glClearColor(1,1,1,1);
    glOrtho(-500, 500, -500, 500, -500, 500);
}
void tetrahedron(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(1.0, 1.0, 0.0);
    divide_triangle(tetra[0], tetra[1], tetera[2], m);
    glColor3f(0, 1, 0);
    divide_triange(tetra[3], tetra[2], tetra[1], m);
    glColor3f(0, 0, 1);
    divide_triangle(tetra[0], tetra[3], tetra[1], m);
    glColor3f(0, 0, 0);
    divide_triangle(tetra[0], tetra[2], tetra[3], m);
    glFlush();
```

Teacher's Signature:

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

```
void draw_triangle (point p1, point p2, point p3) {
    glBegin(GL_TRIANGLES);
    glVertex3fv(p1);
    glVertex3fv(p2);
    glVertex3fv(p3);
    glEnd();
}
```

Teacher's Signature:



Program 3: Seirpinski Gasket

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

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

```
#include <stdlib.h>
#include <GL/glut.h>
#include <algorithm>
#include<iostream>
#include<unistd.h>
using namespace std;
float x[100], y[100];
int n,m;
int wx=500, wy=500;
static float intx[10]={0};

void draw_line (float x1, float y1, float x2, float y2){
    sleep(1);
    glColor3f(1,0,0);
    glBegin(GL_LINES);
    glVertex2f(x1,y1);
    glVertex2f(x2, y2);
    glEnd();
    glFlush();
}

void edgedetect (float x1, float y1, float x2, float y2, int
                 &scanline){
float temp;
```

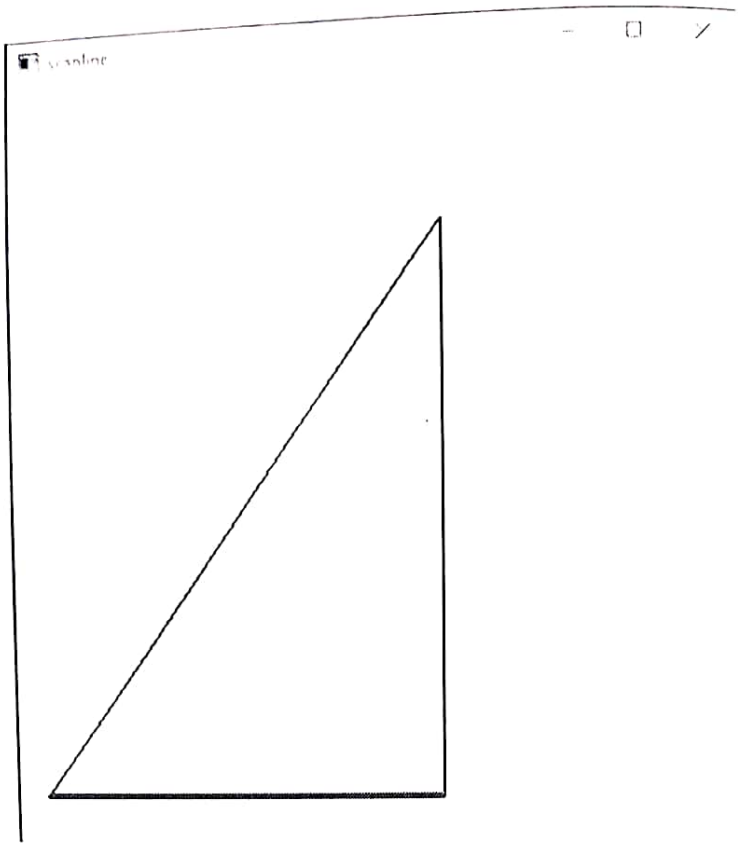
Teacher's Signature:

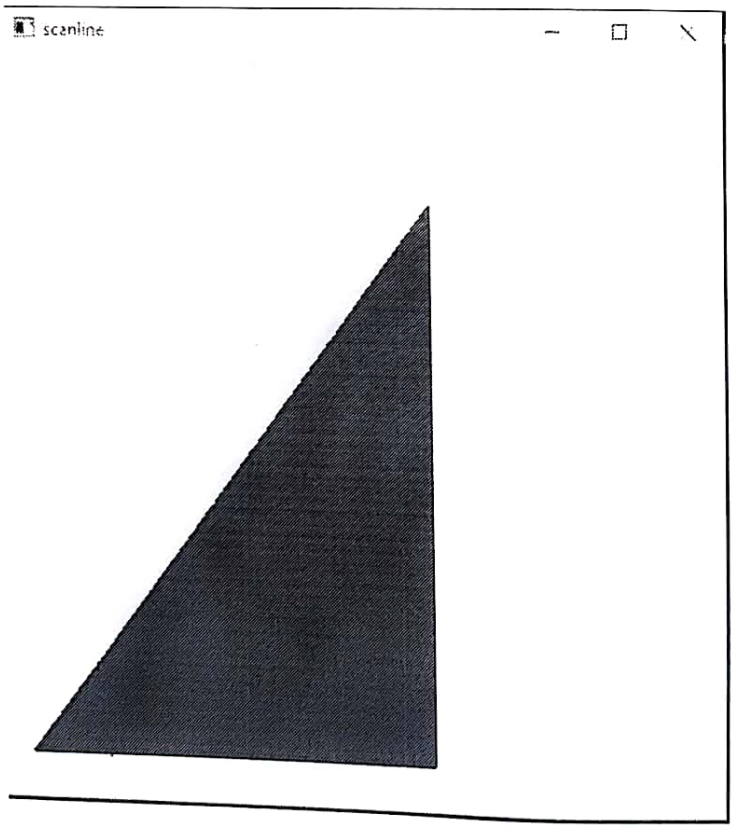
```
if(y2<y1){
    temp = x1; x1 = x2; x2 = temp;
    temp = y1; y1 = y2; y2 = temp;
}
if (scanline > y1 && scanline < y2)
    intx[m++] = x1 + (scanline - y1) * (x2 - x1) / (y2 - y1);
}
void Scanfill (float x[], float y[]) {
    for (int s1 = 0; s1 <= wy; s1++) {
        m = 0;
        for (int i = 0; i < n; i++) {
            edgeDetect (x[i], y[i], x[(i+1)%n],
                        y[(i+1)%n], s1); }
        sort (intx, (intx+m));
        if (m == 2)
            for (int i = 0; i < m; i = i+2)
                draw_line (intx[i], s1, intx[i+1], s1);
        }
}
void display_filled_polygon () {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin (GL_LINE_LOOP);
    for (int i = 0; i < n; i++)
        glVertex2f (x[i], y[i]);
    glEnd();
    Scanfill (x, y);
    glFlush ();
}
```

Teacher's Signature

```
void myInit();
    glClearColor(1,1,1,1);
    glColor3f(0,0,1);
    glPointSize(5);
    gluOrtho2D(0, wx, 0, wy);
}
int main(int argc, char **argv){
    glutInit(&argc, argv);
    printf("Enter the no.of sides:\n");
    scanf("%d", &n);
    printf("Enter coordinates of endpoints:\n");
    for(int i=0; i<n; i++){
        printf("x, y:\n");
        scanf("%d%f", &x[i], &y[i]);
    }
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutCreateWindow("scanline");
    glutDisplayFunc(display_filled_polygon);
    myInit();
    glutMainLoop();
}
```

Teacher's Signature:





Program4: Scan fill

EXPT NO | NAME | PAGE NO | DATE | MARK

Program 5: Write a program to create a house & perform
i) Rotate it about a given fixed point using OpenGL transformations
ii) Reflect it about y=mx+c using OpenGL transformations

```
#include<GL/glut.h>
#include<math.h>
#include<iostream>
#include<stdio.h>
using namespace std;
float house[11][2] = {{100,200},{200,250},{300,200},{100,200},
{100,100},{175,100},{175,150},{225,150},{225,100},{300,100},{300,100}};
int angle;
float m, c, theta;
void display()
{
    glClearColor(1,1,1,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    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();
```

Teacher's Signature

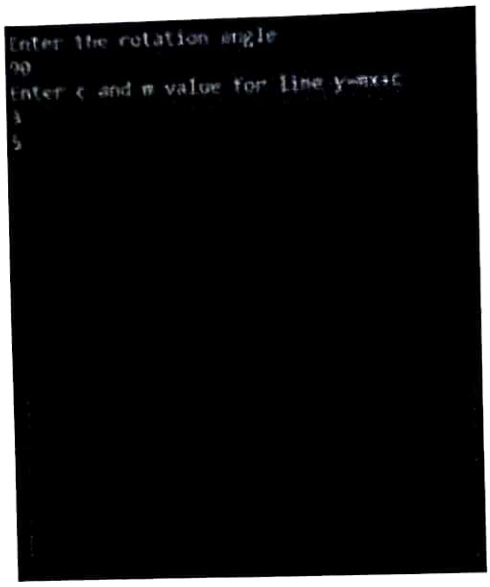
EXPT NO | NAME: | Page No.: | Date | YOUVA

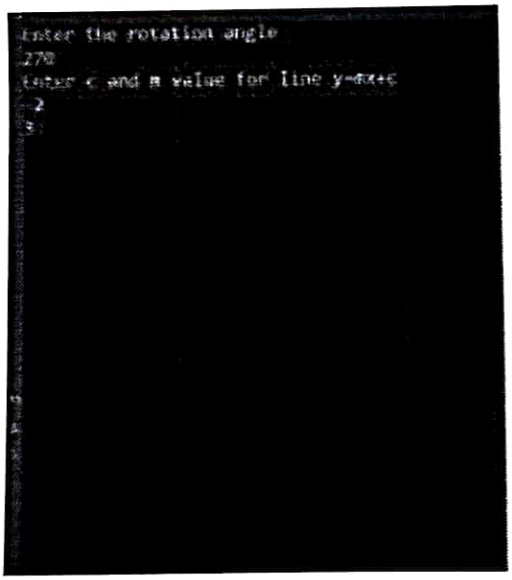
```
        glFlush();
        glPushMatrix();
        glTranslatef(100,100,0);
        glRotatef(angle,0,0,1);
        glTranslatef(-100,-100,0);
        glColor3f(1,1,0);
        glBegin(GL_LINE_LOOP);
        for(int i=0;i<11;i++){
                glVertex2fv(house[i]);
        glEnd();
        glPopMatrix();
        glFlush();
}
void display2(){
    glClearColor(1,1,1,0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_Projection);
    glLoadIdentity();
    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;y2=m*x2+c;
    glColor3f(1,1,0).
    glBegin(GL_LINES);
        glVertex2f(x1,y1);
        glVertex2f(x2,y2);
glEnd();
```

Teacher's Signature:

```
glFlush();
glPushMatrix();
glTranslatef(0,c,0);
theta = atan(m)
theta = theta * 180 / 3.141;
glRotatef(theta,0,0,1);
glScalef(1,-1,1);
glRotatef(-theta,0,0,1);
glTranslatef(0,-c,0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<4; i++)
                glVertex2fv(mouse[i]);

    glEnd();
    glPopMatrix(); glFlush(); }
int main(int argc, char **argv){
printf("Enter the angle");
cin >> angle;
printf("enter c+m value for line y=mx+c\n");
c+m >> c >> m;
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(900,900);
glutCreateWindow("Rotation");
glutDisplayFunc(display);
glutMouseFunc(mouse);
myInit();
glutMainLoop();
}
```

Teacher's Signature:





Program5: House Rotation

EXPT. NO | NAME | Page No. | Date

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

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#define outcode int
#define true 1
#define false 0

double xmin=50, ymin=50, xmax=100, ymax=100;
double xvmin=200, yvmin=200, xvmax=300, yvmax=300;
const int RIGHT=4, LEFT=8, TOP=1, BOTTOM=2;

Outcode computecode(double x, double y){
    Outcode code=0;
    if(y>ymax)
        code |= TOP;
    else if(y<ymin)
        code |= BOTTOM;
    else if(x>xmax)
        code |= RIGHT;
    else if(x<xmin)
        code |= LEFT;
    return code;
}
```

Teacher's Signature:

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

```
void cohensuther(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;
            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;
```

Teacher's Signature:

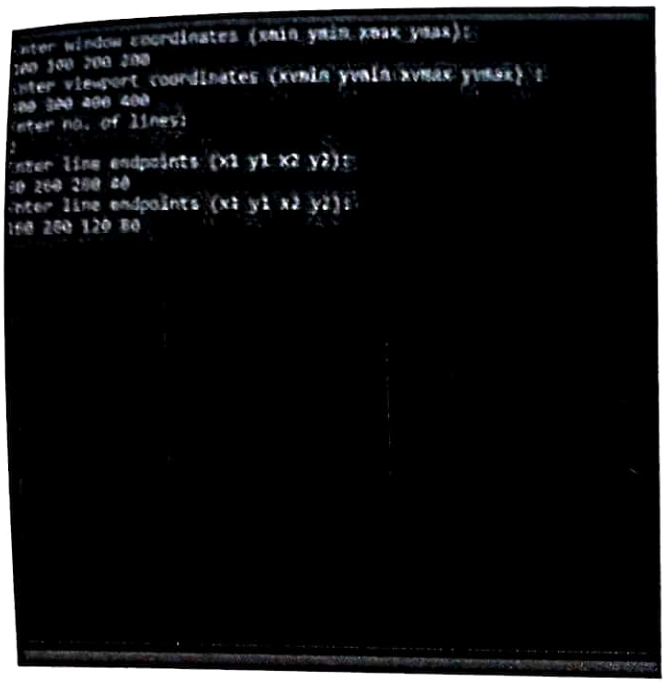
```
        outcode0 = computecode(x0,y0);
    } else {
        x1 = xi;
        y1 = yi;
        outcode1 = computecode(x1,y1);
    }
} while(!done);
if(accept) {
    double sx = (xvmax-xvmin)/(xmax-xmin);
    double sy = (ymax-yvmin)/(ymax-ymin);
    double vx0 = xvmin+(x0-xmin)*sx;
    double vy0 = yvmin+(y0-ymin)*sy;
    double vy1 = yvmin+(y1-ymin)*sy;
    double vx1 = xvmin+(x1-xmin)*sx;
    glColor3f(0,0,1);
    glBegin(GL_LINES);
    glVertex2d(vx0,vy0);
    glVertex2d(vx1,vy1);
    glEnd();
}}
void display(){
    double x0=20, y0=20, x1=70, y1=70;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    glBegin(GL_LINES);
        glVertex2d(x0,y0);
        glVertex2d(x1,y1);
    glEnd();
    cohensuther(x0,y0,x1,y1);
}
```

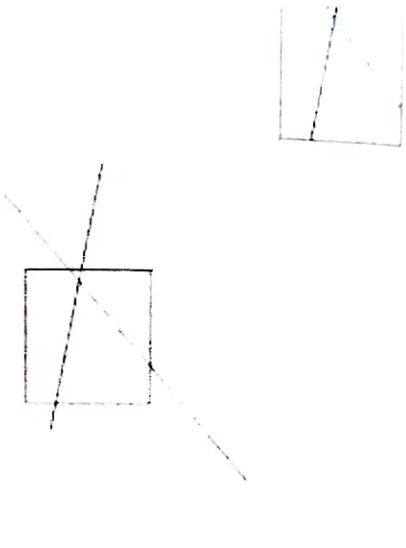
Teacher's Signature:

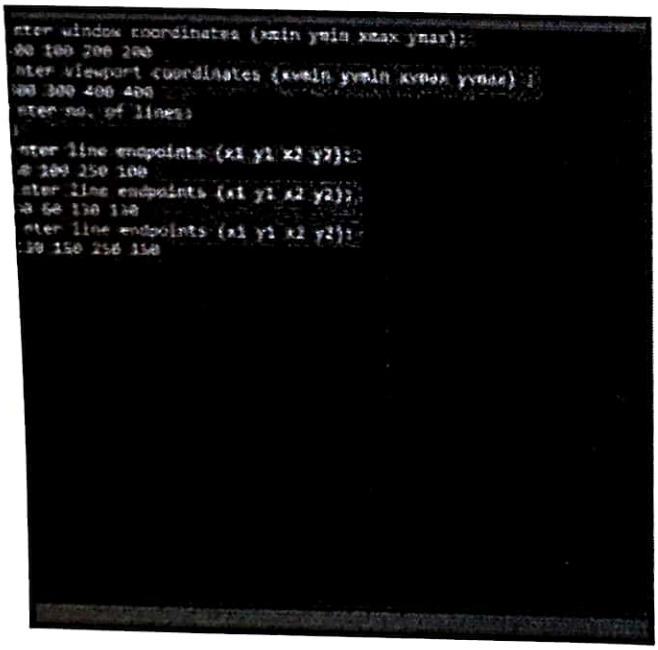
EXPT. NO. | NAME: | Page No. | Date | YOUVA

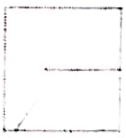
```
        p3e_x = x-250;
        p3e_y = 250-y;
        float exp = (p3e_x - p1e_x) * (p3e_x - p1e_x) +
        (p3e_y - p1e_y) * (p3e_y - p1e_y);
        ry = (int).(sqrt(exp));
        midptellipse();
        paintle_done = 0;
    }
}

void myinit() {
    glClear(1,1,1,1);
        glColor3f(1.0, 0.0, 0.0);
        glPointSize(3.0);
    gluOrtho2D(-250, 250, -250, 250);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0,0);
    // FOR MOUSE
     int id1 = glutCreateWindow("Circle");
     glutSetWindow(id1);
     glutMouseFunc(myMouseFunction(circle));
     glutDisplayFunc(myDrawing());
```

Teacher's Signature:









Program6: CohenSutherLand

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 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 cliptest (double p, double q, double *u1, double *u2){
    double r;
    if(p) r=q/p;
    if(p<0){
        if(r>*u1) *u1=r;
        if(r>*u2) return(false);
    } else if(p>0){
        if(r<*u2) *u2=r;
        if(r<*u1) return (false);
    } else if(p==0){
        if(q<0) return(false);
    }
    return (true);
}
void LiangBarskyLineClipAndDraw(double x0, double y0, double x1, double
    double dx=x1-x0, dy=y1-y0; u1=0.0, u2=0.0;
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xvmin, yvmin);
    glVertex2f(xvmax, yvmin);
    glVertex2f(xvmax, yvmax);
    glVertex2f(xvmin, yvmax);
    glEnd();
```

Teacher's Signature:

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

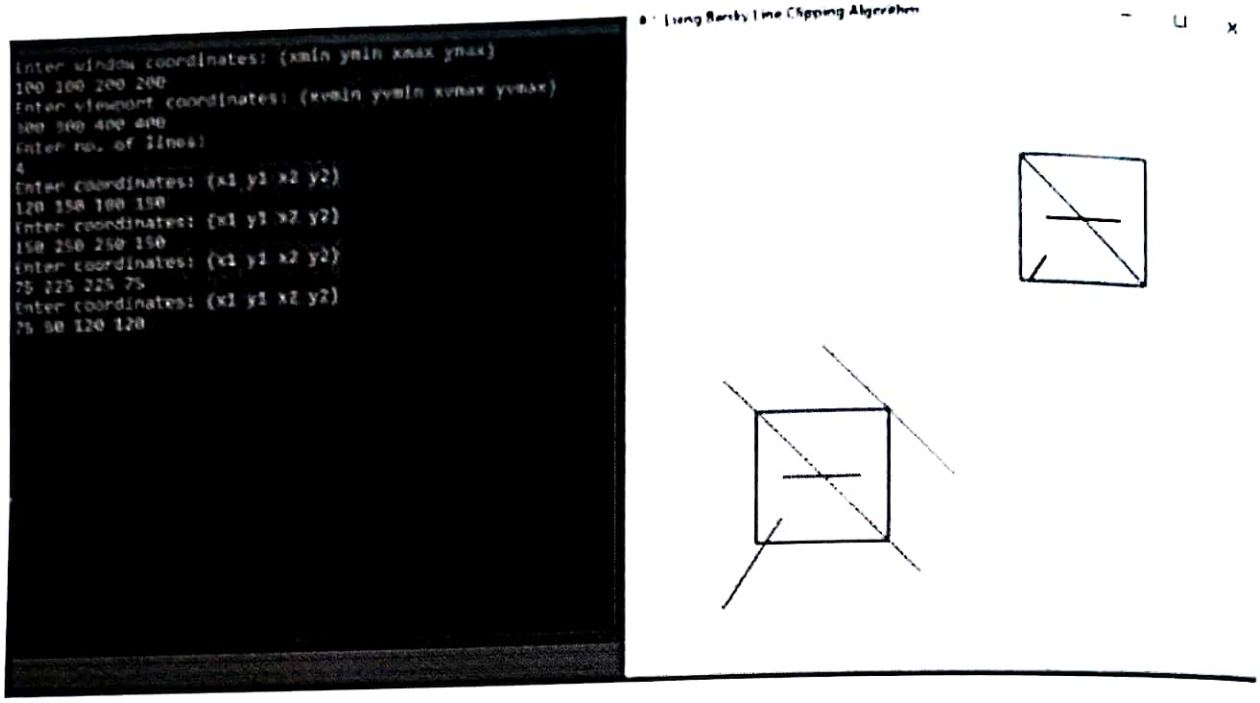
```
if(cliptest(-dx, x0-xmin, &u1, &u2))
if(cliptest(dx, xmax-x0, &u1, &u2))
if(cliptest(-dy, y0-ymin, &u1, &u2))
if(cliptest(dy, ymax-y0, &u1, &u2))
{
    if(u2 < 1){
        x1 = x0 + u2*dx;
        y1 = y0 + u2*dy;
    } if(u1 > 0.0){
        x0 = x0 + u1*dx;
        y0 = y0 + u1*dy;
    }
    glColor3f(0,0,1);
    glBegin(GL_LINES);
    glVertex2d(rx0, ry0); glVertex2d(rx1, ry1);
    glEnd();
}}
void display(){
    double x0=60, y0=60, x1=90, y1=90;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    glBegin(GL_LINES)
        glVertex2d(x0, y0);
        glVertex2d(x1, y1);
    glEnd();
    LiangBarskyLineClipandDraw(x0, y0, x1, y1);
    glFlush();
}
```

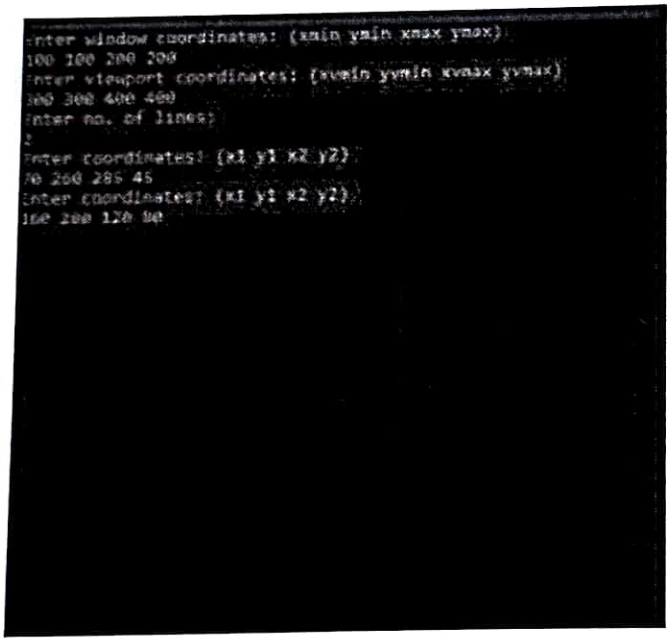
Teacher's Signature:

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

```
        p3e_x = x-250;
        p3e_y = 250-y;
        float exp = (p3e_x - p1e_x) * (p3e_x - p1e_x) +
        (p3e_y - p1e_y) * (p3e_y - p1e_y);
        ry = (int).(sqrt(exp));
        midptellipse();
        paintle_done = 0;
    }
}

void myinit() {
    glClear(1,1,1,1);
        glColor3f(1.0, 0.0, 0.0);
        glPointSize(3.0);
    gluOrtho2D(-250, 250, -250, 250);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0,0);
    // FOR MOUSE
     int id1 = glutCreateWindow("Circle");
     glutSetWindow(id1);
     glutMouseFunc(myMouseFunction(circle));
     glutDisplayFunc(myDrawing());
```

Teacher's Signature:





Program7: LiangBarsky Line Clipping

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

Program 8: Write a program to implement Cohen-Hodgman polygon clipping algorithm. Make window for clipping.

```
#include <iostream>
#include <GL/glut.h>
using namespace std;
int polysize, poly_points[20][2], org_poly_size, org_poly_points[20][2], clipper_size, clipper_points[20][2];
const int MAX_POINTS = 20;
void drawPoly(int p[][2], int n){
    glBegin(GL_POLYGON);
        for(int i=0;i<n;i++)
            glVertex2f(p[i][0], p[i][1]);
    glEnd();
}
int x_intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
    int num = (x1*y2 - y1*x2)*(x3-x4) - (x1-x2)*(x3*y4-y3*x4);
    int den = (x1-x2)*(y3-y4) - (y1-y2)*(x3-x4);
    return num/den;
}
int y_intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
    int num = (x1*y2 - y1*x2)*(y3-y4) - (y1-y2)*(x3*y4-y3*x4);
    int den = (x1-x2)*(y3-y4) - (y1-y2)*(x3-x4);
    return num/den;
}
```

Teacher's Signature:

EXPT NO. NAME Page No. Date KOUVA

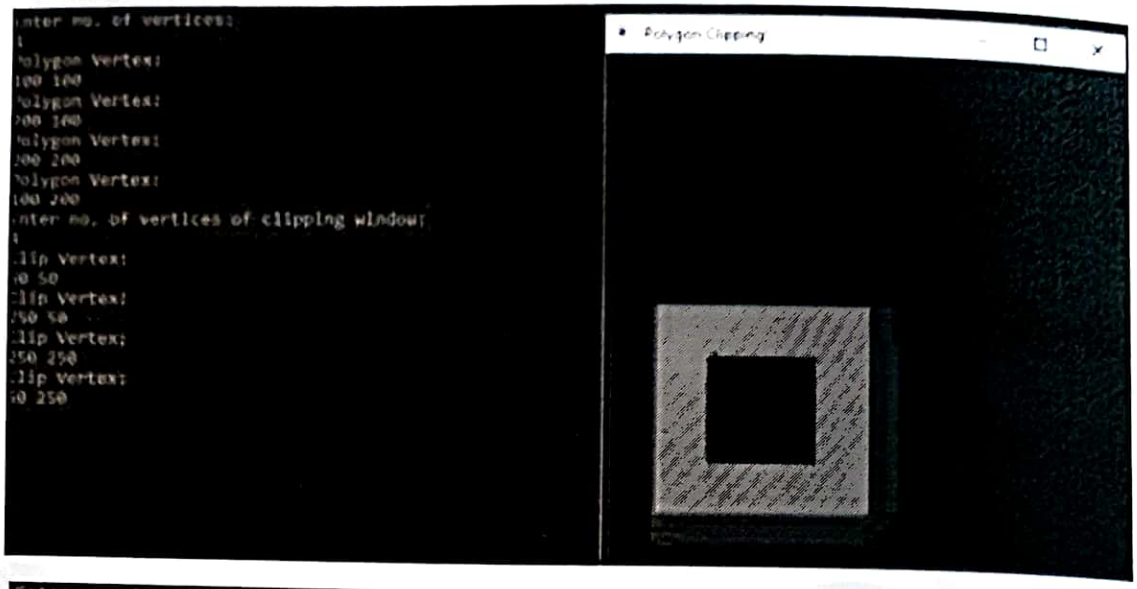
```
void clip(int poly_points[][2], int &poly_size, int x1, int
y1, int x2, int y2) {
    int new_points[MAX_POINTS][2], new_poly_size=0;
    for(int i=0; i<poly_size; i++) {
        int k=(i+1)%poly_size;
        int ix=poly_points[i][0], iy=poly_points[i][1];
        int kx=poly_points[k][0], ky=poly_points[k][1];
        int i_pos=(x2-x1)*(iy-y1)-(y2-y1)*(ix-x1);
        int k_pos=(x2-x1)*(ky-y1)-(y2-y1)*(kx-x1);
        if(i_pos>=0 && k_pos>=0){
            new_points[new_poly_size][0]=kx;
            new_points[new_poly_size++][1]=ky;
        }else if(i_pos<0 && k_pos>=0) {
            new_points[new_poly_size][0]=x_intersect(
            x1,y1,x2,y2,ix,iy,kx,ky);
            new_points[new_poly_size++][1]=y_intersect(
            x1,y1,x2,y2,ix,iy,kx,ky);
            new_points[new_poly_size][0]=kx;
            new_points[new_poly_size++][1]=ky;
        }else if(i_pos>=0 && k_pos<0) {
            new_points[new_poly_size][0]=x_intersect(
            x1,y1,x2,y2,ix,iy,kx,ky);
            new_points[new_poly_size++][1]=y_intersect(
            x1,y1,x2,y2,ix,iy,kx,ky);
    }   }
poly_size=new_poly_size;
for(int i=0; i<poly_size; i++){
    poly_points[i][0]=new_points[i][0];
    poly_points[i][1]=new_points[i][1];
}
}
```

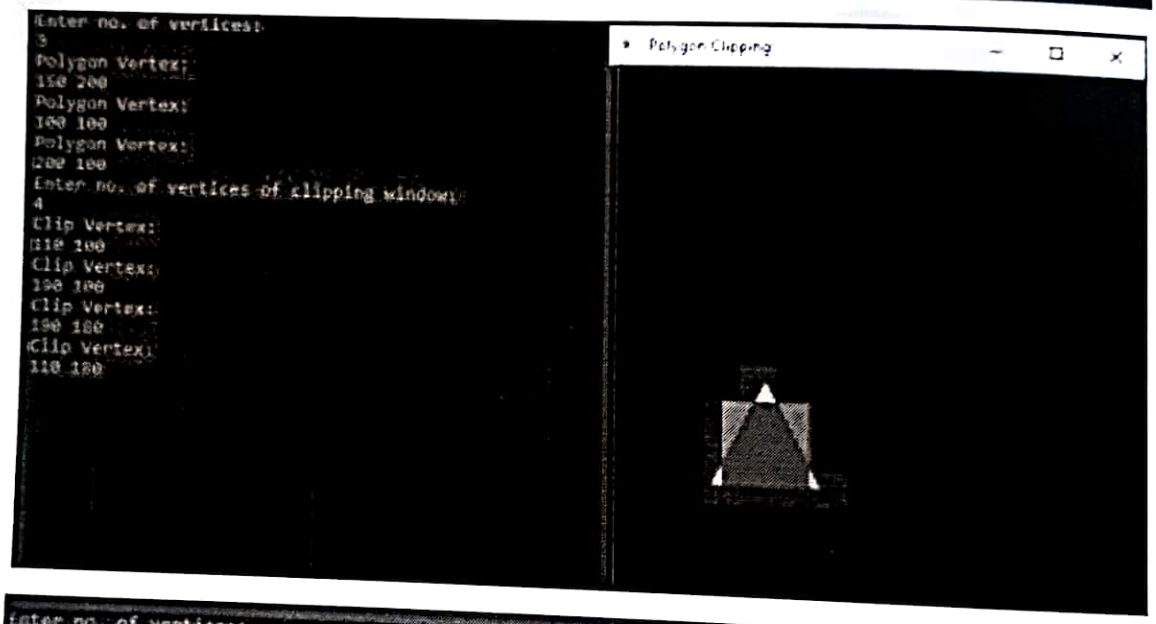
Teacher's Signature:

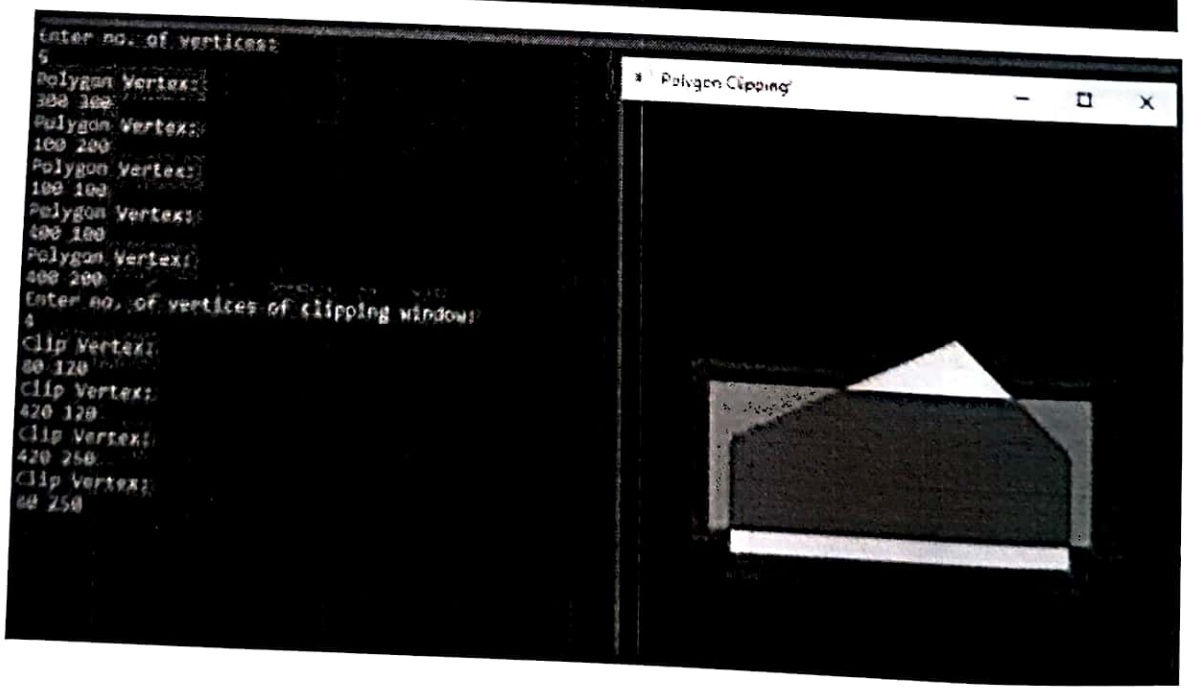
EXPT NO | NAME | Page No. | Date | YOUVA

```
void display(){
    init();
    glColor3f(1,0,0);
    drawPoly(clipper_points, clipper_size);
    glColor3f(0,1,0);
    drawPoly(org_poly_points, org_poly_size);
    for(int i=0; i<clipper_size; i++){
        int k=(i+1)%clipper_size;
        clip(poly_size, poly_points, clipper_points[i][0],
        clipper_points[i][1], clipper_points[k][0], clipper_
        points[k][1]); }
    glColor3f(0,0,1);
    drawPoly(poly_points, poly_size);
    glFlush();
}
int main(int argc, char** argv){
    printf("enter vertices");
    scanf("%d", &poly_size);
    for(int i=0; i<poly_size; i++){
        printf("Polygon vertices");
        scanf("%d%d", &poly_size[i][0], &poly_points[i][0]);
        org_poly_points[i][0] = poly_points[i][0];
        org_poly_points[i][1] = poly_points[i][1];
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400,400);
    glutCreateWindow("Polygon Clipping");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Teacher's Signature:







Program8: Polygon Clipping

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

Program 9: Write a program to model a car and wheel using display lists and move a car from one end of the screen to other.

```
#include <GL/glut.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);
    glEnd();
    glEndList();
}
void wheellist(){
    glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
    glColor3f(0,1,1);
    glutSolidSphere(10,25,25);
    glEndList();
}
void draw_wheel(){
    glColor3f(0,1,1);
    glutSolidSphere(10,25,25);
}
```

Teacher's Signature:

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

```
        p3e_x = x-250;
        p3e_y = 250-y;
        float exp = (p3e_x - p1e_x) * (p3e_x - p1e_x) +
        (p3e_y - p1e_y) * (p3e_y - p1e_y);
        ry = (int).(sqrt(exp));
        midptellipse();
        paintle_done = 0;
    }
}

void myinit() {
    glClear(1,1,1,1);
        glColor3f(1.0, 0.0, 0.0);
        glPointSize(3.0);
    gluOrtho2D(-250, 250, -250, 250);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0,0);
    // FOR MOUSE
     int id1 = glutCreateWindow("Circle");
     glutSetWindow(id1);
     glutMouseFunc(myMouseFunction(circle));
     glutDisplayFunc(myDrawing());
```

Teacher's Signature:

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

```
void movecar(float s){
    glTranslatef(s,0.0,0.0);
    glCallList(CAR);
    glPushMatrix();
    glTranslatef(-75,-25,0);
    glCallList(WHEEL);
    glPopMatrix();
    glTranslatef(75,25,0);
    glCallList(WHEEL);
    glPopMatrix();
    glFlush();
}
void myDisp(){
    glClear(GL_COLOR_BUFFER_BIT);
    Carlist();
    movecar(s);
    wheellist();
}
int main(int argc, char* argv[]){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(600,500);
    glutCreateWindow("car");
    myInit();
    glutDisplayFunc(myDisp);
    glutMouseFunc(mouse);
    glutKeyboardFunc(myKeyboard);
    glutMainLoop();
}
```

Teacher's Signature:

Program9: Car

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

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

```
#include <stdlib.h>
#include <GL/glut.h>
#include <time.h>
GLfloat vertices[] = {-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-10,
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};
GLfloat colors[] = {0.0,0.0,0.0, 1.0,0.0,0.0, 1.0,1.0,0.0,0.0,1.0,
0.0,0.0,1.0,1.0,1.0,0.0,1.0,0.0,1.0,1.0,0.0,1.0,1.0};
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};
static GLfloat theta[] = {0,0,0};
static GLfloat beta[] = {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],0,1,0);
    glRotatef(theta[2],0,0,1);
    glDrawElements(GL_QUADS,24,GL_UNSIGNED_BYTE,cubeIndi
    glBegin(GL_LINES);
```

Teacher's Signature:

EXPT NO | NAME | Page No | Date | YOUVA

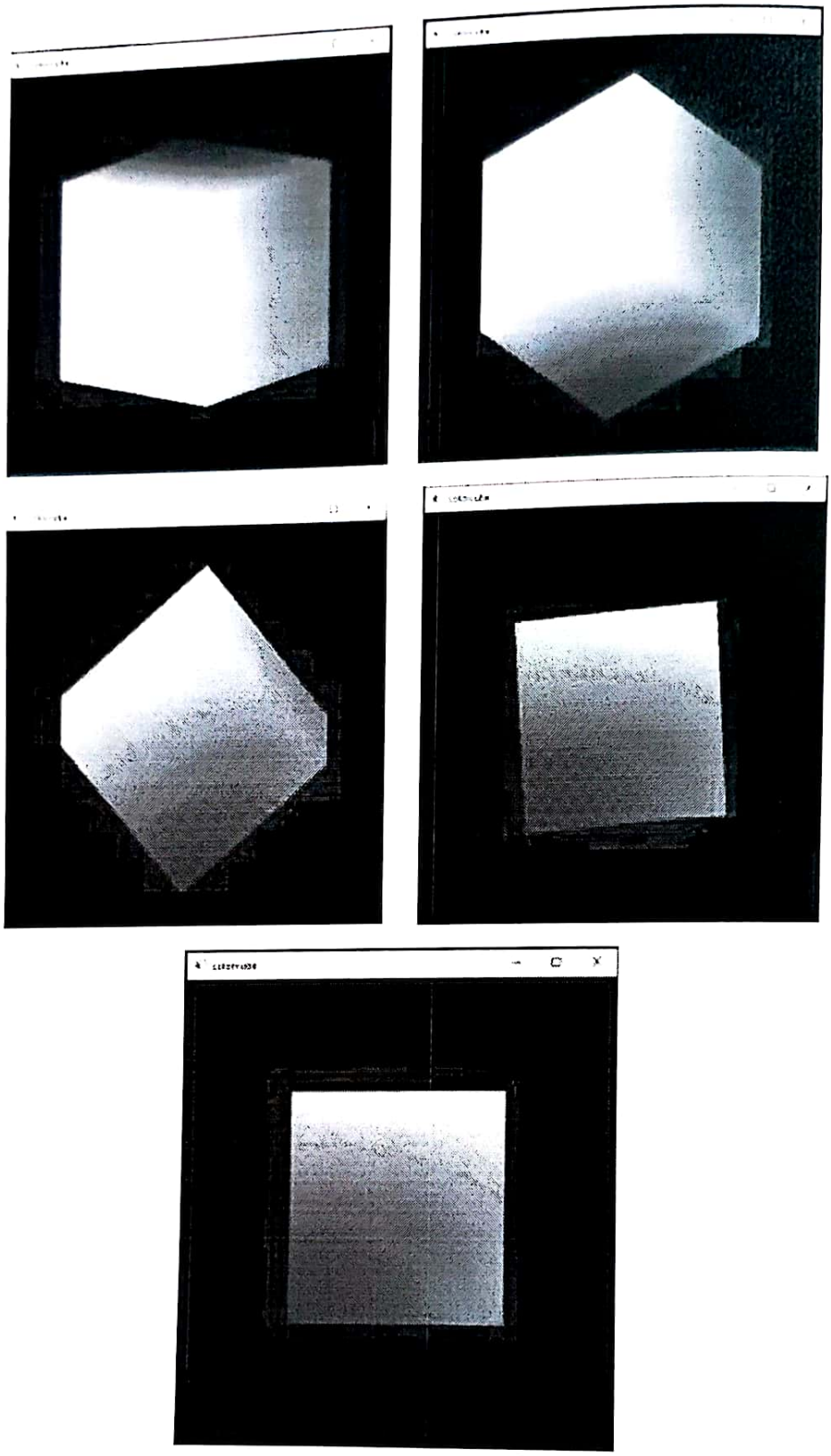
```
    glVertex3f(0,1,0);
    glVertex3f(1,1,1);
    glEnd();
    glFlush();
}
void spinCube(){
    delay(0.01);
    theta[axis] += 2.0;
    if(theta[axis] > 360) theta[axis] -= 360;
    glutPostRedisplay();
}
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutCreateWindow("color cube");
    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);
    glVertexPointer(3, GL_FLOAT, 0, colors);
    glNormalPointer(GL_FLOAT, 0, normals);
    glColor3f(1,1,1);
    glutMainLoop();
}
```

Teacher's Signature:

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

```
        p3e_x = x-250;
        p3e_y = 250-y;
        float exp = (p3e_x - p1e_x) * (p3e_x - p1e_x) +
        (p3e_y - p1e_y) * (p3e_y - p1e_y);
        ry = (int).(sqrt(exp));
        midptellipse();
        paintle_done = 0;
    }
}

void myinit() {
    glClear(1,1,1,1);
        glColor3f(1.0, 0.0, 0.0);
        glPointSize(3.0);
    gluOrtho2D(-250, 250, -250, 250);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0,0);
    // FOR MOUSE
     int id1 = glutCreateWindow("Circle");
     glutSetWindow(id1);
     glutMouseFunc(myMouseFunction(circle));
     glutDisplayFunc(myDrawing());
```

Teacher's Signature:



Program10: Colour Cube

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

Program 11: Create a menu with three entries named curves, colors, quit. The entry curves has a sub menu which has four entries namely Limacon, Cardiod, 3-leaf and Spiral. The color has eight colors of RGB color model. Write a program to create the above hierarchical menu and attach appropriate services to each menu entry with mouse buttons.

```
#include <GL/glut.h>
#include <stdio.h>
#include <math.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;
int red=0, green=0, blue=0;
void myInit(void){
        glClearColor(1,1,1,1);
        glMatrixMode(GL_PROJECTION);
        gluOrtho2D(0,200,0,150);
}
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

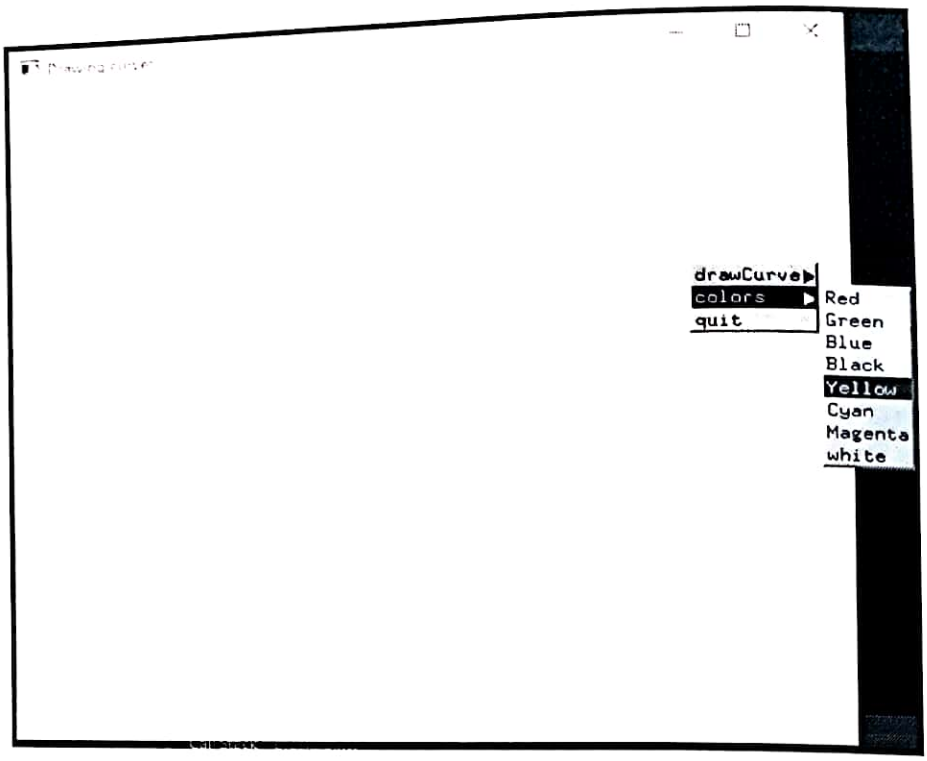
```
void drawCurve (int curveNum) {
    const double twoPi = 6.283185;
    const int a = 175, b = 60;
    float r, theta, dtheta = 1.0 / float(a);
    int x0 = 200, y0 = 250;
    screenPt curvePt[2];
    curve = curveNum;
    glColor3f (red, green, blue);
    curvePt[0].x = x0;
    curvePt[0].y = y0;
    glClear (GL_COLOR_BUFFER_BIT);
    switch (curveNum) {
        case limacon: curvePt[0].x += a+b; break;
        case cardiod: curvePt[0].x += a+a; break;
        case threeLeaf: curvePt[0].x += a; break;
        case spiral: break;
        default: break;
    }
    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]);
```

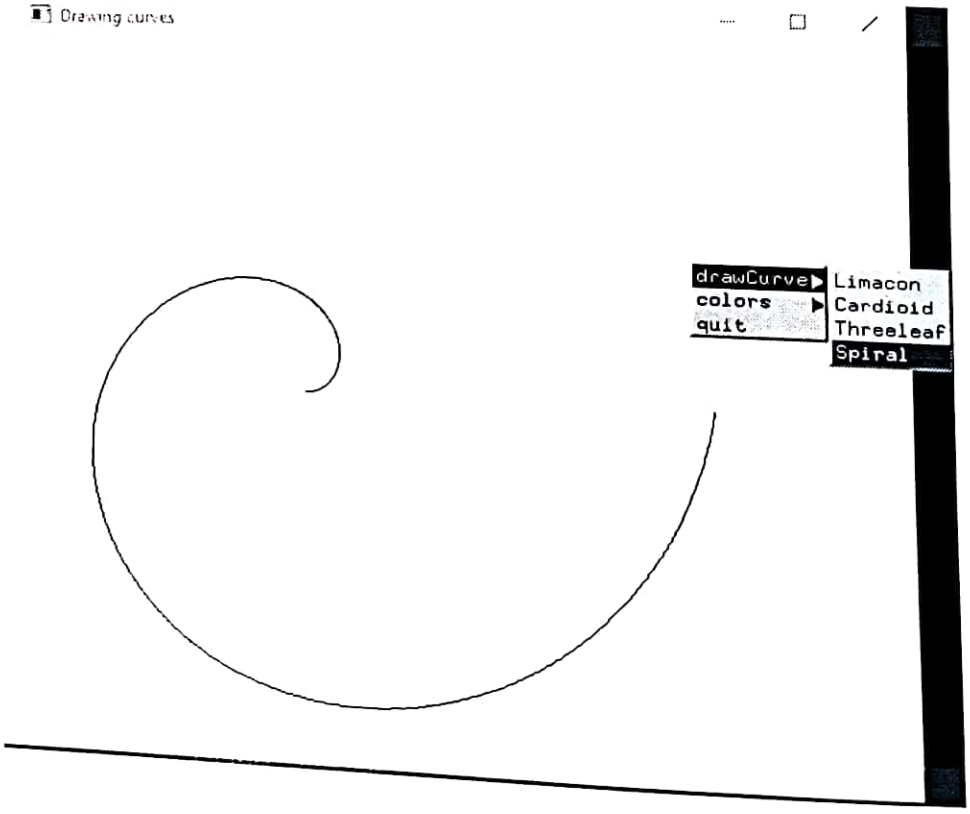
Teacher's Signature:

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

```
        curvePt[0].x = curvePt[1].x;
        curvePt[0].y = curvePt[1].y;
        theta += dtheta;
    }
}
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;
        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 menu(int id){
    switch(id){
        case 3: exit(0); }
}
void mydisplay(){}
void myreshape(int nw, int nh){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, (double)nw, 0.0, (double)nh);
    glClear(GL_COLOR_BUFFER_BIT);
}
```

Teacher's Signature:





Program11: Drawing Curves

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

```
int main (int argc, char ** argv){
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(a,b);
    glutInitWindowPosition(100,100);
    glutCreateWindow ("Drawing Curves");
    int curveId = glutCreateMenu(drawcurve);
    glutAddMenuEntry("Limacon",1);
    glutAddMenuEntry ("cardiod",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);
    glutAddMenuEntry ("White,7);
    glutAttachMenu (GLUT_LEFT_BUTTON);
    glutCreateMenu (main_menu);
    glutAddSubMenu ("drawCurve", curveId);
    glutAddSubMenu ("colors", colorId);
    glutAddSubMenu ("quit",3);
    glutAttachMenu (GLUT_LEFT_BUTTON);
    myInit();
    glutDisplayFunc (mydisplay);
    glutReshapeFunc (myreshape);
    glutMainLoop();
}
```

Teacher's Signature:

3

EXPT NO | NAME

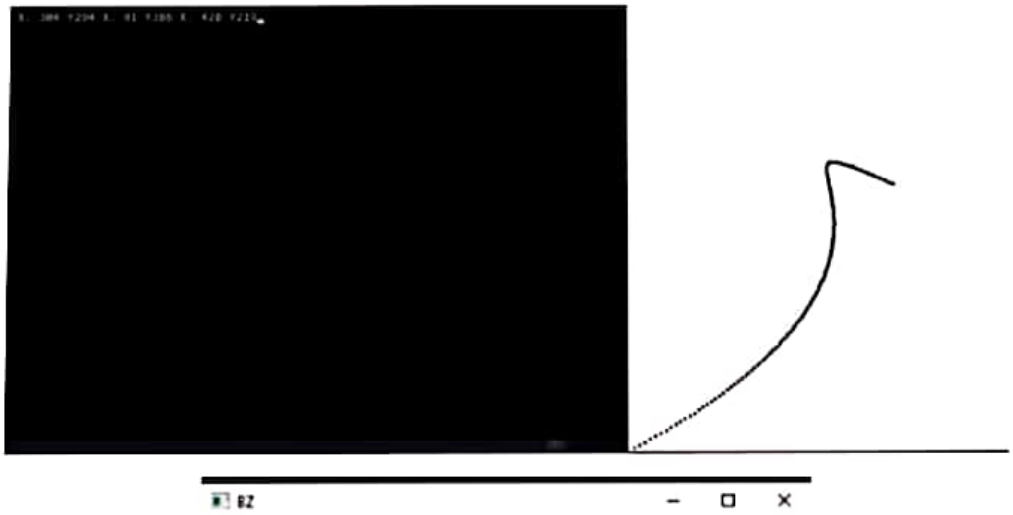
Program 12: Write a program to construct Bezier curve. Control points are supplied through keyboard / mouse.

```
#include <iostream>
#include <math.h>
#include <GL/glut.h>
using namespace std;
float t, g, x, x1[4], y[4];
int flag = 0;
void myInit(){
    glClearColor(1,1,1,1);
    glColor3f(1,1,1);
    glPointSize(5);
    gluOrtho2D(0,500,0,500);
}
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[0]+3*t*pow(1-t,2)*x1[1]+
                    3*pow(t,2)*(1-t)*x1[2]+pow(t,3)*x1[3];
        double yt = pow(1-t,3)*y[0]+3*t*pow(1-t,2)*y[1]+
                    3*pow(t,2)*(1-t)*y[2]+pow(t,3)*y[3];
```

EXPT NO.	NAME:	PAGE NO. DATE	SIGN

```
    glVertex2f(xt,yt);
}
for(i=0; i<4; i++){
        glVertex2f(x[i],y[i]);
        glEnd();
        glFlush();
}}
void mymouse(int btn, int state, int x, int y){
  if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN && flag<4){
        x[flag]=x; y[flag]=500-y;
        glPointSize(3);
        glBegin(GL_POINTS);
        glVertex2i(x, 500-y);
        glEnd();
        glFlush(); flag++;
}}
int main(int argc, char **argv){
    glutInit(&argc,&argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutCreateWindow("BZ");
    glutDisplayFunc(display);
    glutMouseFunc(mymouse);
    myInit();
    glutMainLoop();
}
```

Teacher's Signature:



Program12: BZ