

Name of the Institute ...R.V. College Of Engineering.....

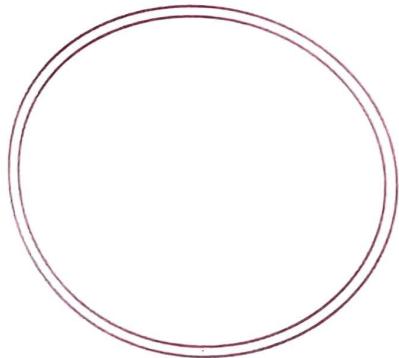
# Laboratory Certificate

This is to certify that Smt./Sri. .... PRIYA DARSHINI.R  
has satisfactorily completed the course of Experiments in  
Practical.... COMPUTER GRAPHICS LAB (16CS73) .... Prescribed by  
the..... R.V. College Of Engineering.....  
University..... —  
in the Laboratory of this College in the year 2020 - 2021

Signature of the Teacher in charge of the batch

Date : .....

Head of the Department



Name of the Candidate .....

..... Reg No. ....

Examination Centre .....

Date of Practical Examination.....

## PROGRAM - 01

write a Program to generate a line using Bresenham's line drawing technique. consider slopes greater than one and slopes less than one. User must able to draw as many lines and specify inputs through Keyboard/ mouse.

```
#include <iostream>
#include <GL/glut.h>
#include <time.h>
#include <stdio.h>
using namespace std;
int x1, x2, y1, y2;
int flag = 0;
```

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

```
void drawLine()
{
    int dx, dy, i, e;
    int incx, incy, incl, incz;
    int x, y;
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 02

$$dx = x_2 - x_1;$$

$$dy = y_2 - y_1;$$

if ( $dx < 0$ )

$$dx = -dx;$$

if ( $dy < 0$ )

$$dy = -dy;$$

$$\text{inc}x = 1;$$

if ( $x_2 < x_1$ )

$$\text{inc}x = -1;$$

$$\text{inc}y = 1;$$

if ( $y_2 < y_1$ )

$$\text{inc}y = -1;$$

$$x = x_1;$$

$$y = y_1;$$

if ( $dx > dy$ )

{

draw\_pixel(x, y);

$$e = 2 * dy - dx;$$

$$\text{inc}1 = 2 * (dy - dx);$$

$$\text{inc}2 = 2 * dy;$$

for (i=0; i < dx; i++)

{ if ( $e > 0$ )

$$\{ y += \text{inc}y;$$

$$e += \text{inc}1;$$

else

$$e += \text{inc}2;$$

$$x += \text{inc}x;$$

draw\_pixel(x, y);

2  
3  
else

chandra's

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No **03**

```

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

```

```

void myinit()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(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)
    }

```

chandras

Name of Experiment.....  
Experiment No. ....

Date.....

Experiment Result.....

Page No. 04

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

4  
y  
break;  
4

void display() { }

```
int main(int ac, char* av[])
{
    // FOR Keyboard
    cout << "x1\n";
    cin >> x1;
    cout << "y1\n";
    cin >> y1;
    cout << "x2\n";
    cin >> x2;
    cout << "y2\n";
    cin >> y2;
    // END OF Keyboard
}
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. **05**

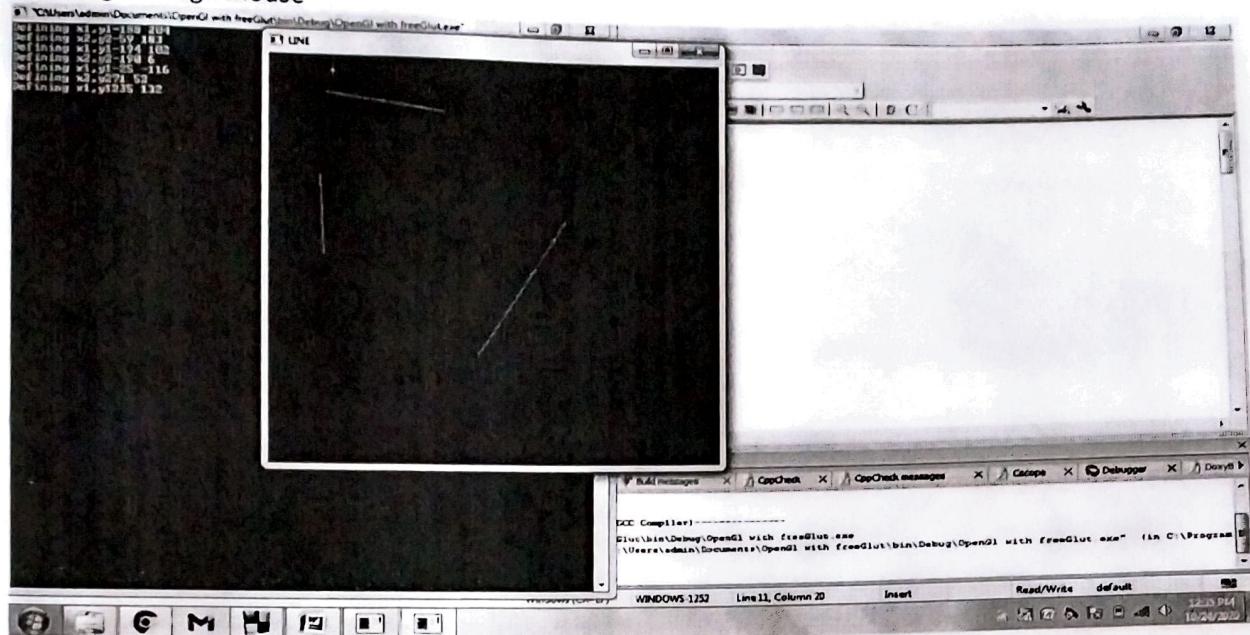
```
glutInit(&ac, &av);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(500, 500);
glutInitWindowPosition(100, 200);
glutCreateWindow("LINE");
myinit();
glutMouseFunc(myMouse); //include to mouse, remove while
                        //using Keyboard
//draw_line(); //include to use Keyboard, remove for mouse
glutDisplayFunc(display);
glutMainLoop();
```

4

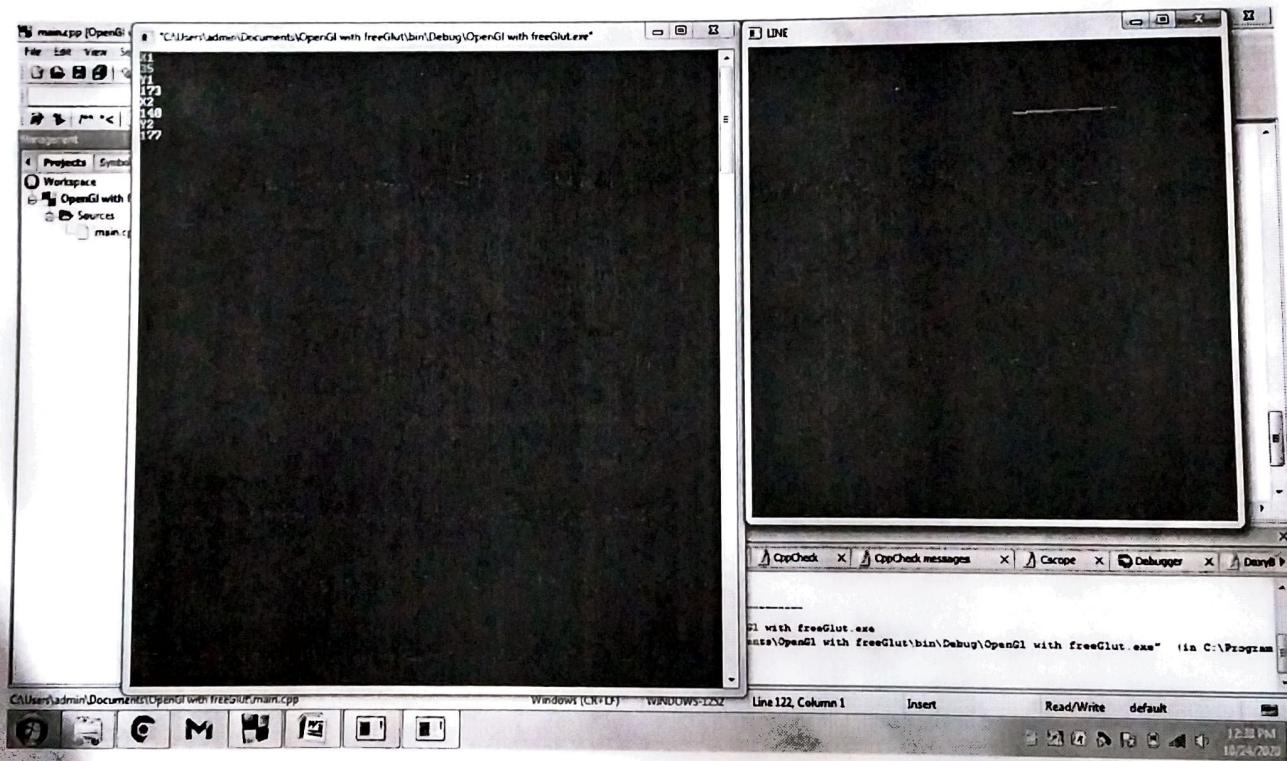
chandra's

## OUTPUT:

### 1. Inputting Through Mouse



### 2. Inputting Through Keyboard



## PROGRAM - 02

write a program to generate a circle and ellipse using Bresenham's circle drawing and ellipse drawing technique. Use two windows to draw circle in one window and ellipse in the other window. User can specify inputs through keyboard/mouse.

```
#include<gl/glut.h>
#include<stdio.h>
#include<math.h>
int xc, yc, r;
int rx, ry, xcc, ycc;
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();
}
```

4

void circlebres()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 07

```
int x=0, y=8;  
int d = 3 - 2*x;  
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 myMousefunc(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;
```

```
q
```

```
else if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
```

```
{
```

chandra's

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No.

08

$$P_2-x = x-250;$$

$$P_2-y = 250-y;$$

$$x_c = P_1-x;$$

$$y_c = P_1-y;$$

$$\text{float exp} = (P_2-x - P_1-x) + (P_2-y - P_1-y) *$$

$$(P_2-y - P_1-y);$$

$$r = (\text{int})(\sqrt{\text{exp}});$$

circle(xc, yc, r);

point1-done = 0;

4

7

// Ellipse drawing

void draw-ellipse(int xc, int yc, int x, int y)

{

glBegin(GL\_POINTS);

glVertex2i(x+xc, y+yc);

glVertex2i(-x+xc, y+yc);

glVertex2i(x+xc, -y+yc);

glVertex2i(-x+xc, -y+yc);

glEnd();

8

void midptellipse()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

float dx, dy, d1, d2, x, y;

x=0;

y=yc;

chandra's

Name of Experiment.....

Experiment No.....

Date.....

Experiment Result.....

Page No. 09

$$dI = (r_y * r_y) - (r_x * r_x * r_y) + (0.25 * r_x * r_x);$$

$$dx = 2 * r_y * r_y * x;$$

$$dy = 2 * r_x * r_x * y;$$

```
while (dx < dy)
```

```
{ draw_ellipse(xce, yce, x, y);
```

```
if (dI < 0)
```

```
{
```

```
x++;
```

$$dx = dx + (2 * r_y * r_y);$$

$$dI = dI + dx + (r_y * r_y);$$

```
}
```

```
else
```

```
{
```

```
x++;
```

```
y--;
```

$$dx = dx + (2 * r_y * r_y);$$

$$dy = dy - (2 * r_x * r_x);$$

$$dI = dI + dx - dy + (r_y * r_y);$$

```
}
```

```
}
```

$$d2 = ((r_y * r_y) * ((x+0.5) * (x+0.5))) + ((r_x * r_x) * ((y-1) * (y-1))) - (r_x * r_x * r_y * r_y);$$

```
while (y >= 0)
```

```
{
```

```
draw_ellipse(xce, yce, x, y);
```

```
if (d2 > 0)
```

```
{
```

```
y--;
```

Name of Experiment.....  
Experiment No. ....

Date.....

Experiment Result.....

Page No. **10**

$dy = dy - (2 * rx * rx);$

$d2 = d2 + (rx * rx) - dy;$

4

else

{      $y--;$

$x++;$

$dx = dx + (2 * ry * ry);$

$dy = dy - (2 * rx * rx);$

$d2 = d2 + dx - dy + (rx * rx);$

4

4

$gflush();$

4

int P1e-x, P2e-x, P1e-y, P3e-x, P3e-y;

int point1e-done = 0;

void myMouseFunc(int button, int state, int x, int y)

{

if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN &&

point1e-done == 0)

{     P1e-x = x - 250;

    P1e-y = 250 - y;

    xce = P1e-x;

    yce = P1e-y;

    point1e-done = 1;

4

else if(button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN &&

point1e-done == 1)

4

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No.

11

$$P_{2e-x} = x - 250;$$

$$P_{2e-y} = 250 - y;$$

$$\text{float } exp = (P_{2e-x} - P_{1e-x}) * (P_{2e-x} - P_{1e-x}) + (P_{2e-y} - P_{1e-y}) \\ * (P_{2e-y} - P_{1e-y});$$

$$rx = (\text{int})(\sqrt{exp});$$

$$\text{point1e-done} = 2;$$

4

else if (button == GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN  
&& point1e-done == 2)

d

$$P_{3e-x} = x - 250;$$

$$P_{3e-y} = 250 - y;$$

$$\text{float } exp = (P_{3e-x} - P_{1e-x}) * (P_{3e-x} - P_{1e-x}) + (P_{3e-y} - P_{1e-y}) \\ * (P_{3e-y} - P_{1e-y});$$

$$ry = (\text{int})(\sqrt{exp});$$

midptellipse();

$$\text{point1e-done} = 0;$$

4

4

void myDrawing()

{ 4

void myDrawing()

{ 4

void minit() {

glClearColor(1, 1, 1, 1);

glClearDepth(1.0, 0.0, 0.0);

glPointSize(3.0);

gluOrtho2D(-250, 250, -250, 250);

4

chandras

```
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(myMouseFuncCircle);
```

```
    glutDisplayFunc(myDrawing);
```

```
    mInit();
```

```
    glutInitWindowSize(500, 500);
```

```
    glutInitWindowPosition(600, 100);
```

```
    int id2 = glutCreateWindow("ellipse");
```

```
    glutSetWindow(id2);
```

```
    glutMouseFunc(myMouseFunc());
```

```
    glutDisplayFunc(myDrawing);
```

```
// end mouse
```

```
// for keyboard
```

```
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", &x0, &y0, &r);
```

```
    glutCreateWindow("circle");
```

Name of Experiment.....  
Experiment No.....

Date.....

Experiment Result.....

Page No. 13

glutDisplayFunc(circles);

break;

case 2:

printf("Enter coordinates of centre of ellipse & major  
and minor radii as ln");

scanf("%d %d %d %d", &xcp, &ycp, &r1, &r2);

glutCreateWindow("Ellipse");

glutDisplayFunc(midptellipse);

break;

4

//End of keyboard

minit();

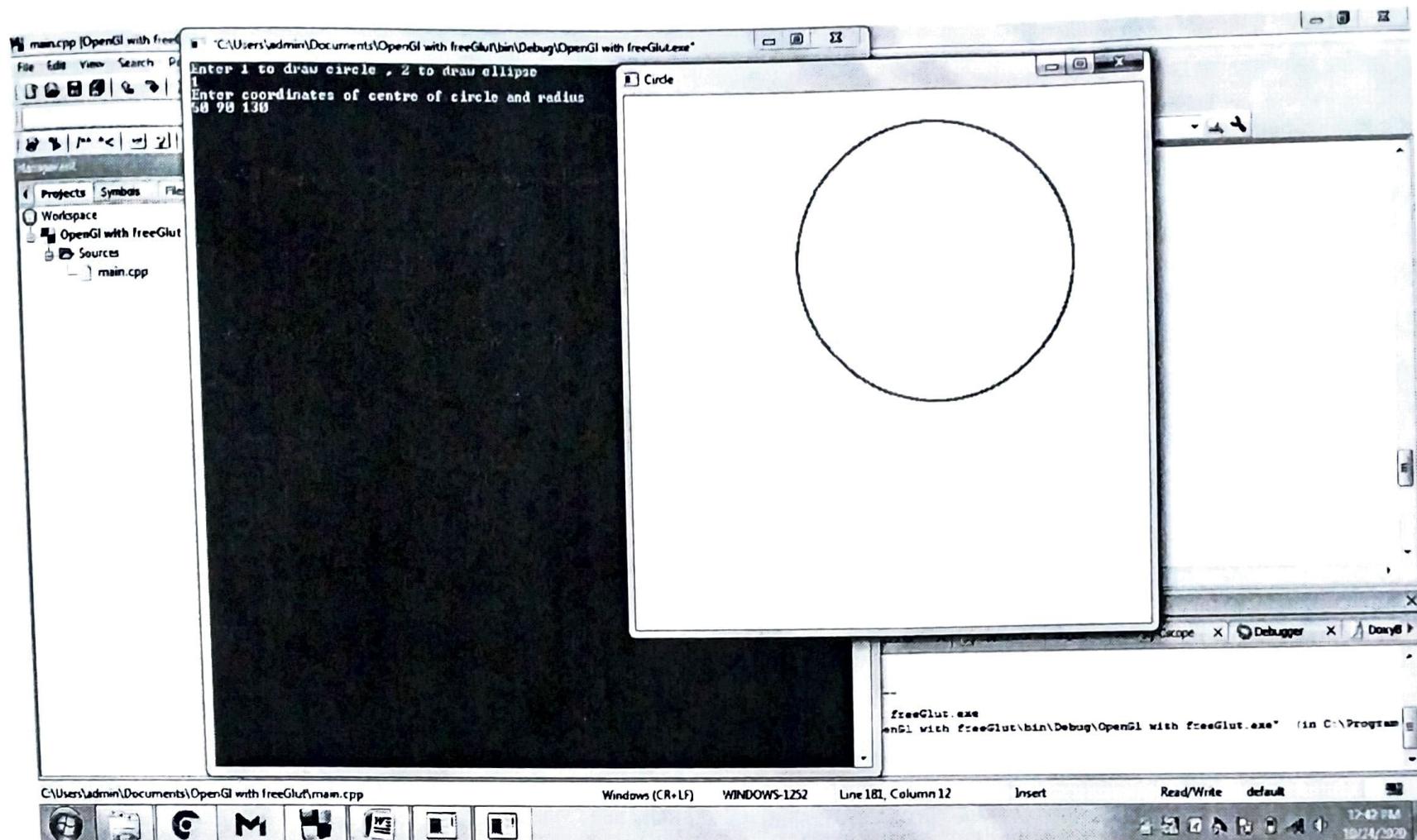
glutMainLoop();

5

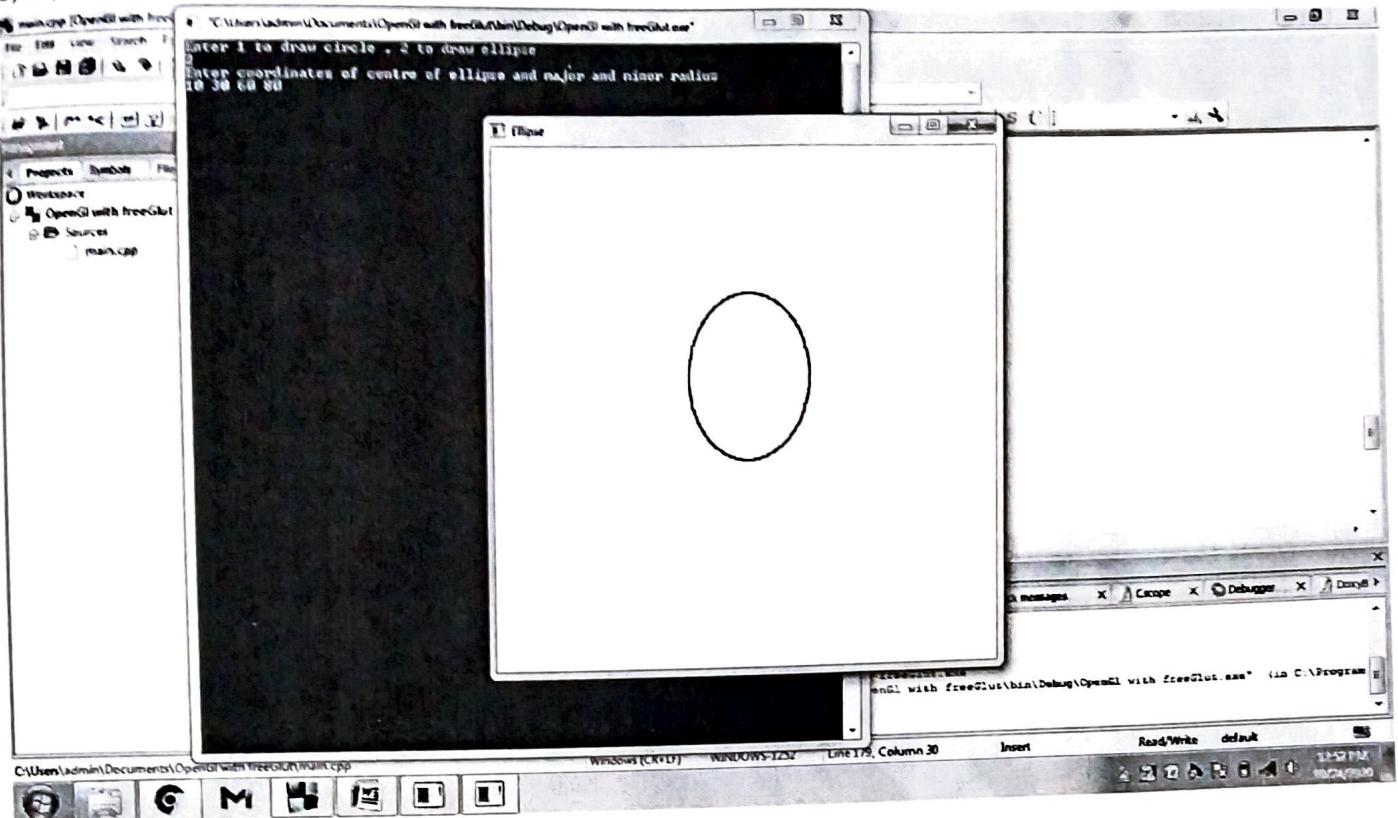
chandras

## OUTPUT :

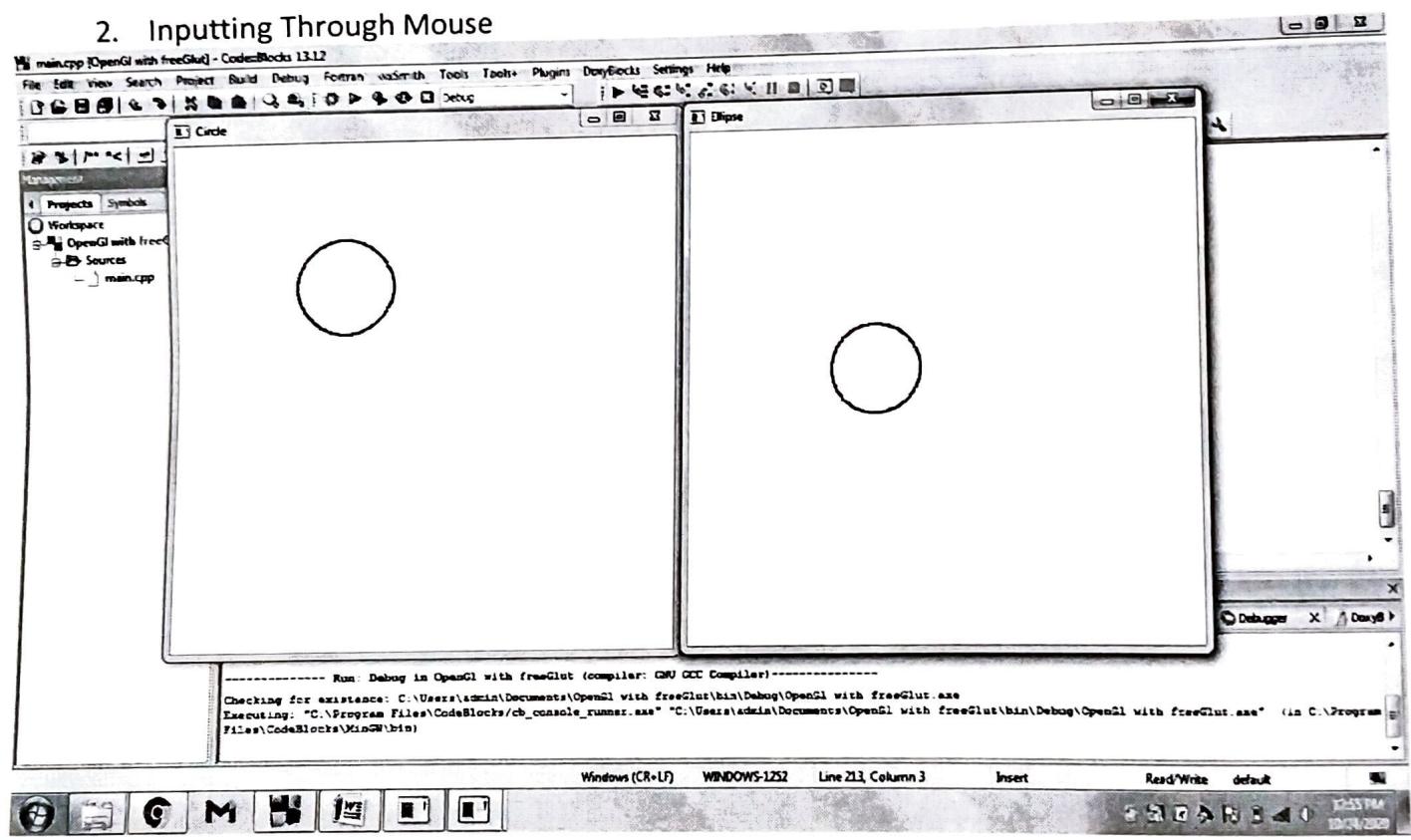
### 1. a) Inputting Through Key board



b) Inputting Through keyboard



## 2. Inputting Through Mouse



## PROGRAM - 03

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

```
#include <gl/glut.h>
#include <stdio.h>
int m;
typedef float point[3];
point tetric[4] = {{0, 100, -100}, {0, 0, 100}, {100, -100, -100}, {100, -100, 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("%f", &m);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(100, 200);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Sierpinski Gasket");
    glutDisplayFunc(tetrahedron);
    glEnable(GL_DEPTH_TEST);
    myinit();
    glutMainLoop();
}
```

```
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;
```

```
        for(j=0; j<3; j++)
```

```
            v2[j] = (a[j]+c[j])/2;
```

```
        for(j=0; j<3; j++)
```

```
            v3[j] = (b[j]+c[j])/2;
```

```
        divide_triangle(a, v1, v2, m-1);
```

```
        divide_triangle(c, v2, v3, m-1);
```

```
        divide_triangle(b, v3, v1, m-1);
```

```
    } else
```

```
        draw_triangle(a, b, c);
```

```
void myinit()
```

```
glClearColor(1,1,1,1);
```

```
glOrtho(-500.0, 500.0, -500.0, 500.0, -500.0, 500.0);
```

```
void draw_triangle(point p1, point p2, point p3)
```

```
}
```

```
glBegin(GL_TRIANGLES);
```

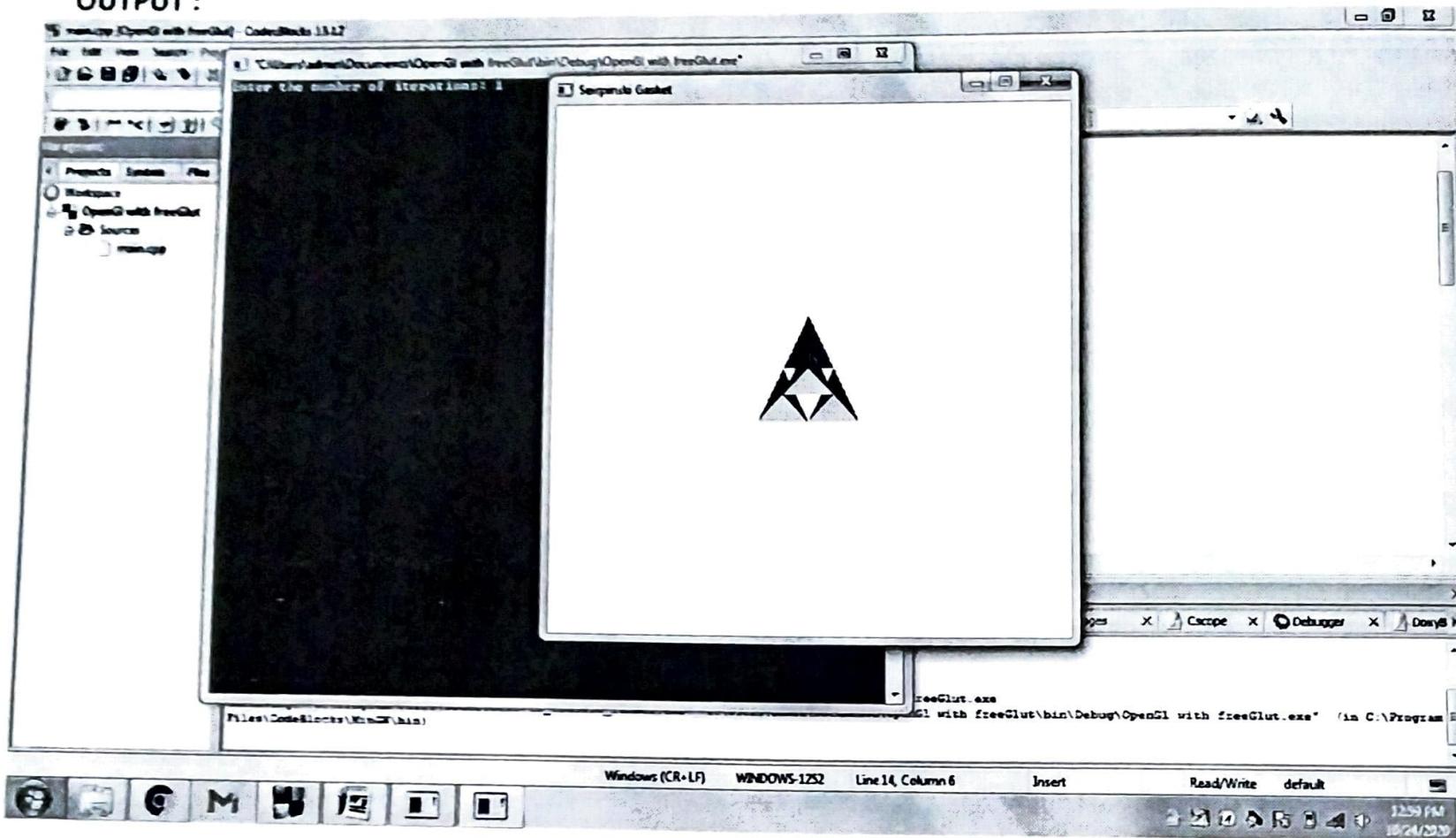
```
glVertex3fv(p1);
```

```
glVertex3fv(p2);
```

```
glVertex3fv(p3);
```

```
glEnd();
```

## OUTPUT :



## PROGRAM - 04

write a program to fill any given polygon using scan-line area filling algorithm.

```
#include<cslib.h>
#include<gl/glut.h>
#include<algorithm>
#include<iostream>
#include<windows.h>
#include<stdio.h>
using namespace std;
float x[100], y[100];
int n, m;
int wx=500, wy=500;
static float intz[10]={0};
void draw_line(float x1, float y1, float x2, float y2){
    sleep(100);
    glColor3f(1,0,0);
    glBegin(GL_LINES);
    glVertex2f(x1,y1);
    glVertex2f(x2,y2);
    glEnd();
    glFlush();
```

4

```
    temp = x1, x1 = x2, x2 = temp,
    temp = y1, y1 = y2, y2 = temp,
```

4

```
    if (scandline > y1 && scandline < y2)
        int z[m+1] = x1 + ((scandline - y1) *
                            (x2 - x1) / (y2 - y1));
    void myInit(){
        glEnable(GL_COLOR);
        glColor3f(0,0,1);
        glPointSize(1);
        glOrtho2D(0,wx,0,wy);
```

4

```
    void display_filled_polygon(){
        glClear(GL_COLOR_BUFFER_BIT);
        glLineWidth(2);
        glBegin(GL_LINE_LOOP);
        for(int i=0; i<n, i++)
            glVertex2f(x[i], y[i]);
        glEnd();
        scantoil(x,y);
```

4

```
    void edgeDetect(float x1, float y1, float x2, float y2, int scandline){
        float temp;
        if (y2 < y1){
```

```
void sconfig (float x[], float y[]){  
    for (int s1=0; s1<=m; s1++) {  
        m=0;  
        for (int i=0; i<n; i++) {  
            edgeDetect (x[i], y[i], x[i+1], y[i+1], s1);  
            sort (int x, float x+m));  
            if (m>=2)  
                for (int i=0; i<m; i=i+2)  
                    drawLine (int x[i], s1, int x[i+1], s1);  
    }  
}
```

```
void main (int ac, char* avs[]){  
    glutInit (&ac, avs);  
    printf ("Enter no of sides : \n");  
    scanf ("%d", &n);  
    printf ("Enter coordinates of endpoints : \n");  
    for (int i=0; i<n; i++)
```

```
{    printf ("X-Coord Y-Coord : \n");  
    scanf ("%f %f", &x[i], &y[i]);  
}
```

4

```
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
```

```
glutInitWindowSize (500, 500);
```

```
glutInitWindowPosition (0, 0);
```

```
glutCreateWindow ("sconfig");
```

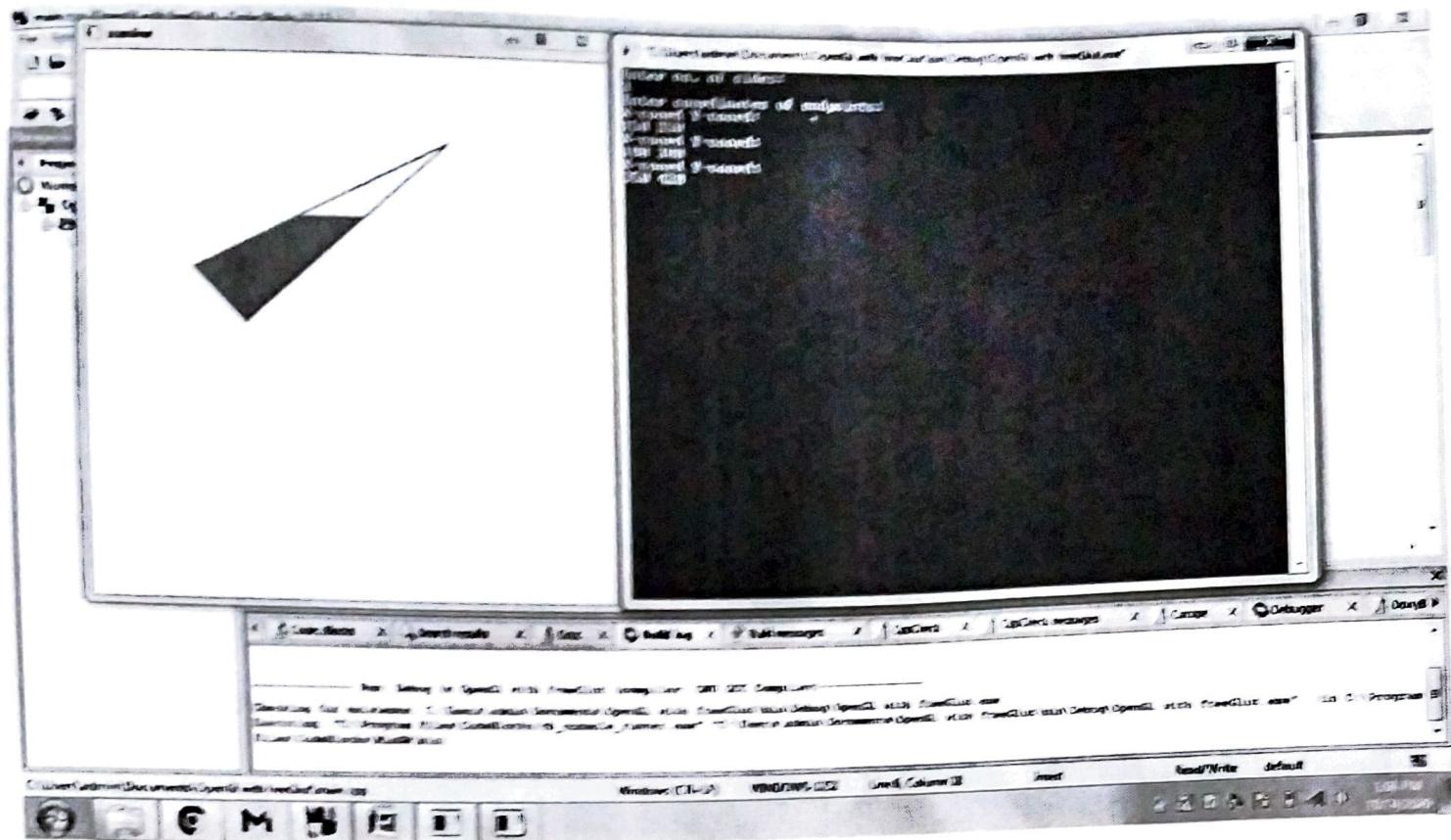
```
glutDisplayFunc (displayFilledPolygon);
```

```
myInit();
```

```
glutMainLoop();
```

4

## OUTPUT :



## PROGRAM - 05

Write a program to create a house like a figure and perform the following operations

- i) Rotate it about a given fixed point using OpenGL transformation functions.
- ii) Reflect it about an axis  $y=mx+c$  using OpenGL transformation functions.

```
#include<gl/glut.h>
#include<cmath.h>
#include<stdio.h>

float house[11][2]={{100,200}, {200,250}, {300,200}, {100,200}, {100,100},
{175,100}, {175,150}, {225,150}, {225,100}, {300,100}, {300,200}, {200,200};

int angle;
float m, c, theta;
void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D (-450, 450, -450, 450);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    //normal house
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
    for (int i=0; i<11; i++)
        glVertex2fv(&house[i]);
    glEnd();
    glFlush();
}
```

chandas

```
void display2() {
    glClearColor(1,1,1,0);
    glEnable(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-450, 450, -450, 450);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    //normal house
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
    for (int i=0; i<11; i++)
        glVertex2fv(house[i]);
    glEnd();
    glFlush();
}
```

```
//line
float x1 = 0, x2 = 500;
float y1 = m * x1 + c;
float y2 = m * x2 + c;
	glColor3f(1,1,0);
 glBegin(GL_LINES);
 glVertex2f(x1, y1);
 glVertex2f(x2, y2);
}
```

```
glEnd();
glFlush();
```

//Reflected

```
glPushMatrix();
glTranslate(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);
```

```
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);
```

}

```
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(); }
```

chandra's

Name of Experiment.....

Date.....

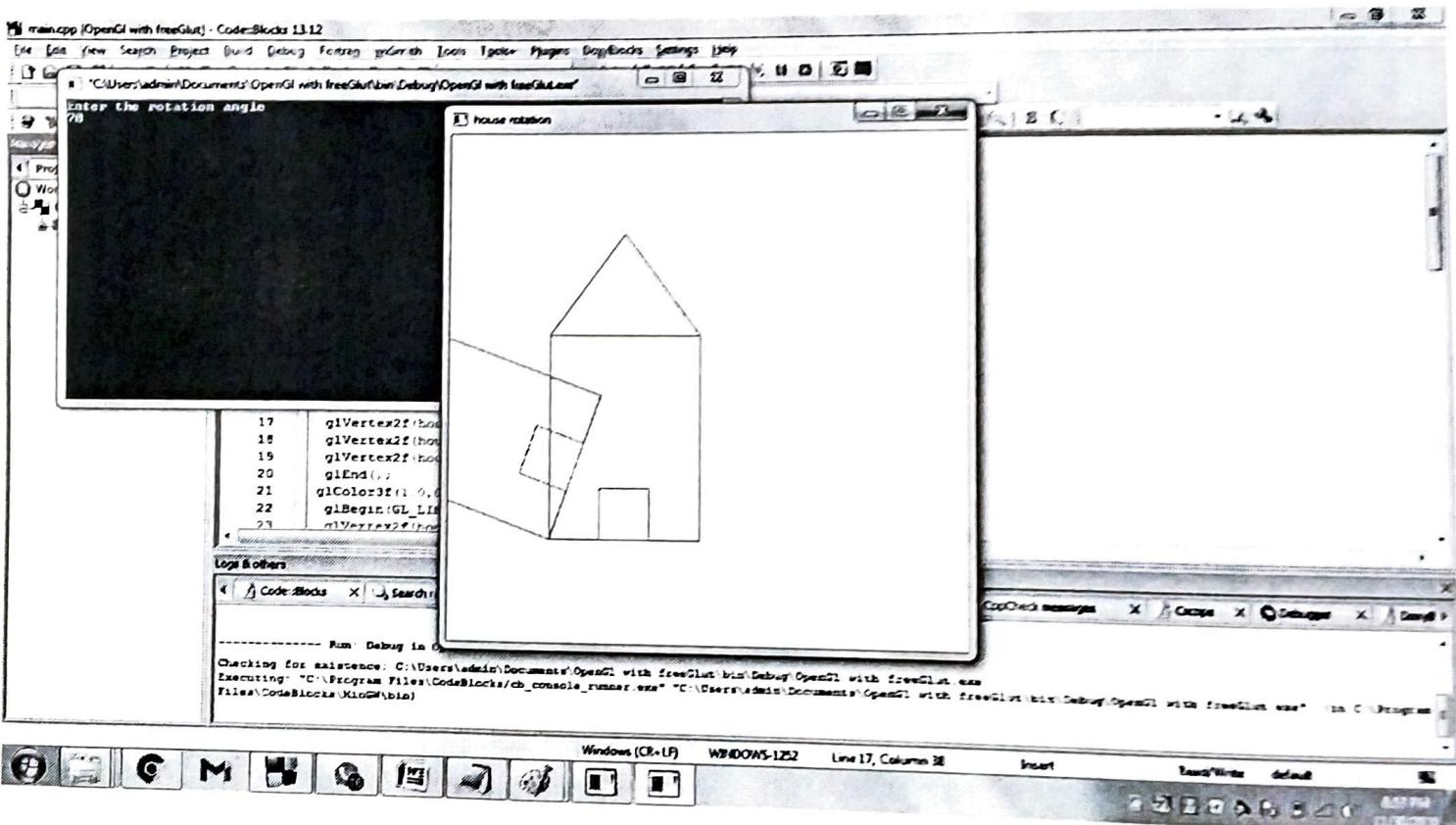
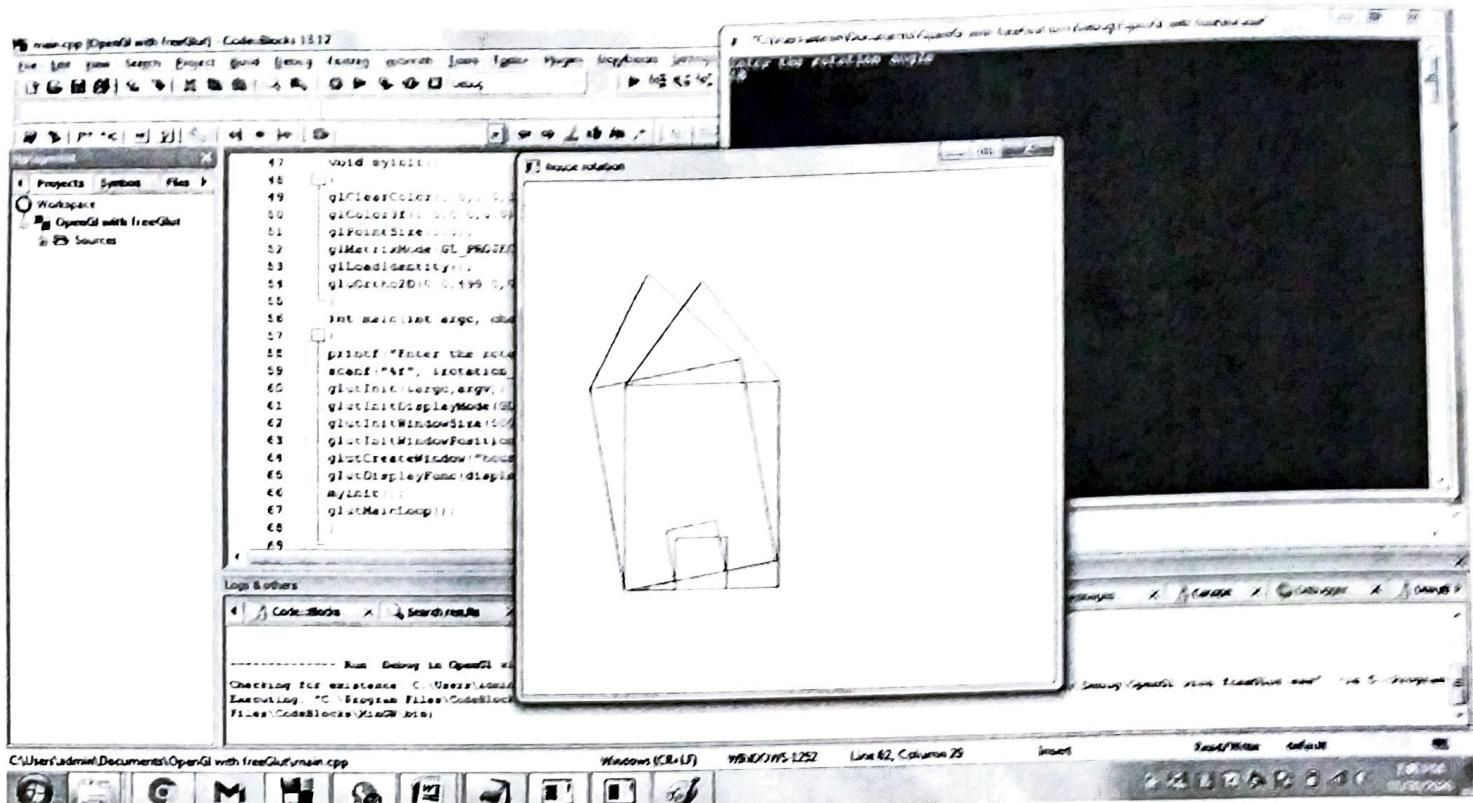
Experiment No.....

Experiment Result.....

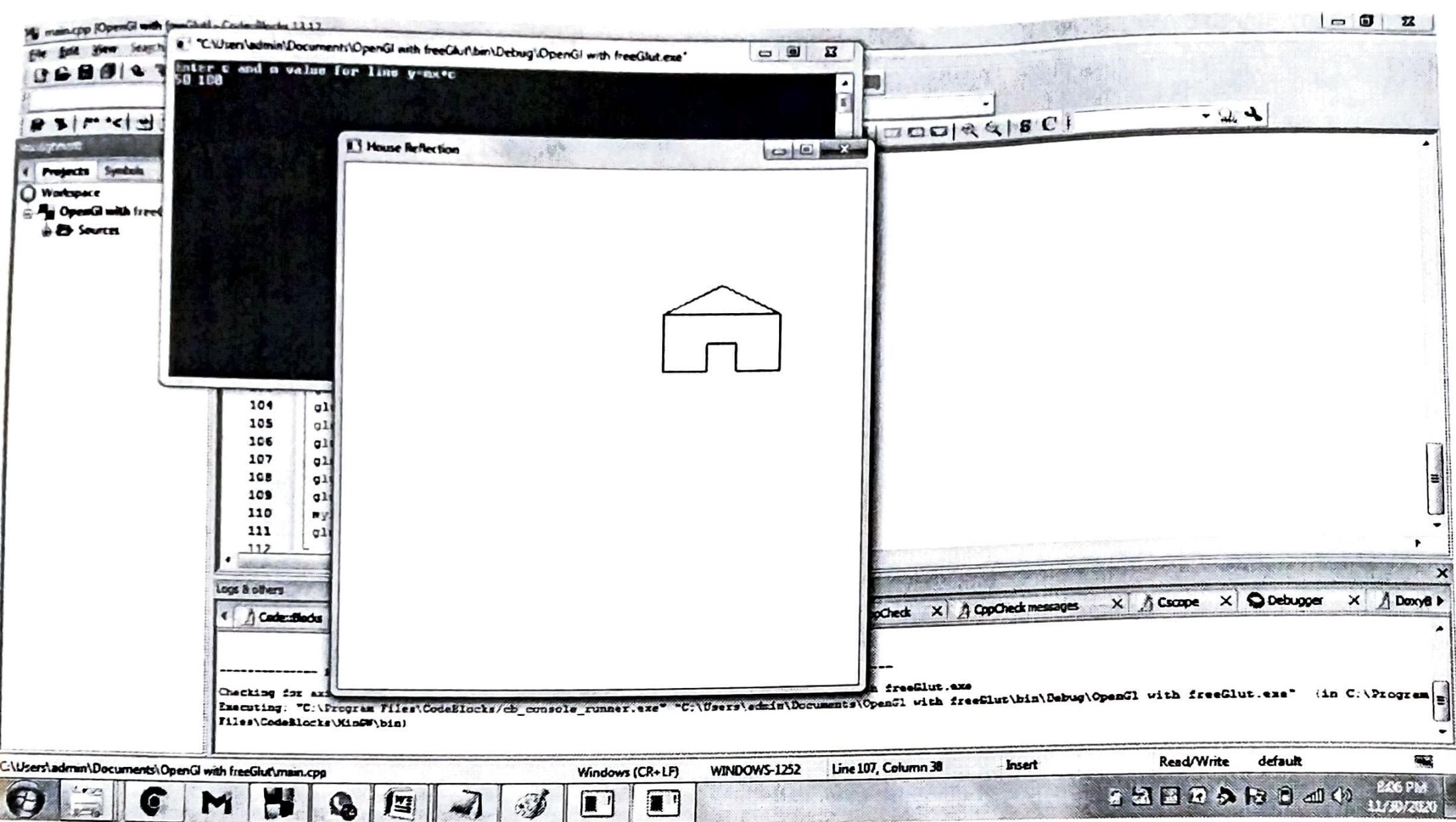
Page No. **20**

```
void main(int argc, char** argv)
{
    printf("Enter the rotation angle");
    scanf("%d", &angle);
    printf("Enter c and m value for line y=mx+c [m]");
    scanf("%f %f", &c, &m);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(900, 900);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Home rotation");
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    myInit();
    glutMainLoop(); q
```

## OUTPUT:



# OUTPUT:



## PROGRAM - 6

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

```

#include <stdio.h>
#include <conio.h>
double xmin=50, ymin=50,
      xmax=100, ymax=100;
double xumin=200, yumin=200,
      xmax=300, ymax=300;
const int RIGHT=8;
const int LEFT=2;
const int TOP=4;
const int BOTTOM=1;
outcode ComputeOutCode(double x,
                      double y);
void CohenSutherland(double x0, double y0, double x1, double y1)
{
    outcode outcode0, outcode1, outcodeOut; // outcode for P0 & P1
    bool accept=false, done=false;           // point lies outside the clipregion
    outcode0 = ComputeOutCode(x0, y0);       // compute outcode
    outcode1 = ComputeOutCode(x1, y1);       // compute outcode
    do {
        if (! (outcode0 | outcode1)) // logical OR is 0 trivially accept & exit
        {
            accept=true;
            done=true;
        }
        else if (outcode0 & outcode1) // logical AND is not 0. Trivially reject
            done=true;
        else
    } while (!done);
}

```

//if both tests, so calculate the line segment to clip from an outside point to an intersection with clip edge  
double x, y;

//at least one endpoint is outside the clip rectangle; pick it.

//now find the intersection point

if (outcodeOut & TOP) //point is above clip rectangle

{

$$x = x_0 + (x_1 - x_0) * (y_{max} - y_0) / (y_1 - y_0);$$

$$y = y_{max};$$

else if (outcodeOut & BOTTOM) //point is below the clip rectangle

$$x = x_0 + (x_1 - x_0) * (y_{min} - y_0) / (y_1 - y_0);$$

$$y = y_{min};$$

}

else if (outcodeOut & RIGHT) //point is to the right of clip rectangle

$$\{ \quad y = y_0 + (y_1 - y_0) * (x_{max} - x_0) / (x_1 - x_0);$$

$$x = x_{max}; \quad \}$$

else

$$\{ \quad y = y_0 + (y_1 - y_0) * (x_{min} - x_0) / (x_1 - x_0);$$

$$x = x_{min}; \quad \}$$

//now we move Outside Point to intersection point to clip

if (outcodeOut == outcode0)

$$\{ \quad x_0 = x; \quad$$

$$y_0 = y;$$

$$\text{outcode0} = \text{computeOutCode}(x_0, y_0);$$

}

else

$$\{ \quad x_1 = x;$$

$$y_1 = y;$$

$$\text{outcode1} = \text{computeOutCode}(x_1, y_1);$$

```

3 while (!done);
if (accept)
{
    //window to viewport mappings
    double sx = (xumax - xumin) / (xmax - xmin); //scale parameter
    double sy = (yumax - yumin) / (ymax - ymin);
    double vx0 = xumin + (x0 - xmin) * sx;
    double vy0 = yumin + (y0 - ymin) * sy;
    double vx1 = xumin + (x1 - xmin) * sx;
    double vy1 = yumin + (y1 - ymin) * sy;
    glClear(GL_COLOR_BUFFER_BIT); // draw a red colored view port
    glColor3f(1.0, 0.0, 0.0)
    glBegin(GL_LINE_LOOP);
    glVertex2f(xumin, yumin);
    glVertex2f(xumax, yumax);
    glVertex2f(xmax, ymax);
    glVertex2f(xmin, ymax);
    glEnd();
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
    glVertex2d(vx0, vy0);
    glVertex2d(vx1, vy1);
    glEnd();
}

```

4)

diagonally by  $(xmin, ymin)$  &  
 $(xmax, ymax)$

//compute bitcode for a point  $(x, y)$  using clip rectangle bounded  
 $\begin{array}{l} \text{outcode ComputeOutCode(double x, double y)} \\ \{ \text{outcode code=0; } \end{array}$

$\text{if } (y > ymax) \text{ // above the clip window}$

$\text{code1 = TOP; }$

$\text{else if } (y < ymin) \text{ // below the clip window}$

chandras

$\text{code1 = BOTTOM; }$

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No 24

```
if (x > xmax) // to the right of the clip window
    code1 = RIGHT;
```

```
else if (x < xmin) // to the left of clip window
    code1 = LEFT;
return code1;
```

4

```
void display()
```

```
{ double x0=60, y0=20, x1=80, y1=120;
glClear(GL_COLOR_BUFFER_BIT);
```

```
// draw the line with red color
	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_LINES_LOOP);
```

```
 glVertex2f(xmin, ymin);
```

```
 glVertex2f(xmax, ymin);
```

```
 glVertex2f(xmax, ymax);
```

```
 glVertex2f(xmin, ymax);
```

```
 glEnd();
```

```
 glColor3f(x0, y0, x1, y1);
```

```
 glFlush();
```

4 void myinit() { }

```
int main(int argc, char ** argv)
```

```
{ glutInit(&argc, argv);
```

```
 glutDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

```
 glutInitWindowSize(500, 500);
```

chandra's

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

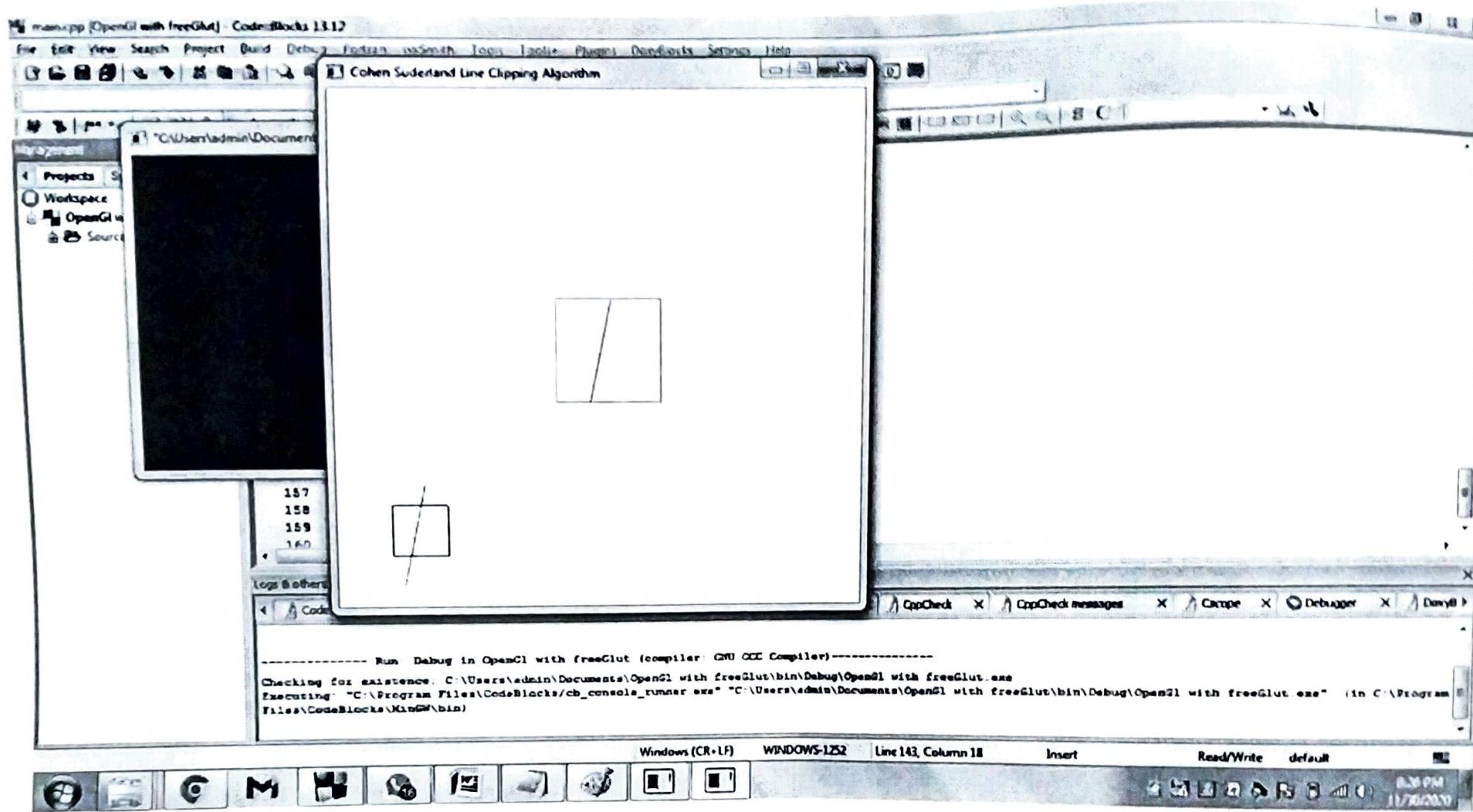
Page No **25**

```
glutInitWindowPosition(0,0);  
glutCreateWindow ("coleen sutherland line clipping algo");  
glutDisplayFunc(display);  
myInit();  
glutMainLoop();
```

4

chandra's

## OUTPUT:



## PROGRAM 7

Write a program to implement the Liang-Barsky line clipping algorithm. make provision to specify the IP for multiple lines, window for clipping & viewport for displaying the clipped image.

```
#include<stdio.h>
#include<cbl/glut.h>
double xmin=50, ymin=50, xmax=100, ymax=100; //window boundaries
double xumin=200, yumin=200, xumax=300, yurnax=300; //viewport boundaries
int cliptest(double p, double q, double *t1, double *t2)
{
    double t = q/p;
    if(p<0.0) //potentially entry point, update t
    {
        if(t>*t1)*t1=t;
        if(t>*t2) return(false); //line portion is outside
    }
    else
    {
        if(p>0.0) //potentially leaving point, update t'
        {
            if(t<*t2)*t2=t;
            if(t<*t1) return(false); //line portion is outside
        }
    }
    if(p==0.0)
    {
        if(q<0.0) return(false); //line parallel to edge but
        //outside
    }
    return(true);
}
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 27

```

void LiangBarskyLineClip(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)) // inside test w.r.t left edge
    if (clipTest(dx, xmax - x0, &te, &t1)) // - " - right edge
    if (clipTest(dy, y0 - ymin, &te, &t1)) // bottom
    if (clipTest(dy, ymax - y0, &te, &t1)) // top
    {
        if (t1 < 0)
        {
            x1 = x0 + t1 * dx;
            y1 = y0 + t1 * dy;
        }
        if (te > 0.0)
        {
            x0 = x0 + te * dx;
            y0 = y0 + te * dy;
        }
    }

    double sx = (xmax - xmin) / (xmax - xmin);
    double sy = (ymax - ymin) / (ymax - ymin);
    double vx0 = xmin + (x0 - xmin) * sx;
    double vy0 = ymin + (y0 - ymin) * sy;
    double vx1 = xmin + (x1 - xmin) * sx;
    double vy1 = ymin + (y1 - ymin) * sy;
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2f(xmin, ymin);
    glVertex2f(xmax, ymin);
    glVertex2f(xmax, ymax);
    glVertex2f(xmin, ymax);
    glEnd();
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
    glEnd();
}

```

chandras

## OUTPUT:

main.cpp [OpenGL with freeGlut] - Code::Blocks 13.12

File Edit View Search Project Build Debug Fortran smSmith Tools Tools+ Plugins DocBlocks Settings Help

Liang Barsky Line Clipping Algorithm

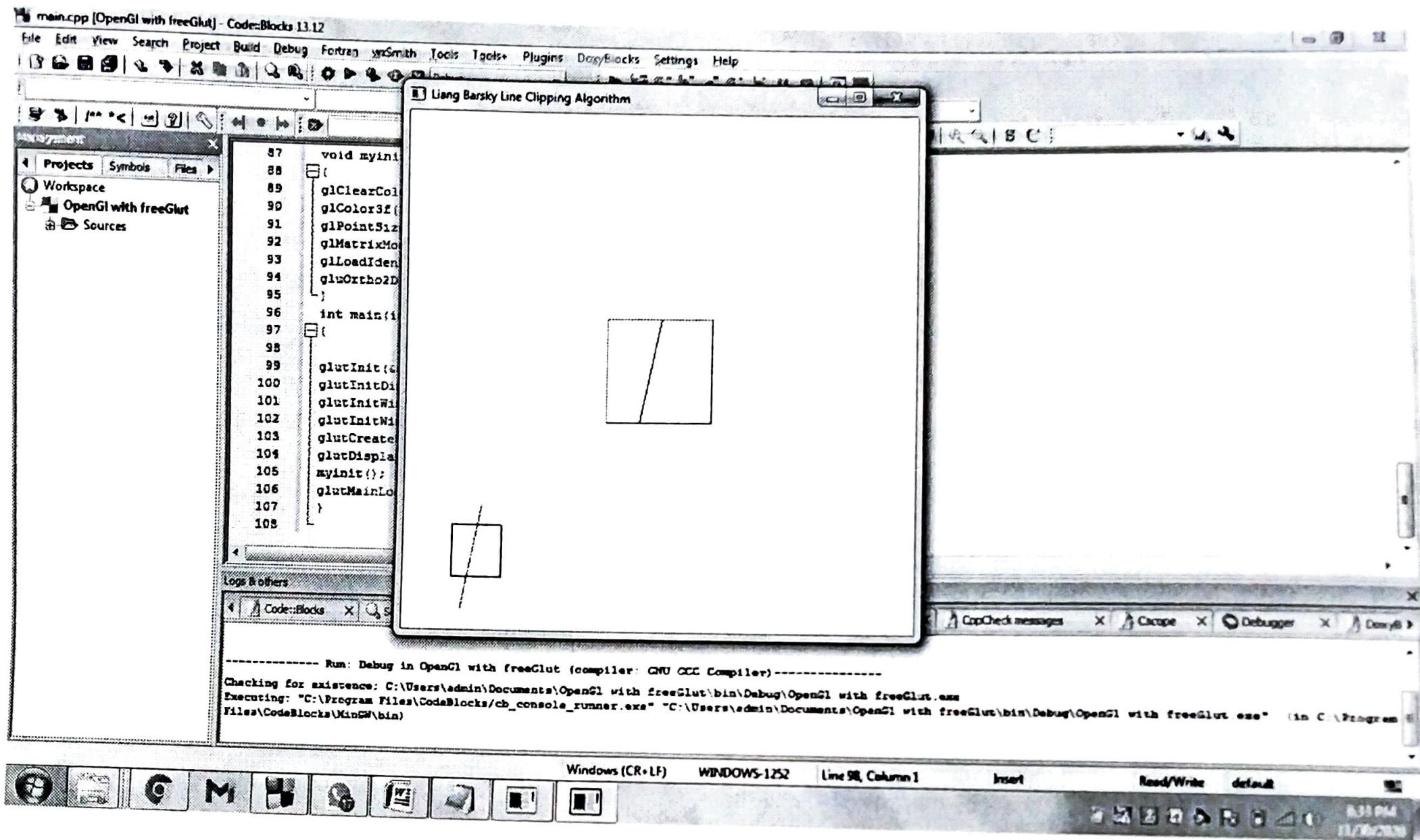
void myinit()  
{  
 glClear(GL\_COLOR\_BUFFER\_BIT);  
 glColor3f(1.0, 1.0, 1.0);  
 glPointSize(1.0);  
 glMatrixMode(GL\_MODELVIEW);  
 glLoadIdentity();  
 gluOrtho2D(-1.0, 1.0, -1.0, 1.0);  
}  
  
int main(int argc, char \*\*argv)  
{  
 glutInit(&argc, argv);  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  
 glutInitWindowPosition(100, 100);  
 glutInitWindowSize(600, 600);  
 glutCreateWindow("Liang Barsky Line Clipping Algorithm");  
 glutDisplayFunc(display);  
 myinit();  
 glutMainLoop();  
}

Logs & Errors

Code::Blocks

----- Run: Debug in OpenGL with freeGlut (compiler: GNU GCC Compiler) -----  
Checking for existence: C:\Users\admin\Documents\OpenGL with freeGlut\bin\Debug\OpenGL with freeGlut.exe  
Executing: "C:\Program Files\CodeBlocks\cb\_console\_runner.exe" "C:\Users\admin\Documents\OpenGL with freeGlut\bin\Debug\OpenGL with freeGlut.exe" (in C:\Program Files\CodeBlocks\MinGW\bin)

Windows (CR+LF) WINDOWS-1252 Line 98, Column 1 Insert Read/Write default 8:31 PM 11/26/2020



Name of Experiment .....

Date .....

Experiment No. ....

Experiment Result.....

Page No. 28

## PROGRAM 8

Write a Program to implement Cohen-hodgeman polygon clipping also. make provision to specify the input polygon & window for clipping.

```
#include <iostream>
#include <GL/glut.h>
#include <stdio.h>
using namespace std;
int poly-size, Poly-points[20][20], org-size, org-Poly-points[20][20],
clipper-size, clipper-points[20][20];
const int MAX_POINTS = 20;
void drawPoly(int p[20], int n) {
    glBegin(GL_POLYGON);
    for (int i=0; i<n; i++)
        glVertex2f(p[i][0], p[i][1]);
    glEnd();
}
int x-intersection(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-intersection(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 - x4) -
(x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
    int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
    return num / den;
}
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 29

```

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;
            new_poly_size++;
        } 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_intersection(x1, y1, x2, y2, ix, iy, kx, ky);
            new_poly_size++;
            new_points[new_poly_size][0]=kx;
            new_points[new_poly_size][1]=ky;
            new_poly_size++;
        } else if (i_pos>=0 && k_pos<0) {
            new_points[new_poly_size][0]=x_intersect(x1, y1, x2, y2, ix, iy);
            new_points[new_poly_size][1]=y_intersection(x1, y1, x2, y2, ix, iy, kx, ky);
            new_poly_size++;
        } else {
            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];
            }
        }
    }
}

```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. **30**

```

void display() {
    init();
    glColor3f(1.0f, 0.0f, 0.0f);
    drawPoly(clipper.points, clipper.size);
    glColor3f(0.0f, 1.0f, 0.0f);
    drawPoly(org_poly.points, org_poly.size);
    for(int i=0; i<clipper.size; i++) {
        int k = (i+1) % clipper.size;
        clip_poly.points[poly_size] = clipper.points[i];
        clipper.points[k] = clipper.points[i];
        clipper.points[i] = clipper.points[k];
        glColor3f(0.0f, 0.0f, 1.0f);
        drawPoly(poly.points, poly.size);
        glFlush();
    }
}

```

```

int main(int argc, char* argv[]) {
    printf("Enter no of vertices");
    scanf("%d", &poly_size);
    org_poly_size = poly_size;
    for(int i=0; i<poly_size; i++) {
        printf("polygon vertex");
        s = "%d %d", &poly.points[i][0], &poly.points[i][1];
        org_poly.points[i] = poly.points[i];
        org_poly.points[i][0] = poly.points[i][1];
        org_poly.points[i][1] = -poly.points[i][0];
    }
    printf("Enter no of vertices of clipping window");
    sf("%d", &clipper.size);
    for(int i=0; i<clipper.size; i++) {
        printf("clip vertex");
        s = "%d %d", &clipper.points[i][0], &clipper.points[i][1];
        clipper.points[i][0] = clipper.points[i][1];
        clipper.points[i][1] = -clipper.points[i][0];
    }
}

```

chandras

Name of Experiment.....

Date.....

Experiment No. ....

Experiment Result.....

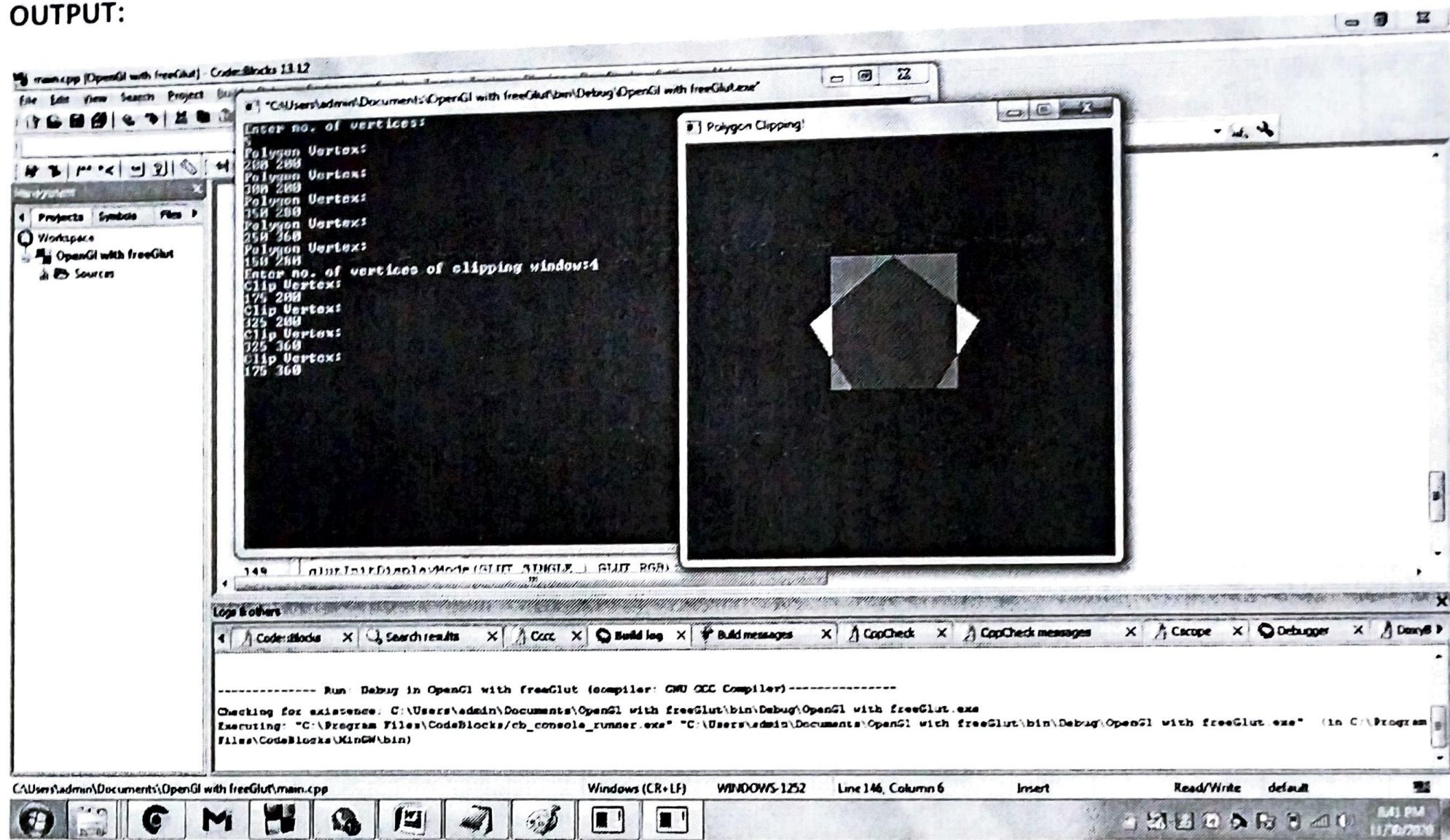
Page No. 38

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400, 400);
glutInitWindowPosition(100, 100);
glutCreateWindow("polygon Clipping");
glutDisplayFunc(display);
glutMainLoop();
return 0;
```

4

chandras

## OUTPUT:



## PROGRAM - 09

write a Program to model 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(){
    glGenLists(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();
    glEndList();
}
void wheelList(){
    glGenLists(WHEEL, GL_COMPILE_AND_EXECUTE);
    glColor3f(0,1,1);
    glutSolidSphere(10, 25, 25);
    glEndList();
}
```

Name of Experiment.....

Date.....

Page No. 33

Experiment No.....

Experiment Result.....

```
void myKeyboard(unsigned char key, int x, int y){  
    switch(key){  
        case 'e': glutPostRedisplay();  
        break;  
        case 'q': exit(0);  
        default: break;  
    }
```

4

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

3

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

4

```
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();  
}
```

5

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 34

```

void myDisp(){
    glClear(GL_COLOR_BUFFER_BIT);
    cosList();
    moveCos(s);
    wheelList();
}

```

4

```

void mouse(int btn, int state, int x, int y){
    if(btn == GLUT_LEFT_BUTTON & state == GLUT_DOWN){
        st = 5;
        myDisp();
    }
    else if(btn == GLUT_RIGHT_BUTTON & state == GLUT_DOWN){
        st = 2;
        myDisp();
    }
}

```

5

```

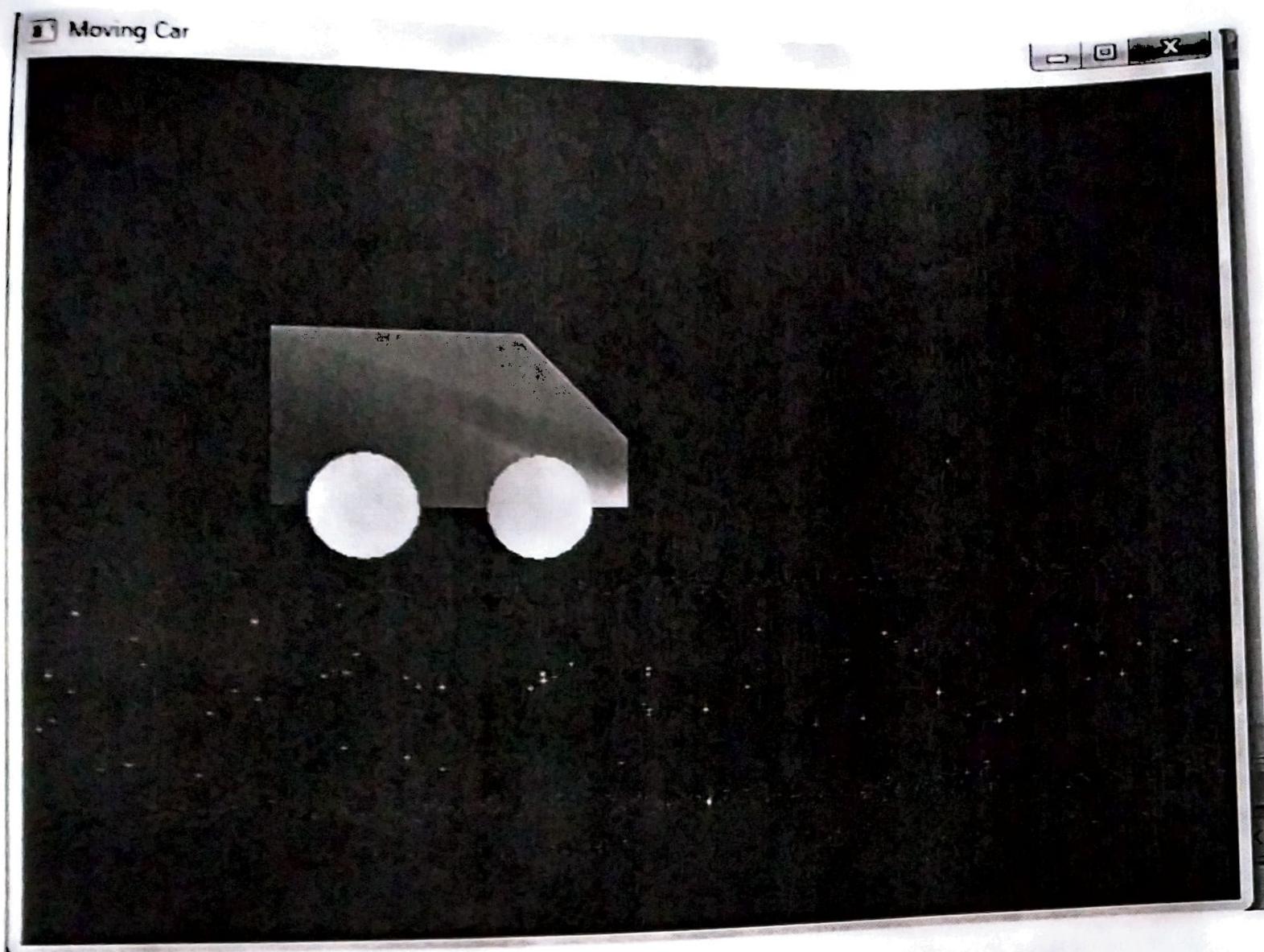
int main(int argc, char* argv[]){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(600, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("cos");
    myInit();
    glutDisplayFunc(myDisp);
    glutMouseFunc(mouse);
    glutKeyboardFunc(keyboard);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

```

6

chandras

## OUTPUT:



## 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/gle.h>
#include <GL/glut.h>
#include <ctime.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};

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, 1.0};

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

GLuint cubeIndices[] = {0, 3, 2, 1, 2, 3, 7, 0, 4, 7, 3, 1, 2, 6, 5, 4, 5, 6, 7, 0, 15, 4, 5};

static GLfloat theta[3] = {0.0, 0.0, 0.0};

static GLfloat beta[3] = {0.0, 0.0, 0.04};

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);
}
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 36

```

glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);
 glBegin(GL_LINES);
 glVertex3f(0.0, 0.0, 0.0);
 glVertex3f(1.0, 1.0, 1.0);
 glEnd();
 glFlush();

```

3

```

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

```

4

```

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;
}

```

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);

chandras

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

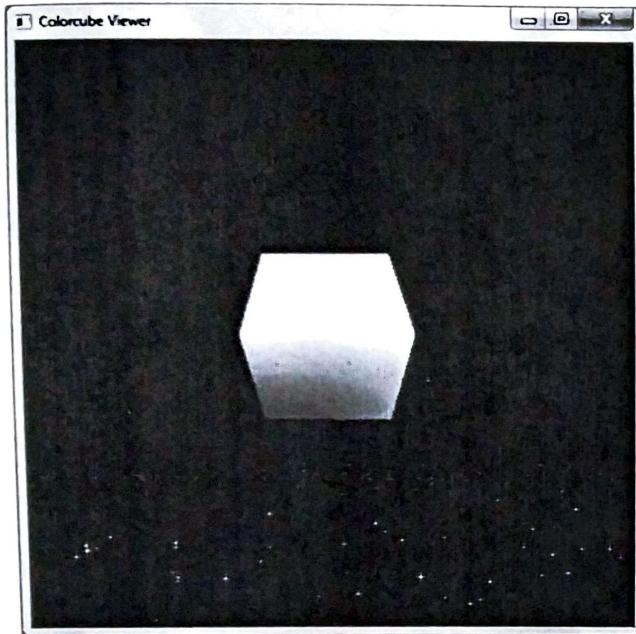
Page No. 37

```
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);
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glRasterPosClientState(GL_COLOR_ARRAY);
    glRasterPosClientState(GL_NORMAL_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();
}
```

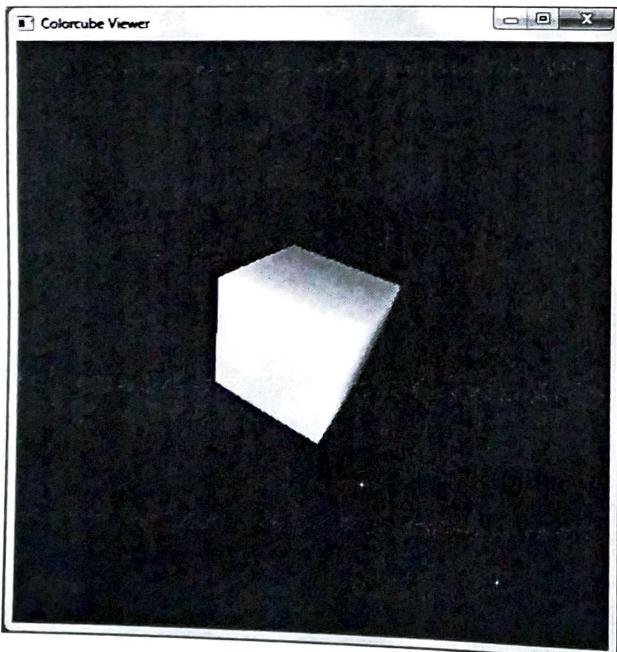
2

**Output:**

Rotation through X axis



Perspective view with Key 'x'



## Perspective view with Key 'y'



## 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, Cardioid, Three-leaf, and spiral. The color menu has sub menu with all eight colors of RGB color model. write programs to create the above hierarchical 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, cardioid=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(1.0, 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();
}
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 39

```

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;
    glClearColor(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;
    }
}

```

4

```

theta = dtheta;
while (theta < twoPi) {
    switch (curveNum) {
        case limacon: r = a * cos(theta) + b; break;
        case cardioid: 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);
}

```

4

```

curvePt[1].x = x0 + r * cos(theta);
curvePt[1].y = y0 + r * sin(theta);

```

chandra's

`lineSegment(CurvePt[0], CurvePt[1]);`

`CurvePt[0].x = CurvePt[1].x;`

`CurvePt[0].y = CurvePt[1].y;`

`theta += deltaTheta;`

?

?

`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 main-menu(int id){`

`switch(id){`

`case 3: exit(0);`

`default: break;`

4 4

chandra's

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 41

```

void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);
}

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(drawCurve);
    glutAddMenuEntry("Limaçon", 1);
    ("cardioid", 2);
    ("Threeleaf", 3);
    ("Spiral", 4);

    glutAttachMenu(GLUT_LEFT_BUTTON);
    int colorid = glutCreateMenu(colormenu);
    glutAddMenuEntry("Red", 4);
    glutAddMenuEntry("Green", 2);
    ("Blue", 1);
    ("Black", 0);
    ("Yellow", 6);
    ("Cyan", 3);
    ("Magenta", 5);
}

```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

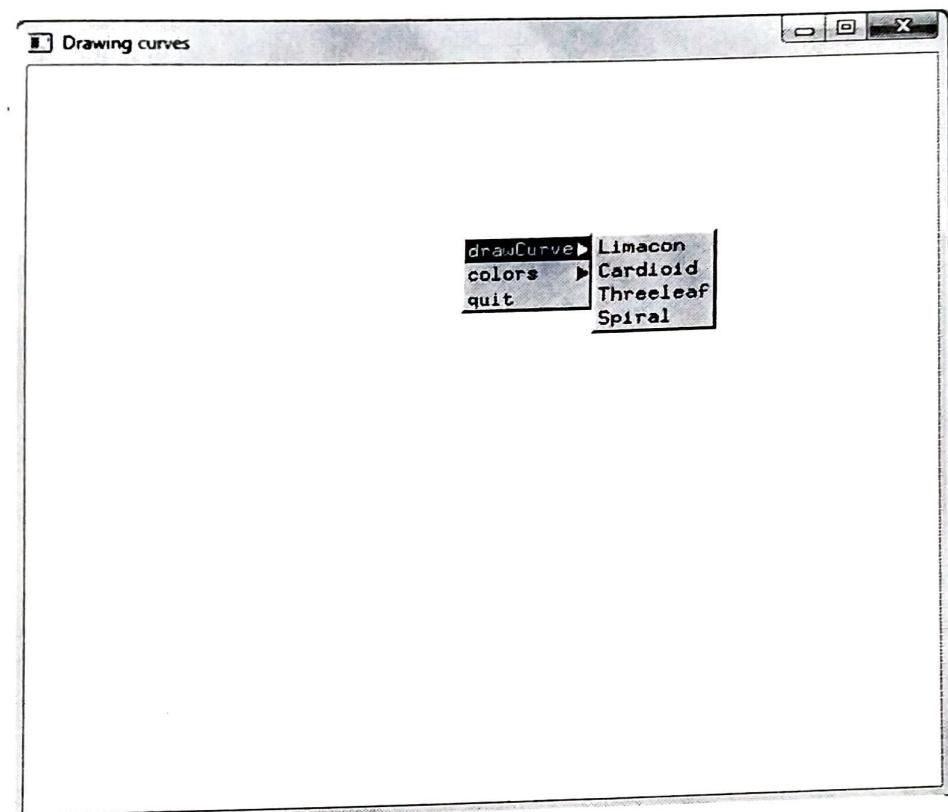
Page No. 42

```
glutAddmenuentry ("white", 7);  
glutAttachmenu (GLUT_LEFT_BUTTON);  
glutCreateMenu (main_menu);  
glutAddSubMenu ("drawCurve", curveid);  
glutAddSubMenu ("color", colorid);  
glutAddmenuentry ("quit", 3);  
glutAttachmenu(GLUT_LEFT_BUTTON);  
myinit ();  
glutDisplayFunc (mydisplay);  
glutReshapeFunc (myreshape);  
glutmainLoop();
```

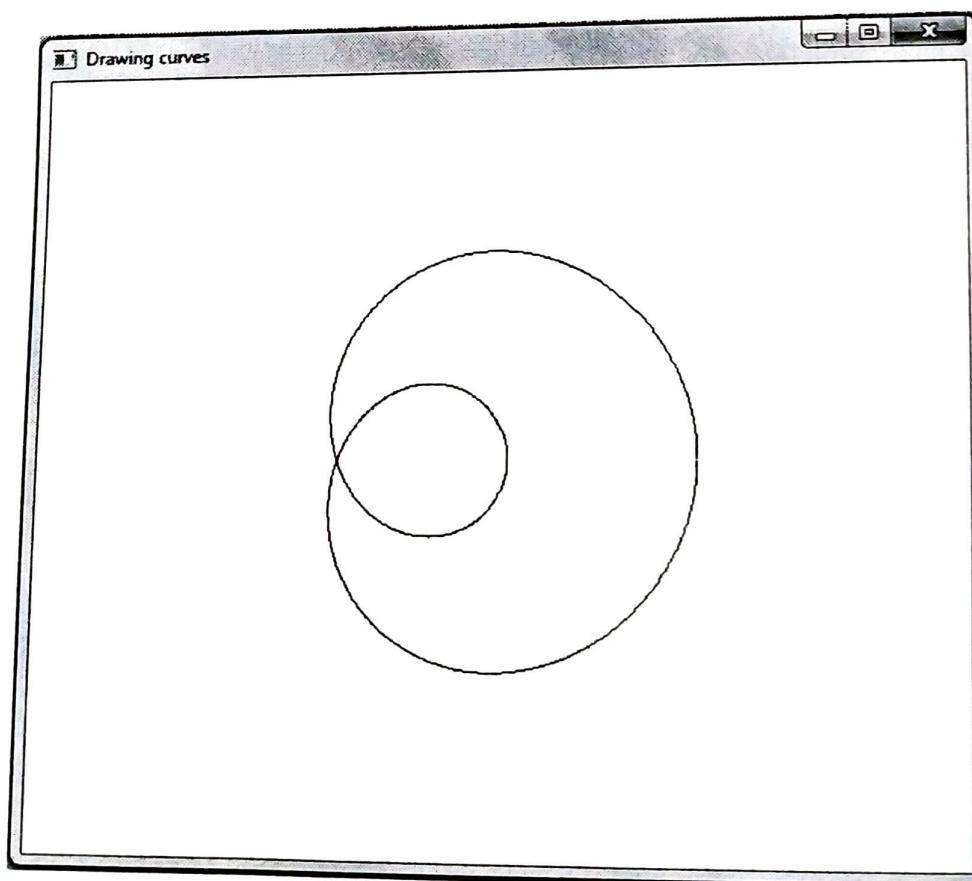
4

**Output:**

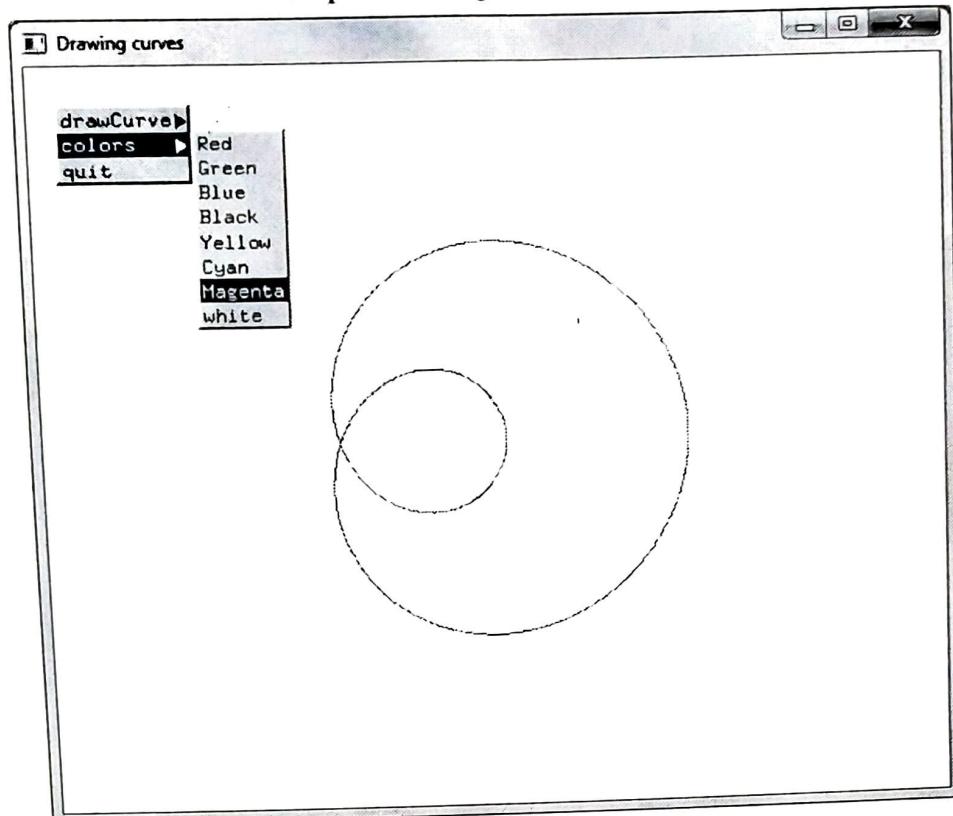
- Click left button show and select the menu.



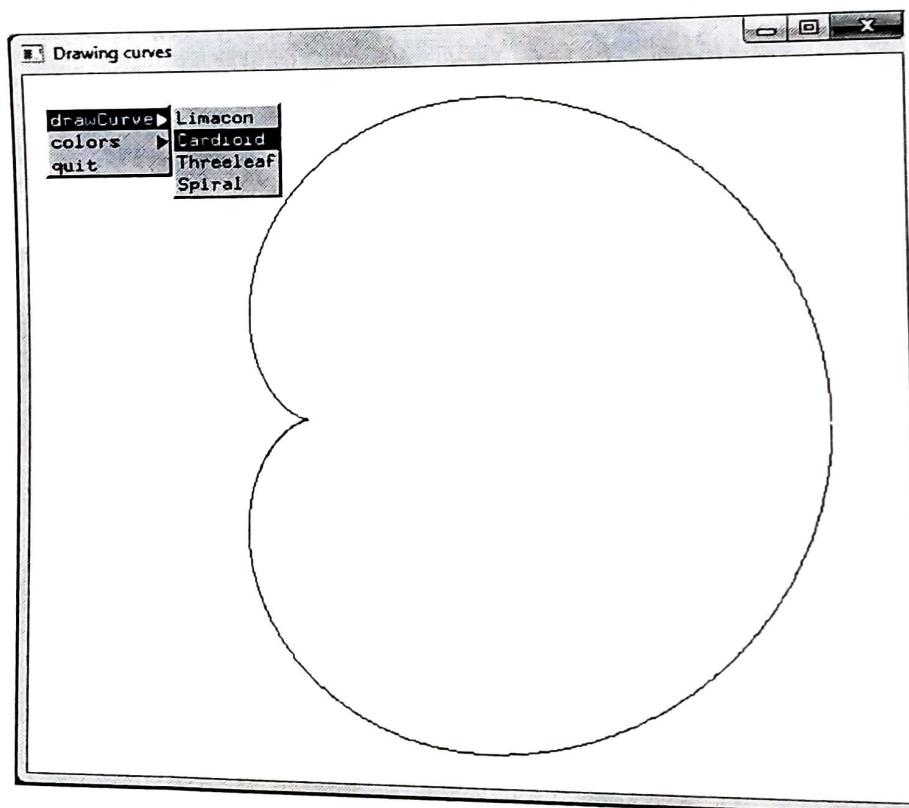
- Choose menu options to draw respective objects.

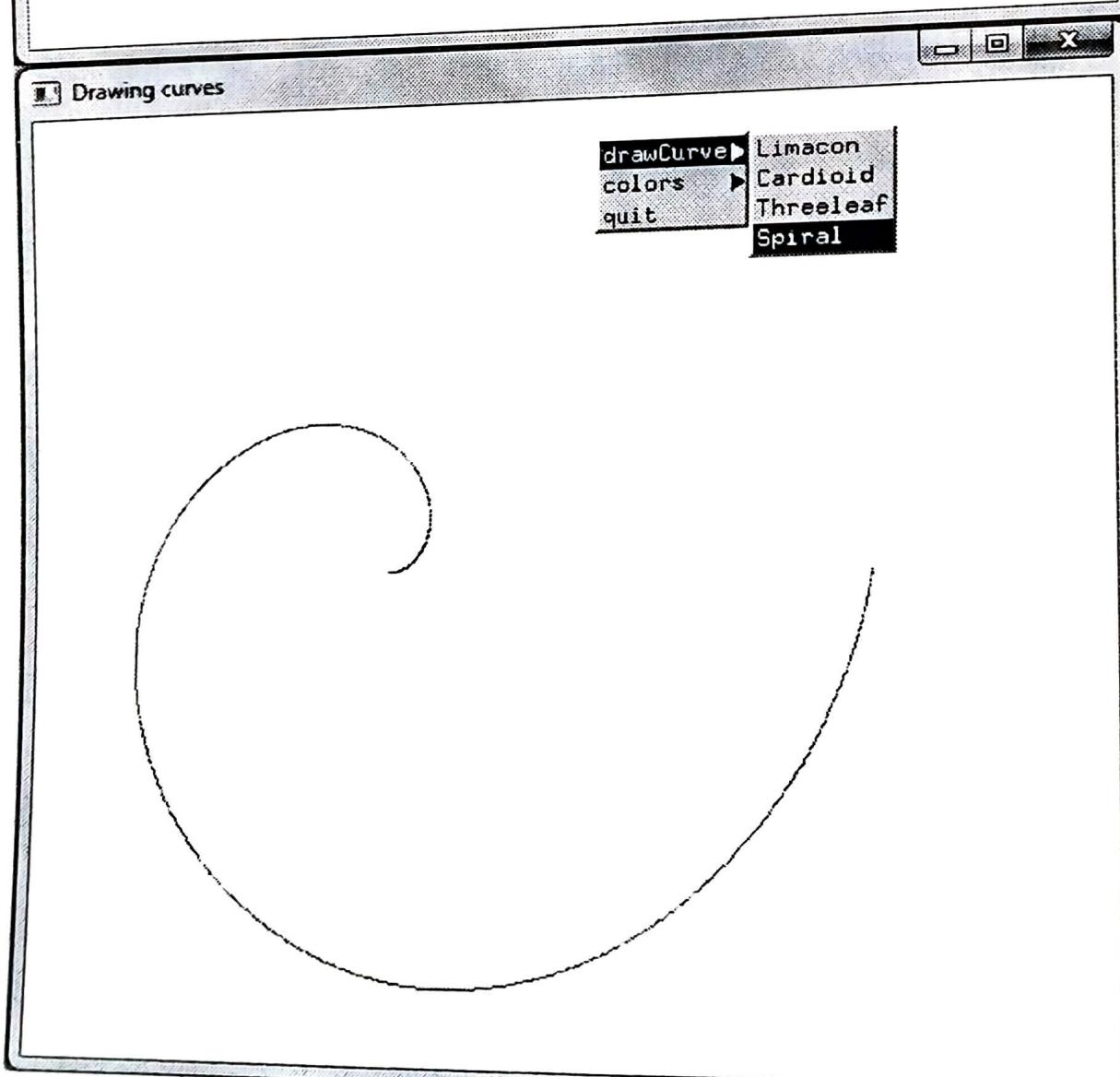
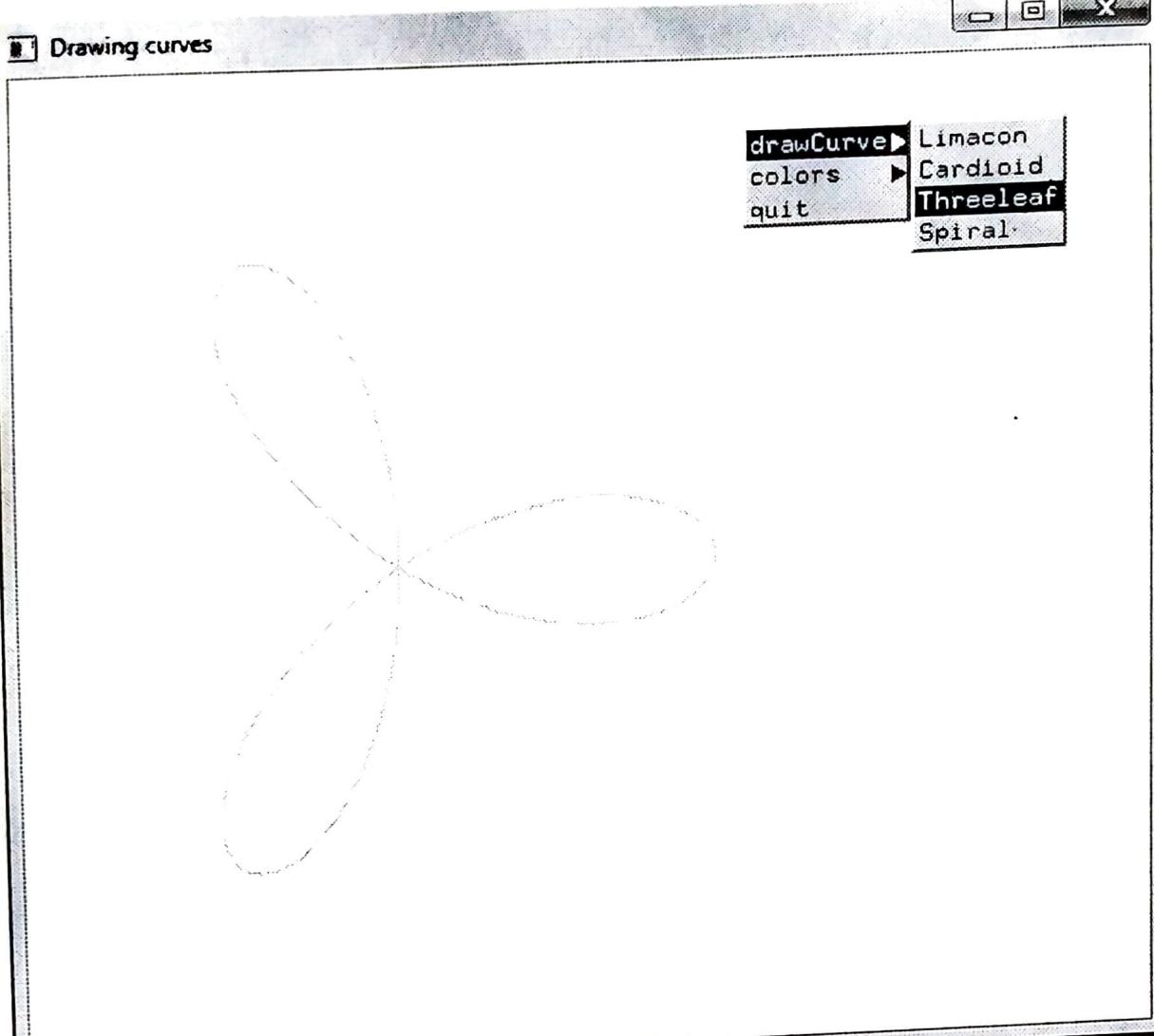


- Choose submenu ‘colors’ to draw respective objects in the selected color(within eight colors).

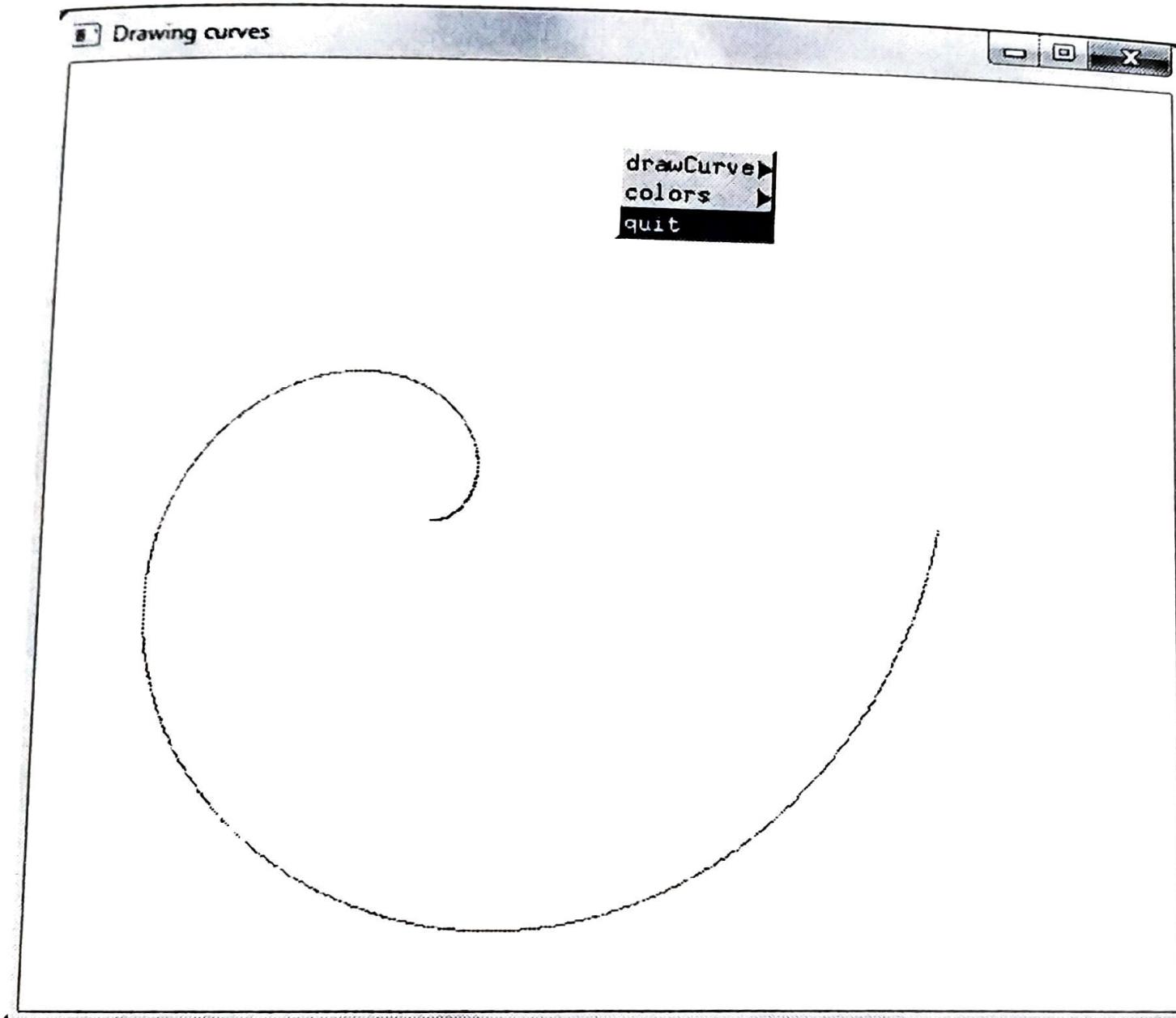


- Select limacon/cardiod/threeleaf/spiral to draw Respective object in selected color.





, select quit to exit.



## 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, x1[4], y1[4];
int flag=0;
void myInit(){
    glClearColor(1,1,1,1);
    glColor3f(1,1,1);
    glPointSize(5);
    glutOrtho2D(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){
```

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 44

```
double xt = pow(1-t, 3) * x[0] + 3*t * pow(1-t, 2) * x[1] + 3 * pow(t, 2) * (1-t) * x[2] + pow(t, 3) * x[3];
```

```
double yt = pow(1-t, 3) * y[0] + 3*t * pow(1-t, 2) * y[1] + pow(t, 2) * (1-t) * y[2] + pow(t, 3) * y[3];
```

```
glVertex2f(xt, yt);
```

4

```
	glColor3f(1, 1, 0);
```

```
for(i=0; i<4; i++) {
```

```
	glVertex2f(x[i], y[i]);
```

```
glEnd();
```

```
glFlush();
```

5

```
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;
```

```
cout << "x" << x << "y" << 500 - y;
```

```
glPointSize(3);
```

```
	glColor3f(1, 0, 0);
```

```
glBegin(GL_POINTS);
```

```
glVertex2f(x, 500, -y);
```

```
glEnd();
```

```
glFlush();
```

```
flag++;
```

6

```
if(flag >= 4 && btn == GLUT_LEFT_BUTTON)
```

```
{ glColor3f(0, 0, 1);
```

```
display();
```

```
flag = 0;
```

chandra's

Name of Experiment.....

Date.....

Experiment No.....

Experiment Result.....

Page No. 45

```
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutWindowPosition(0, 0);
    glutCreateWindow("BZ");
    glutDisplayFunc(display);
    glutMouseFunc(mouse); //include for mouse
    //mouse
}
```

use keyboard

```
cout << "Enter the x coordinate";
cin >> x1[0] >> x1[1] >> x1[2] >> x1[3];
cout << "Enter y coordinate";
cin >> y1[0] >> y1[1] >> y1[2] >> y1[3];
```

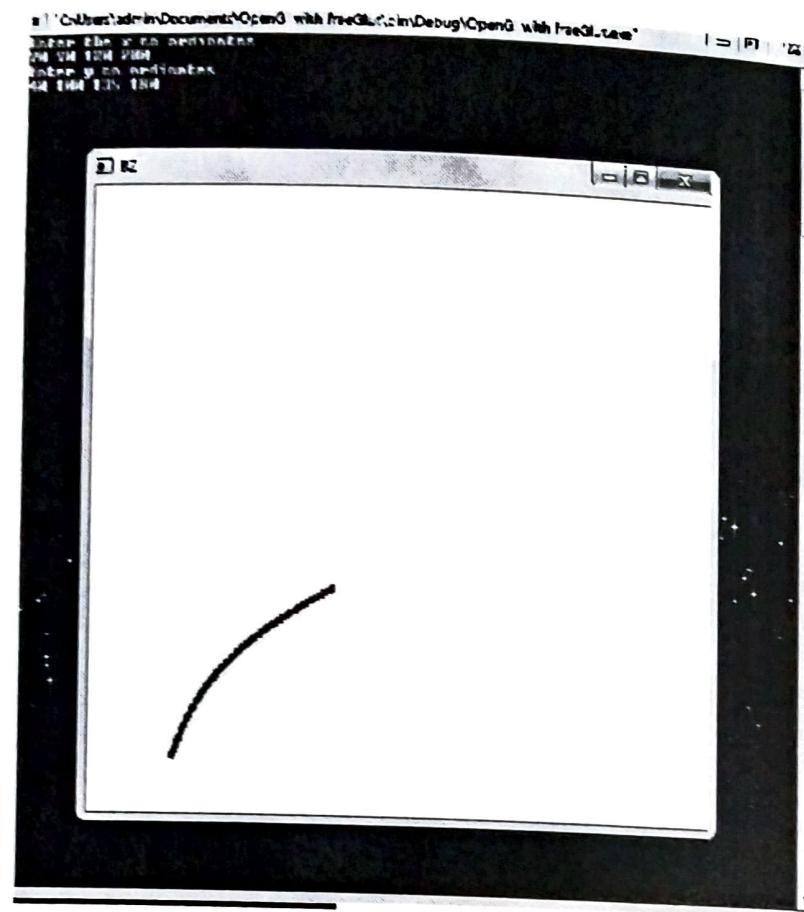
End keyboard

```
myInit();
glutMainLoop();
```

3

**Output:**

- Drawing BZ curve by inputting control points through keyboard.



- Drawing BZ curve through Mouse

