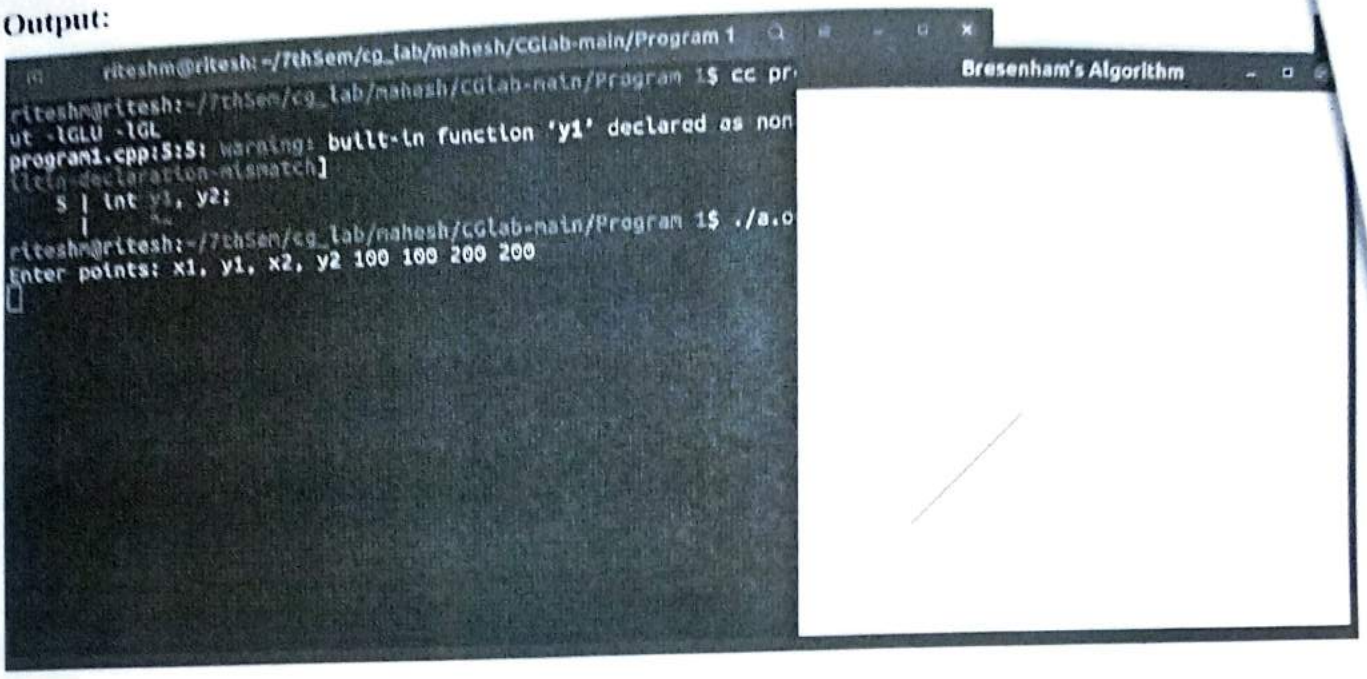
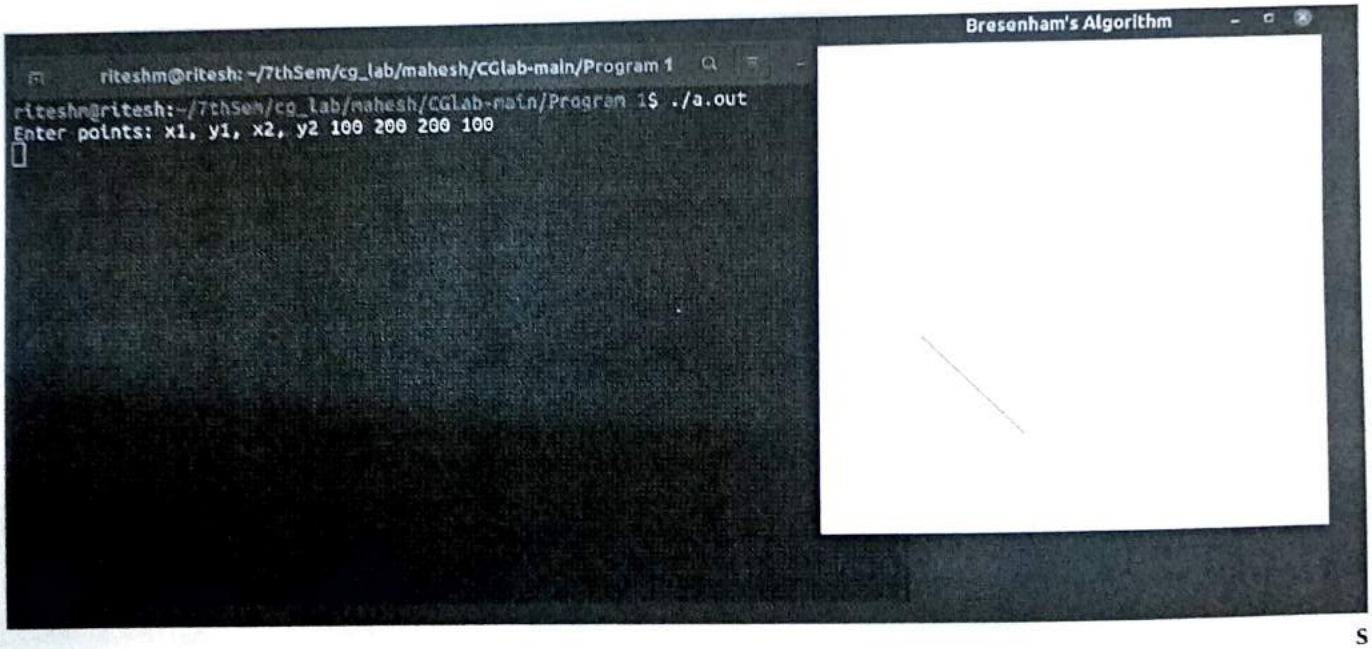
Output:


riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 1
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 1$ cc pr
ut -lGLU -lGL
program1.cpp:5:5: warning: built-in function 'y1' declared as non
[-Wbuiltin-declaration-mismatch]
5 | int y1, y2;
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 1$ ./a.o
Enter points: x1, y1, x2, y2 100 100 200 200
Bresenham's Algorithm


riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 1
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 1$ ./a.out
Enter points: x1, y1, x2, y2 100 200 200 100
Bresenham's Algorithm

S

Name of Experiment ... Bresenham's Algorithm ..... Date .....

Experiment No. ... 1 ..... Page No. ... 1 ...

1. Write a program to generate a line using Bresenham's Line drawing technique. Consider slopes greater than one and slopes less than one.

Code:

```
#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;
    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;
        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;
            }
            else e += inc2;
            y += incy;
            draw_pixel(x, y, BLACK);
        }
    }
}
```

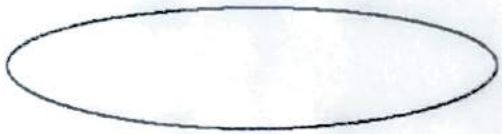
```
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    brez(x1, y1, x2, y2);
    glFlush();
}
void myinit() {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 499.0, 0.0, 499.0);
}
void main(int argc, char **argv) {
    printf("Enter points: x1, y1, x2, y2");
    scanf("%d%d%d%d", &x1, &x2, &y1, &y2);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Bresenhams Algorithm");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

```
riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 2
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ cc prc
ut -lGLU -lGL
/usr/bin/ld: /tmp/ccYZf5oo.o: undefined reference to symbol 'sqrtf
/usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding symbols
from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ cc prc
ut -lGLU -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ ./a.ou
Enter 1 to draw circle , 2 to draw ellipse
0
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ ./a.ou
Enter 1 to draw circle , 2 to draw ellipse
1
Enter coordinates of centre of circle and radius
0 0 50
```

Circle

```
riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 2
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ cc program2.cpp -lgl
ut -lGLU -lGL
/usr/bin/ld: /tmp/ccYZf5oo.o: undefined reference to symbol 'sqrtf@@GLIBC_2.2.5'
/usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing
from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ cc program2.cpp -lgl
ut -lGLU -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ ./a.out
Enter 1 to draw circle , 2 to draw ellipse
0
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ ./a.out
Enter 1 to draw circle , 2 to draw ellipse
1
Enter coordinates of centre of circle and radius
0 0 50
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 2$ ./a.out
Enter 1 to draw circle , 2 to draw ellipse
2
Enter coordinates of centre of ellipse and major and minor radius
0 0 200 50
```

Ellipse



Name of Experiment: Bresenham's Circle Date:

Experiment No: 2 Page No: 4

2. Write a program to generate a circle by bresenham's circle drawing technique. User can specify inputs through keyboard.

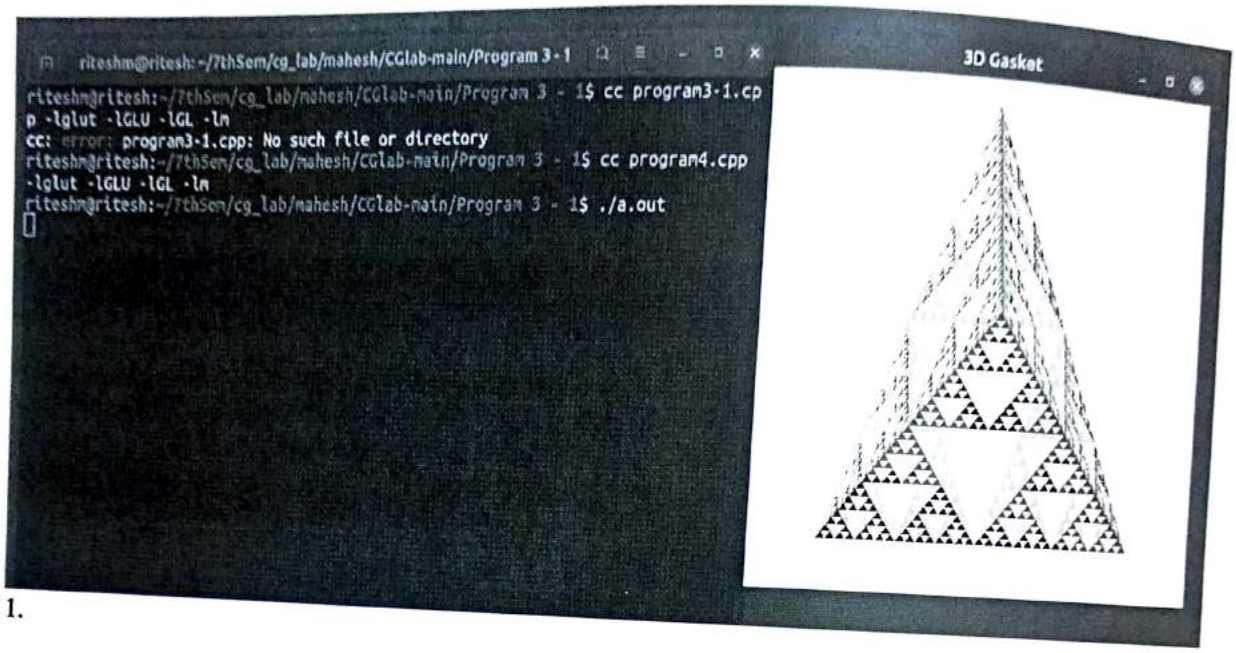
```
# include <stdio.h>
# include <gl/glut.h>
int xc, yc, r;
void drawCircle (int xc, int yc, int x, int y) {
    glBegin (GL_POINTS);
    glVertex2i(xc+x, yc+y);
    glVertex2i(xc-x, yc+y);
    glVertex2i(xc+x, yc-y);
    glVertex2i(xc-x, yc-y);
    glVertex2i(xc+y, yc+x);
    glVertex2i(xc-y, yc+x);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

Name of Experiment ..... Date .....

Experiment No..... Page No.....5.....

```
            d = d + 4*(x-y) + 10;
        }
        drawCircle(xc,yc,x,y);
    }
}
void display() {
    int j;
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    circleBres(xc,yc,r);
    glFlush();
}
void main(int argc, int **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(400,500);
    printf("Enter coord of centre of circle & radius:");
    scanf("%d %d %d", &xc, &yc, &r);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Bresenhams Circle");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

Output:


riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 3 - 1
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 3 - 1$ cc program3-1.cp
p -lglut -lGLU -lGL -lm
cc: error: program3-1.cpp: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 3 - 1$ cc program4.cpp
-lglut -lGLU -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 3 - 1$ ./a.out
3D Gasket

1.

Name of Experiment ..... Sierpinski Gasket ..... Date .....

Experiment No. ..... 3 ..... Page No. ..... 6 .....

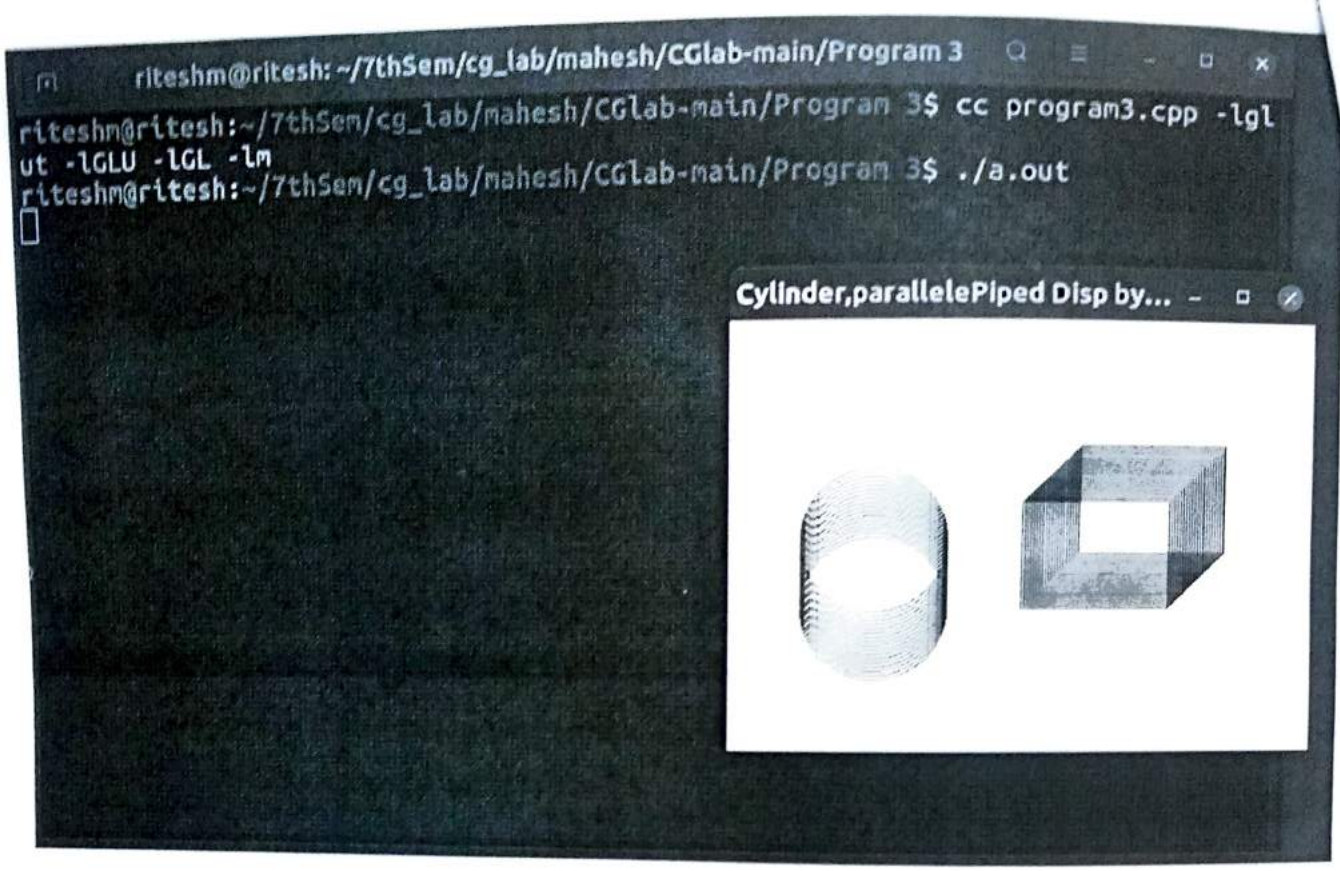
3. Write a program to recursively subdivides a tetrahedron to form 3D Sierpinski gasket. The number of recursive steps to be specified.

```
#include<GL/glut.h>
#include<stdio.h>
int m;
type 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() {
    printf("Enter no. of iterations");
    scanf("%d", &m);
    glutInit(&argv, argc);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowPosition(100,200);
    glutInitWindowSize(500,500);
    glutCreateWindow("Serpinski basket");
    glutDisplayFunc(tetrahedron);
    glEnable(GL_DEPTH_TEST);
    myinit();
    glutMainLoop();
}
void myinit() {
    glClearColor(1,1,1,1);
    glOrtho(-500.0,500.0,-500.0,500.0,-500.0,500.0);
}
```

```
void tetrahedron (void) {
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f (1.0, 0.0, 0.0);
    divide_triangle (tetra[0], tetra[1], tetra[2], m);
    glColor3f (0.0, 1.0, 0.0);
    divide_triangle (tetra[3], tetra[2], tetra[1], m);
    glColor3f (0.0, 0.0, 1.0);
    divide_triangle (tetra[0], tetra[3], tetra[1], m);
    glColor3f (0.0, 0.0, 0.0);
    divide_triangle (tetra[0], tetra[2], tetra[3], m);
    glFlush ();
}
void draw_triangle (point p1, point p2, point p3) {
    glBegin (GL_TRIANGLES);
    glVertex3fv (p1);
    glVertex3fv (p2);
    glVertex3fv (p3);
    glEnd ();
}
```

Output:

1.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 3
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 3$ cc program3.cpp -lgl
ut -lGLU -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 3$ ./a.out
Cylinder,parallelePiped Disp by...

Name of Experiment: Scan-line area fill | Date:

Experiment No.: 4 | Page No.: 8

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

```
#include <GL/glut.h>
#include <Windows.h>
#include <stdlib.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 int x[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;
    if (y2 < y1) {
        temp = x1; x1 = x2; x2 = temp;
        temp = y1; y1 = y2; y2 = temp;
    }
```

Name of Experiment .....

Experiment No. ..... Date .....

Page No. 9

```
        if (scanline > y1 && scanline < y2)
            int x[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) {
                drawline (intx[i], s1, intx[i+1], s1);
        }
}
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 ();
    scanfill (x, y);
}
void myInit() {
    glClearColor (1, 1, 1, 1);
    glColor3f (0, 0, 1);
    glPointSize (1);
    gluOrtho2D (0, wx, 0, wy);
}
```

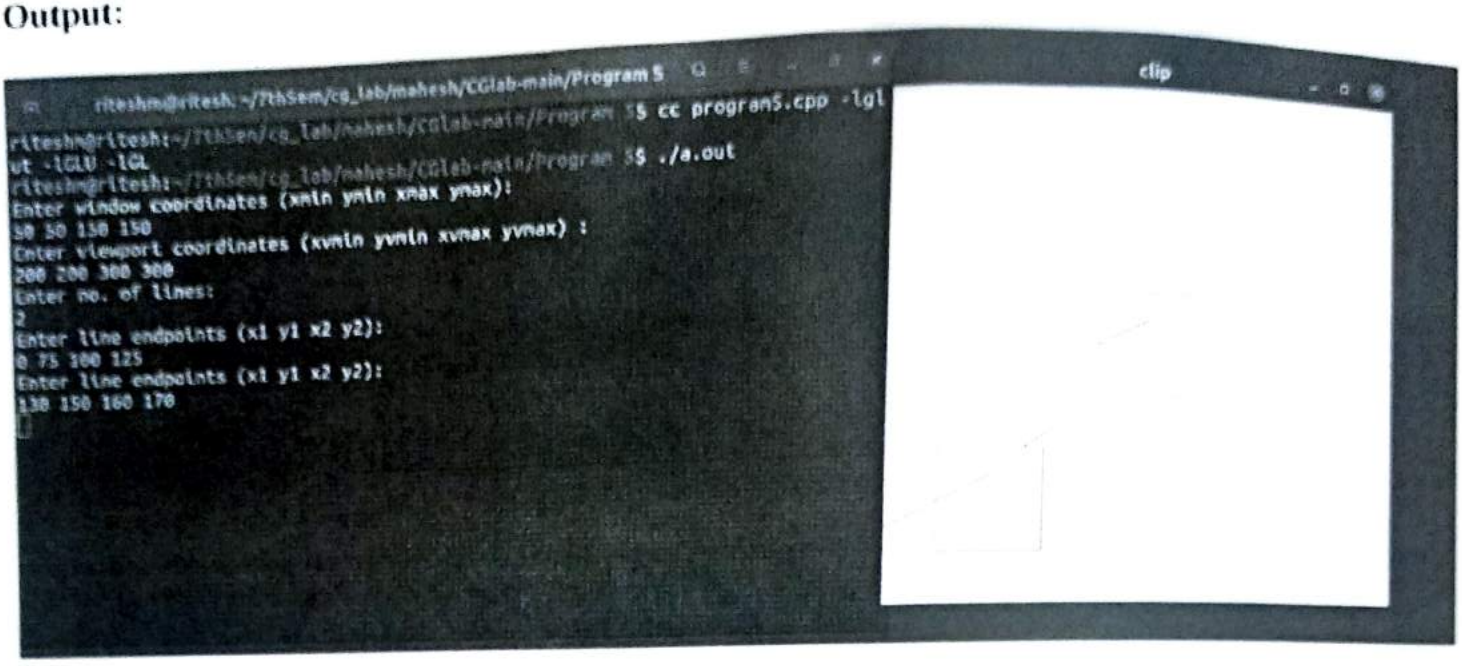
APPLE Office Solution

Name of Experiment ..... Date .....

Experiment No..... Page No. 10

```
int main (int ac, char * av[]) {
    glutInit (&ac, av);
    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]);
    }
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (0,0);
    glutCreateWindow ("scan line");
    glutDisplayFunc (display_filled_polygon);
    myinit ();
    glutMainLoop ();
}
```

Output:


riteshm@ritesh: ~/7thSem/cg_lab/mahesh/CGlab-main/Program 5
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 5$ cc program5.cpp -lgl
ut -lGLU -lGL
riteshm@ritesh:~/7thSem/cg_lab/mahesh/CGlab-main/Program 5$ ./a.out
Enter window coordinates (xmin ymin xmax ymax):
50 50 150 150
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
200 200 300 300
Enter no. of lines:
2
Enter line endpoints (x1 y1 x2 y2):
0 75 100 125
Enter line endpoints (x1 y1 x2 y2):
130 150 160 170
clip

1.

Name of Experiment ..... Cohen - Sutherland ..... Date .....

Experiment No. ..... 05 ..... Page No. .....

05. Write a program to implement the Cohen-Sutherland line clipping algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
#define outcode int
#define true 1
#define false 0
double xmin, ymin, xmax, ymax;
double xvmin, xyvmin, xvmax, yvmin;
const int RIGHT = 4;
const int LEFT = 8;
const int TOP = 1;
const int BOTTOM = 2;
int n;
struct line_segment {
    int x1;
    int y1;
    int x2;
    int y2;
};
struct line_segment ls[10];

outcode computeoutcode (double x, double y) {
    outcode code = 0;
    if (y > ymax)
        code |= TOP;
```

Name of Experiment ..... Date .....

Experiment No..... Page No.....

```
else if(y<ymin)
    code |= BOTTOM;
if(x>xmax)
    code |= RIGHT;
else if(x<xmin)
    code |= LEFT;
return code;
}
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;
            outcode out = outcode0 ? outcode0 : outcode1;
            if (outcodeout & TOP) {
                x = x0 + (x1-x0) * (ymax-y0)/(y1-y0);
                y = ymin;
            }
            else if (outcodeout & RIGHT) {
                y = y0 + (y1-y0) * (xmax-x0)/(x1-x0);
```

Name of Experiment ..... Date .....

Experiment No. ..... Page No. .....

```
        * xmax.
    }
    else {
        y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
        x = xmin;
    }
    if (outcode0 == outcode1) {
        x0 = x;
        y0 = y;
        outcode0 = computeOutCode(x0, y0);
    }
    else {
        x1 = x;
        y1 = y;
        outcode1 = computeOutCode(x1, y1);
    }
}
} while (!done);
if (accept) {
    double sx = (xvmax - xvmin) / (xmax - xmin);
    double sy = (yvmax - yvmin) / (ymax - ymin);
    double vx0 = xvmin + (x0 - xmin) * sx;
    double vy0 = yvmin + (y0 - ymin) * sy;
    double vx1 = xvmin + (x1 - xmin) * sx;
    double vy1 = yvmin + (y1 - ymin) * sy;

    glColor3f(1.0, 0.0);
    glBegin(GL_LINE_LOOP);
```

Name of Experiment ..... Date .....

Experiment No..... Page No.....

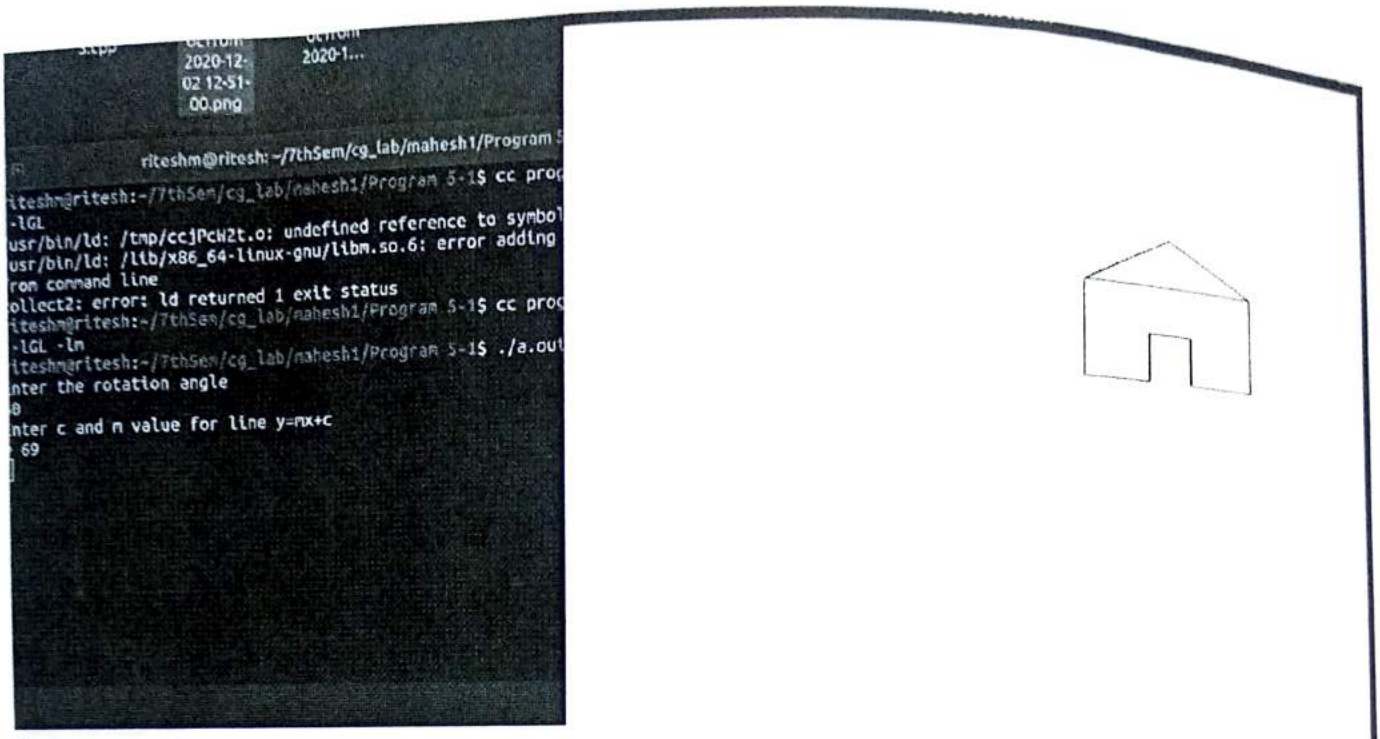
```
        glVertex2f(xvmin,yvmin);
        glVertex2f(xvmax,yvmin);
        glVertex2f(xvmax,yvmax);
        glVertex2f(xvmin,yvmax);
        glEnd();
        glColor3f(0,0,1);
        glBegin(GL_LINES);
        glVertex2d(vx0,vy0);
        glVertex2d(vx1,vy1);
        glEnd();
    }
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0,0,1);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xmin,ymin);
    glVertex2f(xmax,ymin);
    glVertex2f(xmax,ymax);
    glVertex2f(xmin,ymax);
    glEnd();
    for(int i=0;i<n;i++){
        glBegin(GL_LINES);
        glVertex2d(ls[i].x1,ls[i].y1);
        glVertex2d(ls[i].x2,ls[i].y2);
        glEnd();
    }
    for(int i=0;i<n;i++)
        cohensuther(ls[i].x1,ls[i].y1,ls[i].x2,ls[i].y2);
```

Name of Experiment ..... Date .....

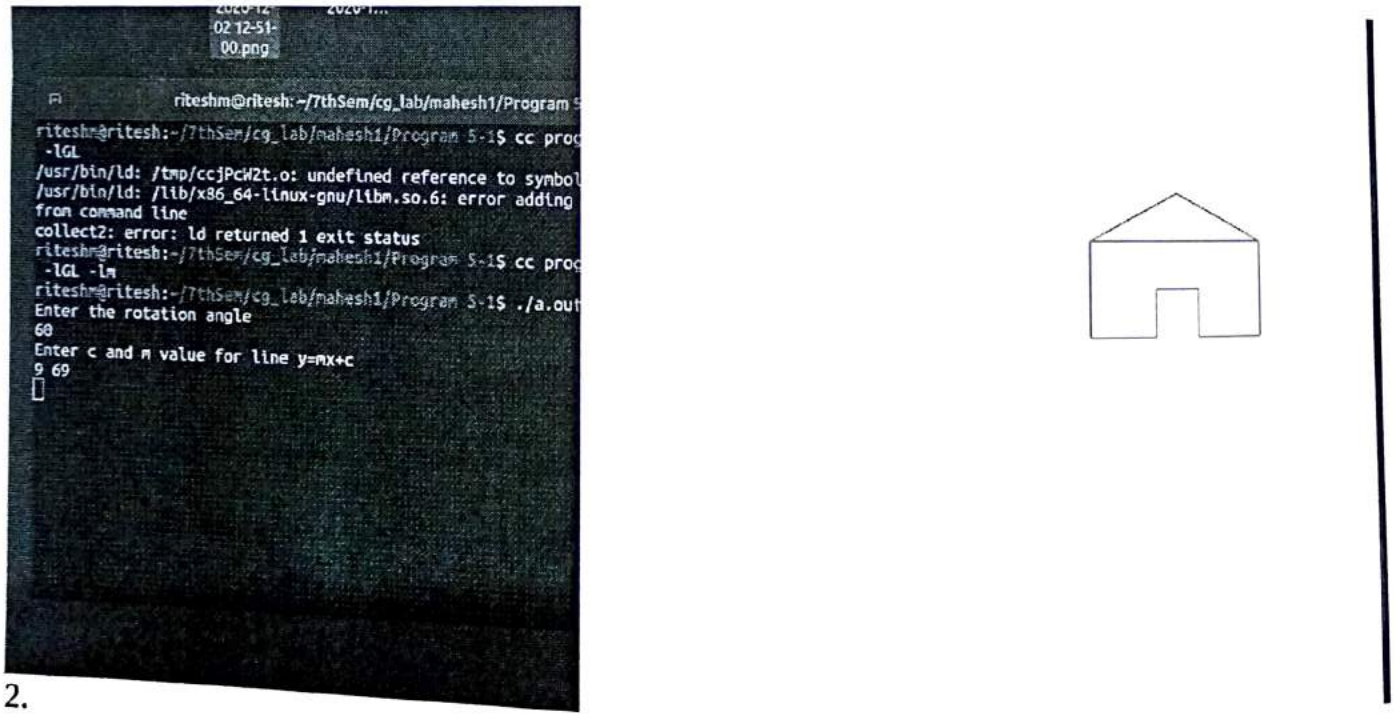
Experiment No..... Page No.....

```
    glFlush();
}
void myinit() {
    glClearColor(1.0,1.0,1.0);
    glColor3f(1.0,0.0,0.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,500.0,0.0,500.0);
}
int main(int argc, char *argv[]){
    printf("Enter window coordinates (xmin ymin xmax ymax):\n");
    scanf("%lf %lf %lf %lf", &xmin, &ymin, &xmax, &ymax);
    printf("Enter viewport coordinates (xvmin, yvmin, xvmax, yvmax):\n");
    scanf("%lf %lf %lf %lf", &xvmin, &yvmin, &xvmax, &yvmax);
    printf("Enter no of lines:\n");
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        printf("Enter line endpoints (x1 y1 x2 y2):\n");
        scanf("%d %d %d %d", &ls[i].x1, &ls[i].y1, &ls[i].x2, &ls[i].y2);
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(200,200);
    glutCreateWindow("clip");
    myinit();
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Output:


2020-12-02 12-51-00.png
2020-1...
riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 5
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 5-1$ cc prog
-lGL
usr/bin/ld: /tmp/ccjPcW2t.o: undefined reference to symbol
usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding
rom command line
ollect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 5-1$ cc prog
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 5-1$ ./a.out
nter the rotation angle
0
nter c and m value for line y=mx+c
69

1.


02 12-51-00.png
riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 5
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 5-1$ cc prog
-lGL
/usr/bin/ld: /tmp/ccjPcW2t.o: undefined reference to symbol
/usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding
from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 5-1$ cc prog
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 5-1$ ./a.out
Enter the rotation angle
60
Enter c and m value for line y=mx+c
9 69

2.

6 Write a program to create a house like figure and perform:
(a) Rotate it about a given fixed point using OpenGL transformation function
(b) Reflect it about an axis $y = mx + c$ using OpenGL transformation.

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

float house[11][2] = {{100,200},{200,250},{300,200},{100,200},{100,100},{175,150},
{175,150},{225,100},{300,100},{100,200}};
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();
    glFlush();
    glPushMatrix();
    glTranslatef(100,100,0);
```

Name of Experiment ..... Date .....

Experiment No..... Page No.....

```
        glRotatef(angle,0,0,1);
        glTranslatef(-100,-100,0);
        glColor3f(1,1,0);
        glBegin(GL_LINE_LOOP);
        for(int i=0;i<110;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();
        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);
```

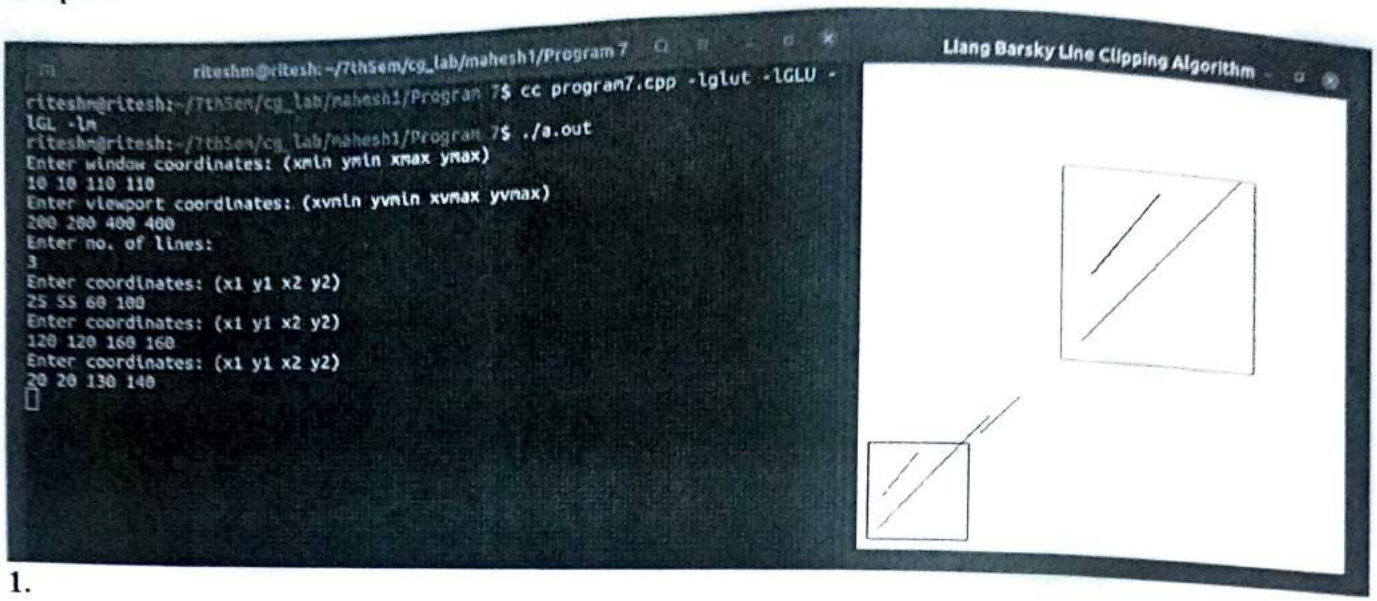
```
glBegin(GL_LINES);
glVertex2f(x1,y1);
glVertex2f(x2,y2);
glEnd();
glFlush();
glPushMatrix();
glTranslate(0,c,0);
theta = atan(m);
theta = theta * 180/3.14;
glRotatef(theta,0,0,1);
glScalef(1,-1,1);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++)
        glVertex2fv(house[i]);
glEnd();
glPopMatrix();
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();
glOrtho2D(-450,450,-450,450);
}
void mouse(int btn, int state, int x, int y){
    if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN){
```

Experiment ..... Date .....

Int No..... Page No.....

```
        display1();
    }
    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);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(900,900);
    glutInitWindowPosition(100,100);
    glutCreateWindow("House Rotation");
    glutMouseFunc(mouse);
    glutDisplayFunc(display);
```

Output:


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 7
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 7$ cc program7.cpp -lglut -lGLU -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 7$ ./a.out
Enter window coordinates: (xmin ymin xmax ymax)
10 10 110 110
Enter viewport coordinates: (xvmin yvmin xvmax yvmax)
200 200 400 400
Enter no. of lines:
3
Enter coordinates: (x1 y1 x2 y2)
25 55 60 100
Enter coordinates: (x1 y1 x2 y2)
120 120 160 160
Enter coordinates: (x1 y1 x2 y2)
20 20 130 140
Liang Barsky Line Clipping Algorithm

1.

e of Experiment ..... Liang - Barsky ..... Date .....

riment No..... 07 ..... Page No.....

Write a program to implement Liang-Barsky line clipping algorithm
Make provision to specify multiple input lines, 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<0.0) {
        if(r<*u2) *u2=r;
        if(r<*u1) return(false);
    }
    else if (p>0.0) {
        if(r<*u2) *u2=r;
        if(r<*u1) return(false);
    }
    else if (p==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, u1=0.0, u2=1.0;
```

Name of Experiment ..... Date .....

Experiment No..... Page No.....

```
glColor3f(1.0,0.0,0.0);
glBegin(GL_LINE_LOOP);
    glVertex2f(xvmin, yvmin);
    glVertex2f(xvmax, yvmin);
    glVertex2f(xvmax, yvmax);
    glVertex2f(xvmin, yvmax);
glEnd();
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<1.0) {
            x1=x0+u2*dx;
            y1=y0+u2*dy;
        }
        if(u1>0.0) {
            x0=x0+u1*dx;
            y0=y0+u1*dy;
        }
        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(0.0,0.0,1.0);
        glBegin(GL_LINES);
            glVertex2d(vx0, vy0);
            glVertex2d(vx1, vy1);
```

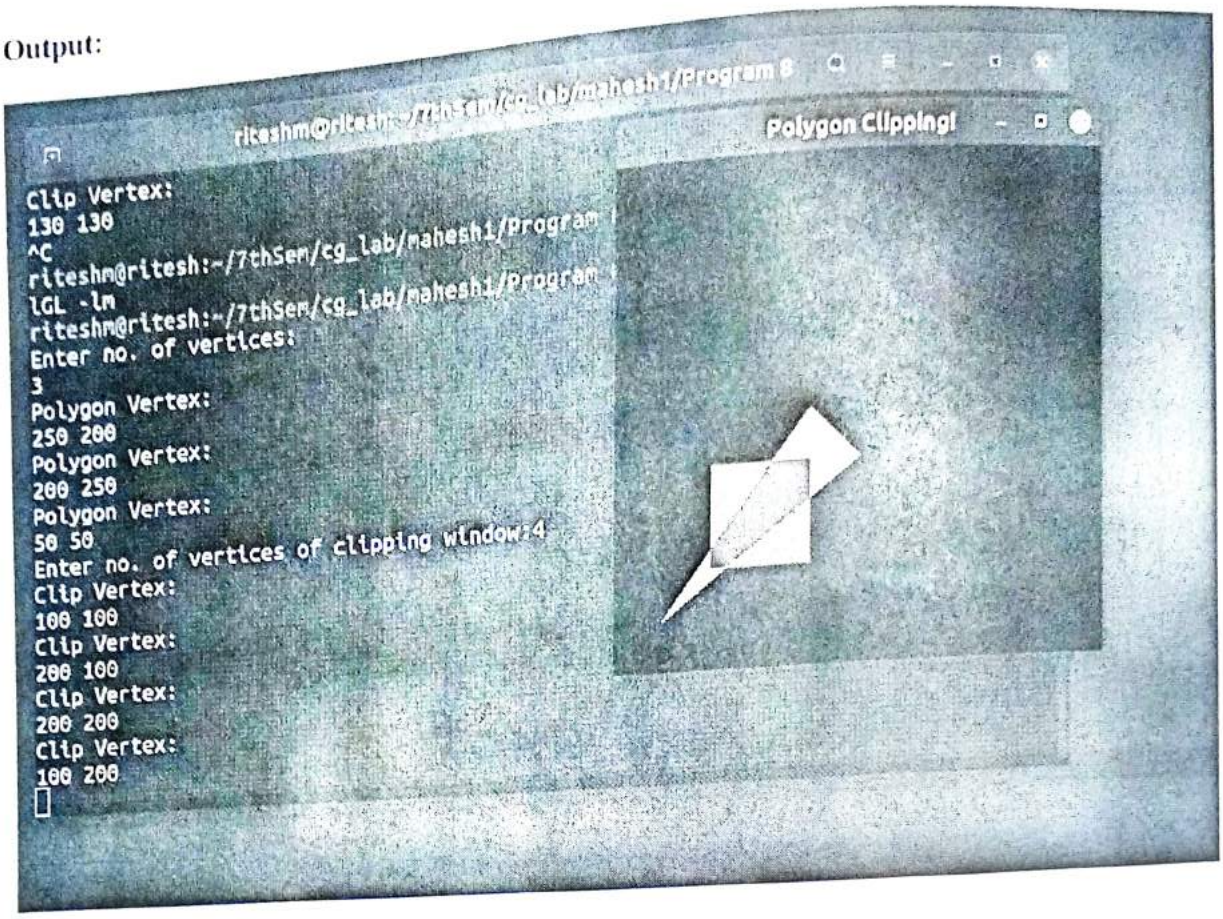
```
        glEnd();
    }
}
void display() {
    double x0=60, y0=60, x1=90, y1=90,
    double p0=110, q0=30, p1=30, q1=120;
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0,0.0,0.0);
    glBegin(GL_LINES);
        glVertex2d(x0,y0);
        glVertex2d(x1,y1);
        glVertex2d(p0,q0);
        glVertex2d(p1,q1);
    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);
    LiangBarskyLineClipAndDraw(p0,q0,p1,q1);
    glFlush();
}
void myinit() {
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f(1.0,0.0,0.0);
```

Name of Experiment ..... Date .....

Experiment No..... Page No.....

```
    glLineWidth(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 499.0, 0.0, 499.0);
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Liang Barsky Line Clipping");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

Output:


Clip Vertex:
130 130
^C
riteshm@ritesh:~/7thSem/cg_lab/maheshi/Program
lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/maheshi/Program
Enter no. of vertices:
3
Polygon Vertex:
250 200
Polygon Vertex:
200 250
Polygon Vertex:
50 50
Enter no. of vertices of clipping window:4
Clip Vertex:
100 100
Clip Vertex:
200 100
Clip Vertex:
200 200
Clip Vertex:
100 200
Polygon Clipping!

1.

Name of Experiment Poly Clip Algorithm Date

Experiment No. 08 Page No.

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

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
int poly_size, poly_points[20][2], org_poly_size, org_poly_points[20][2], clipper_size,
    clipper_points[20][2];
const int MAX_POINTS=9;

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;
}

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;
```

Experiment No.....

```
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_intersect(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;
```

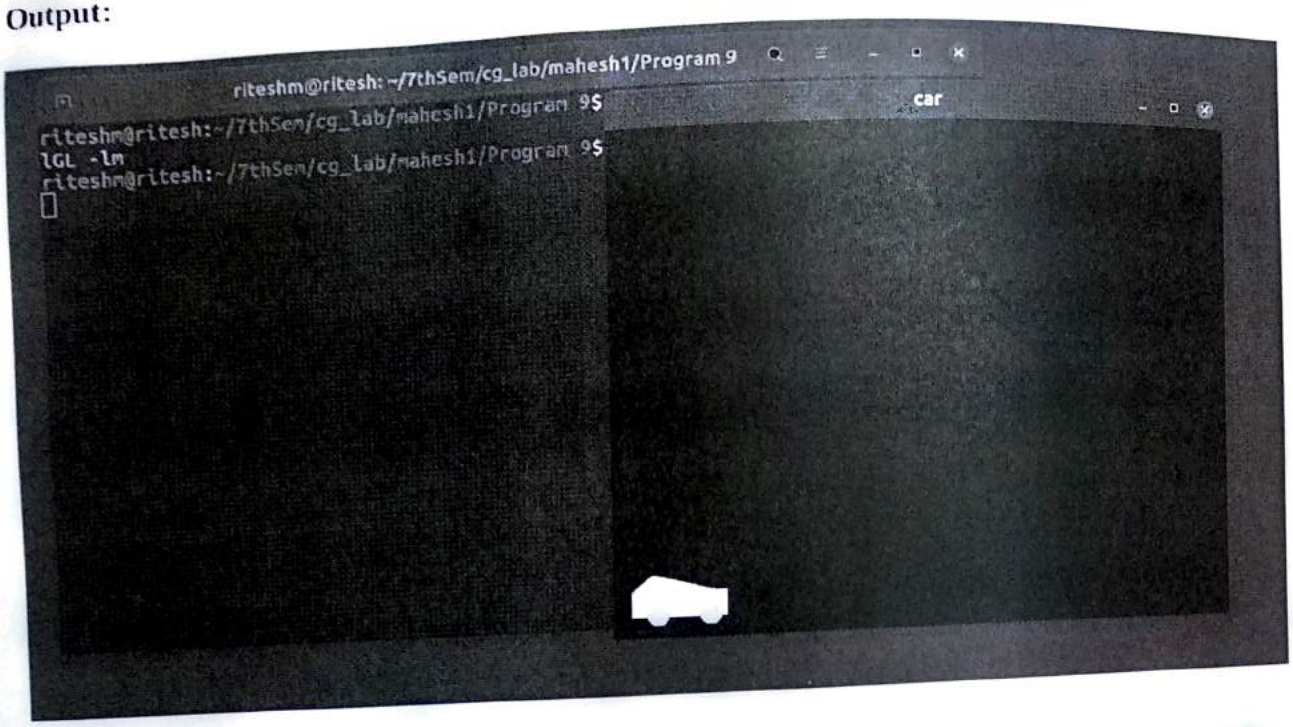
Name of Experiment ..... Date .....

Experiment No. ..... Page No. .....

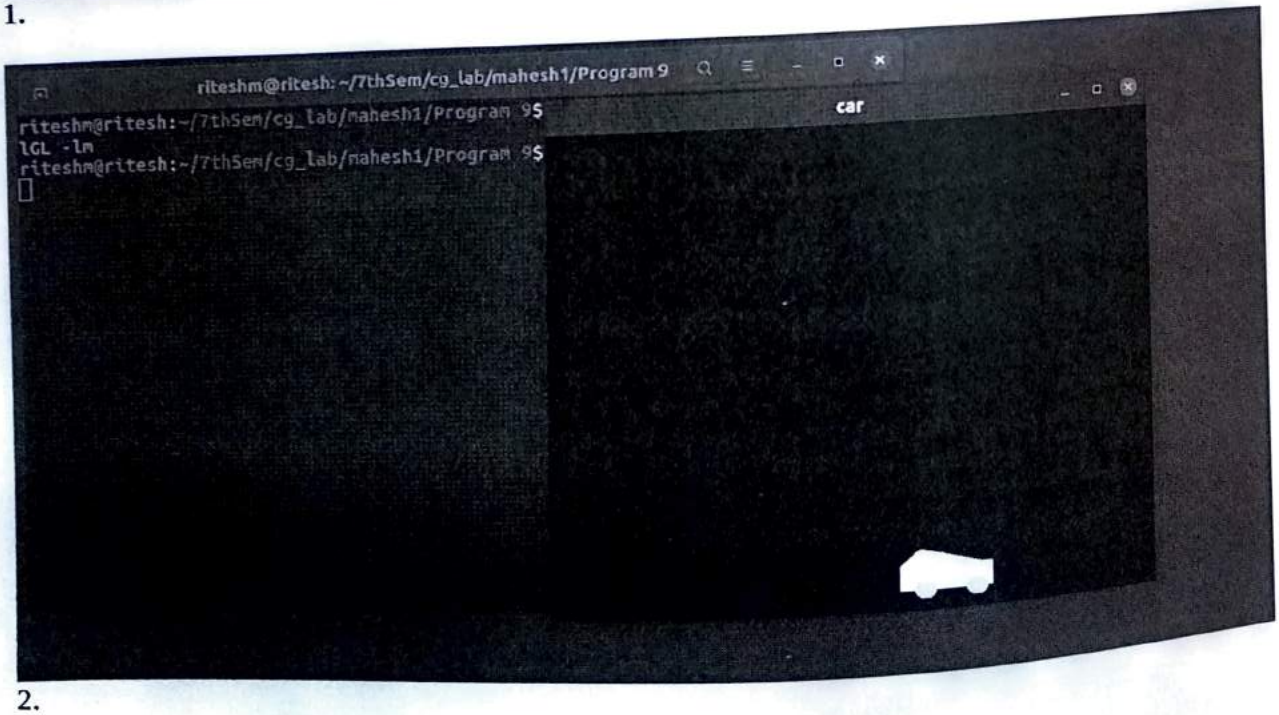
```
        poly_size = new_poly_size;
        for (int i=0; i< poly_size; i++) {
            poly_points[i][0] = new_points[i][0];
            poly_points[i][1] = new_points[i][1];
        }
    }
void init() {
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho2D(0.0, 500.0, 0.0, 500.0, 0.0, 500.0);
    glClear(GL_COLOR_BUFFER_BIT);
}
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;
```

```
int main(int argc, char * argv[]) {
    printf("Enter no. of vertices:\n");
    scanf_s("%d", &poly_size);
    org_poly_size = poly_size;
    for(int i=0; i<poly_size; i++){
        printf("Polygon Vertex\n");
        scanf_s("%d %d", &poly_points[i][0], &poly_points[i][1]);
        org_poly_points[i][0] = poly_points[i][0];
        org_poly_points[i][1] = poly_points[i][1];
    }
    printf("Enter no. of vertices of clipping window:");
    scanf_s("%d", &clippersize);
    for(int i=0; i<clipper_size; i++){
        printf("Clip Vertex:\n");
        scanf_s("%d %d", &clipper_points[i][0], &clipper_points[i][1]);
    }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400, 400);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Polygon Clipping");
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Output:


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 9
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 9$
lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 9$
car

1.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 9
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 9$
lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 9$
car

2.

Name of Experiment ..... car ..... Date .....

Experiment No. ..... 09 ..... Page No. .....

09. Write a program to model a car using display lists.

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

Name of Experiment .....

Experiment No..... Date .....

Page No.....

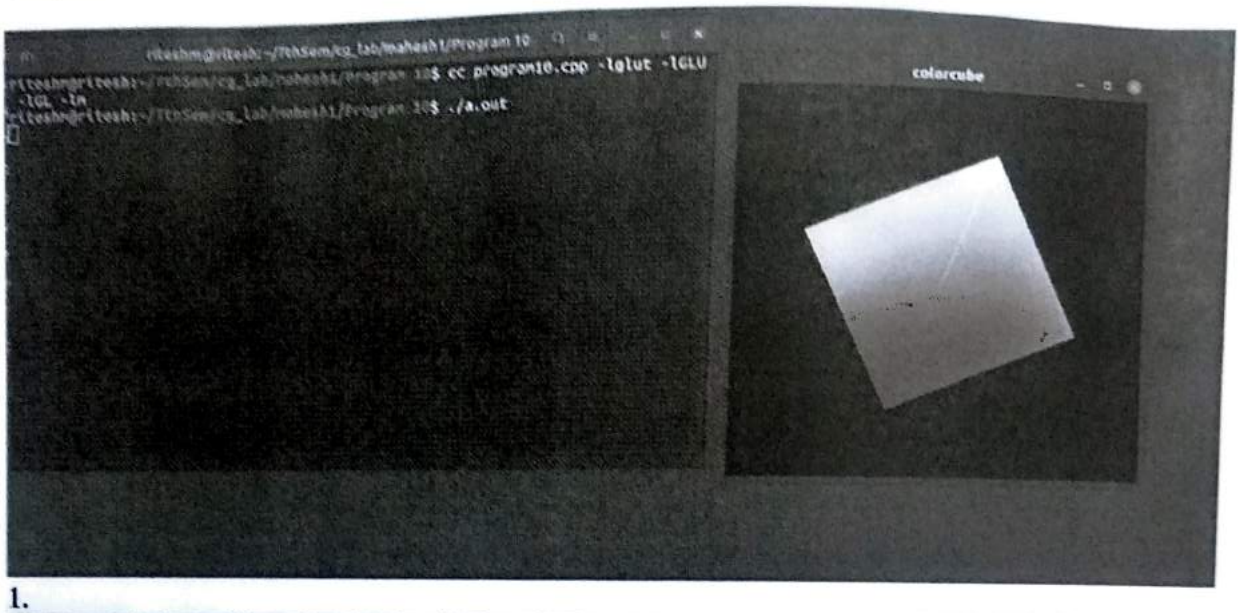
```
void mykeyboard (unsigned char key, int x, int y) {
    switch (key) {
        case 'f': glutPostRedisplay();
                  break;
        case 'q': exit(0);
        default : break;
    }
}
void myInit() {
    glClearColor(0,0,0,0);
    glOrtho(0,600,0,600,0,600);
}
void draw_wheel() {
    glColor3f(0,1,1);
    glutSolidSphere(10,25,25);
}
void movecar(float s) {
    glTranslate(s,0.0,0.0);
    glCallList(CAR);
    glPushMatrix();
    glTranslate(25,21,0.0);
    glCallList(WHEEL);
    glPopMatrix();
    glPushMatrix();
    glTranslate(75,25,0.0);
    glCallList(WHEEL);
    glPopMatrix();
    glFlush();
}
```

Name of Experiment ..... Date .....

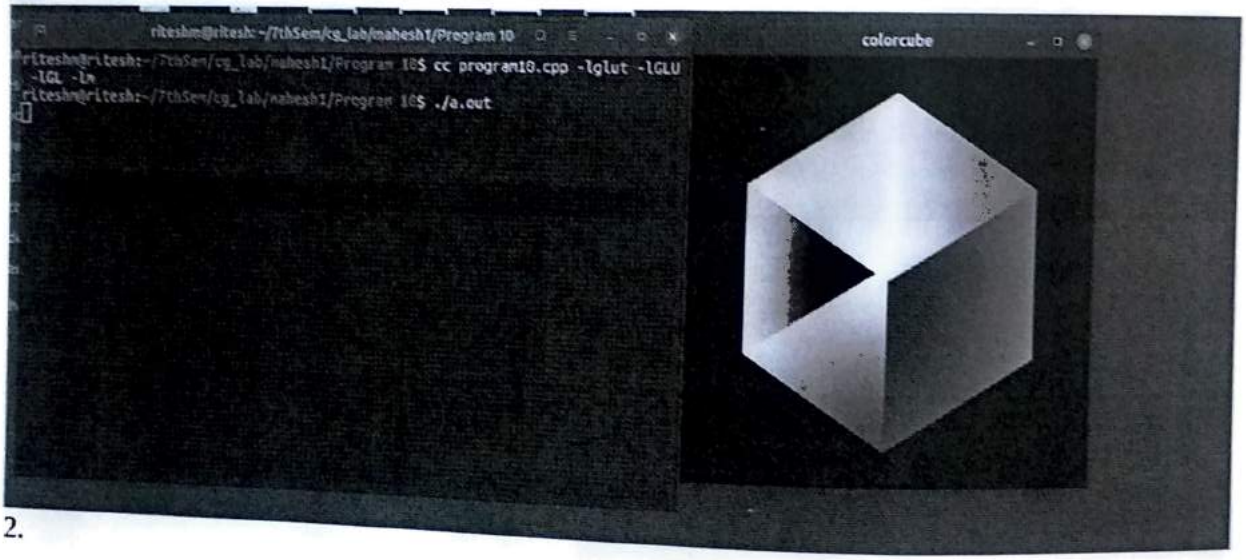
Experiment No. ..... Page No. .....

```
void myDisp() {
    glClear(GL_COLOR_BUFFER_BIT);
    carList();
    moveCar(s);
    wheelList();
}
void mouse(int btn, int state, int x, int y) {
    if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        s = 5;
        myDisp();
    }
    else if(btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
        s = 2;
        myDisp();
    }
}
int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(600, 600);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("car");
    myInit();
    glutDisplayFunc(myDisp);
    glutMouseFunc(mouse);
    glutKeyboardFunc(myKeyboard);
    glutMainLoop();
}
```

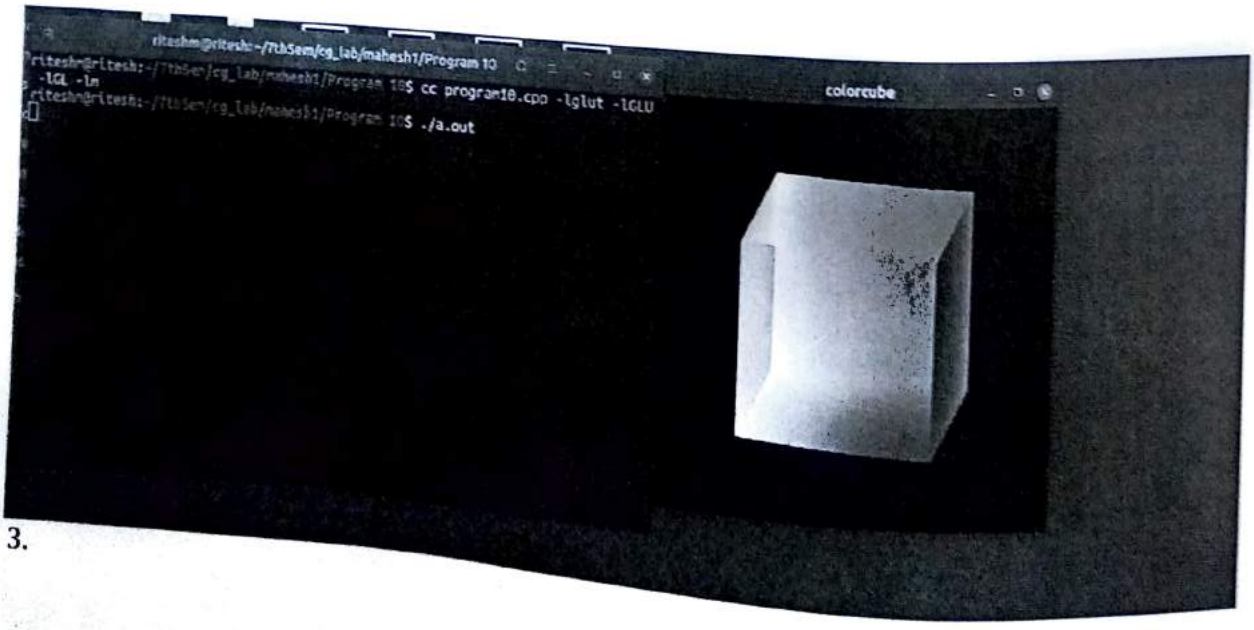
Output:


riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10$ cc program10.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10$ ./a.out
colorcube

1.


riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10$ cc program10.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10$ ./a.out
colorcube

2.


riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10$ cc program10.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 10$ ./a.out
colorcube

3.

Name of Experiment ..... Color Spin Cube ..... Date .....

Experiment No. ..... 10 ..... Page No. .....

1: Write a program to create a color cube and spin it using OpenGL Transforms

```
#include<stdio.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};
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();
```

Name of Experiment ..... Date .....

Experiment No..... Page No.....

```
    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);
    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() {
    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);
```

Experiment No..... Date .....

Page No.....

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);

}

void main(int argc, char **argv){

glutInit(&argc, argv);

glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

glutInitWindowPosition(100,100);

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);

glColorPointer(3, GL_FLOAT, 0, colors);

glNormalPointer(GL_FLOAT, 0, normals);

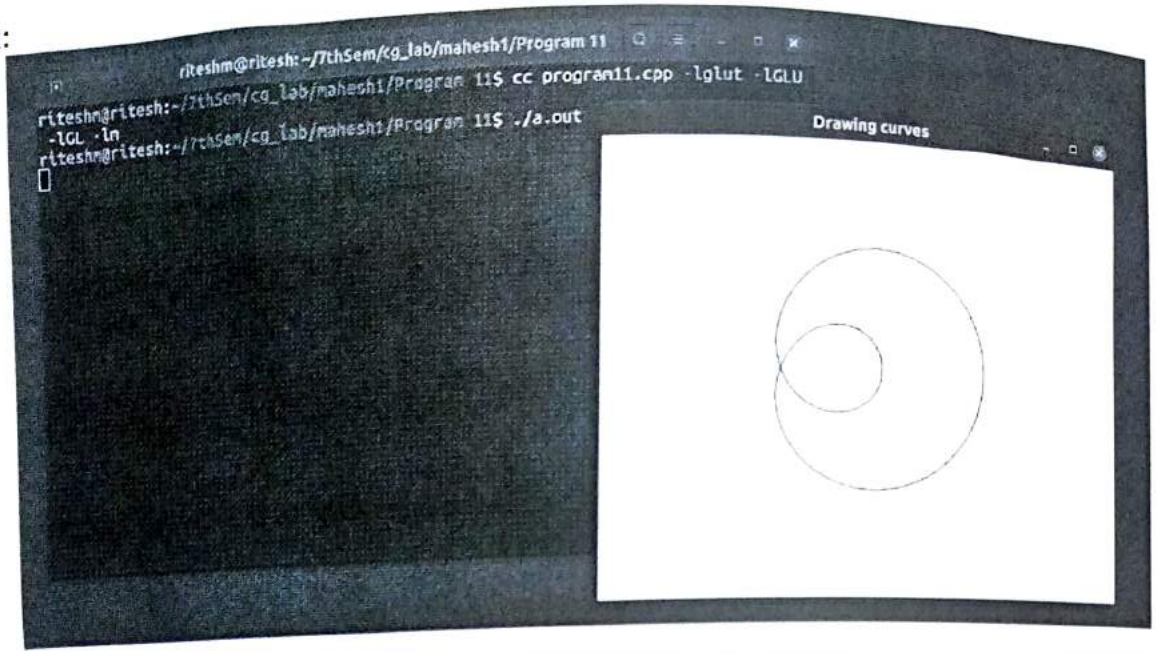
glColor3f(1.0, 1.0, 1.0);

glutMainLoop();

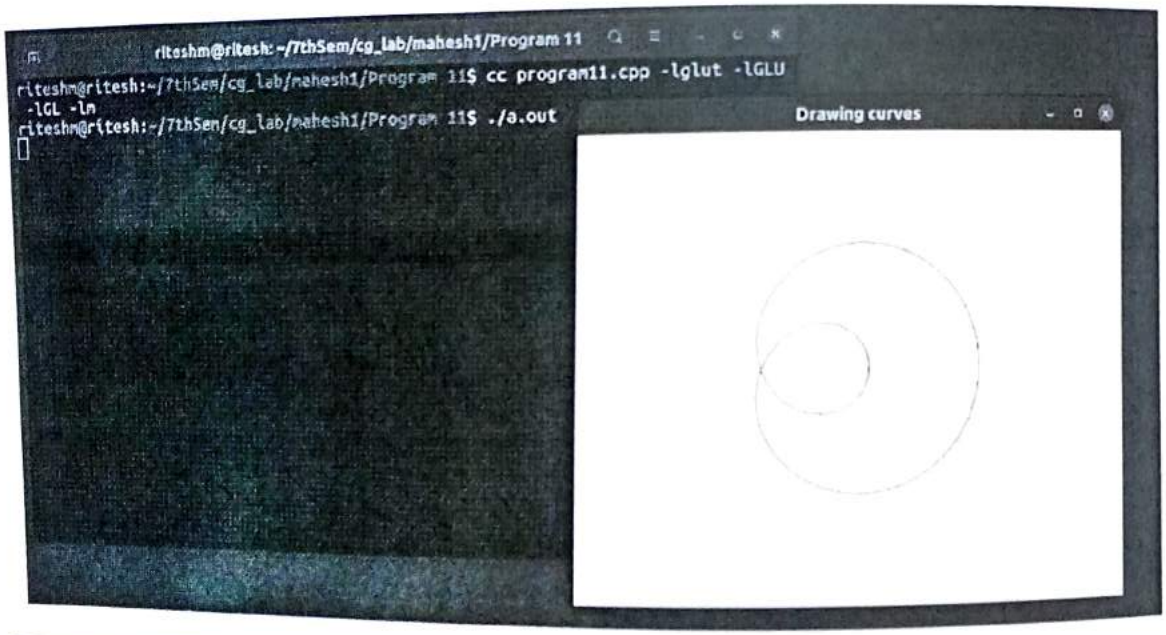
}

Output:

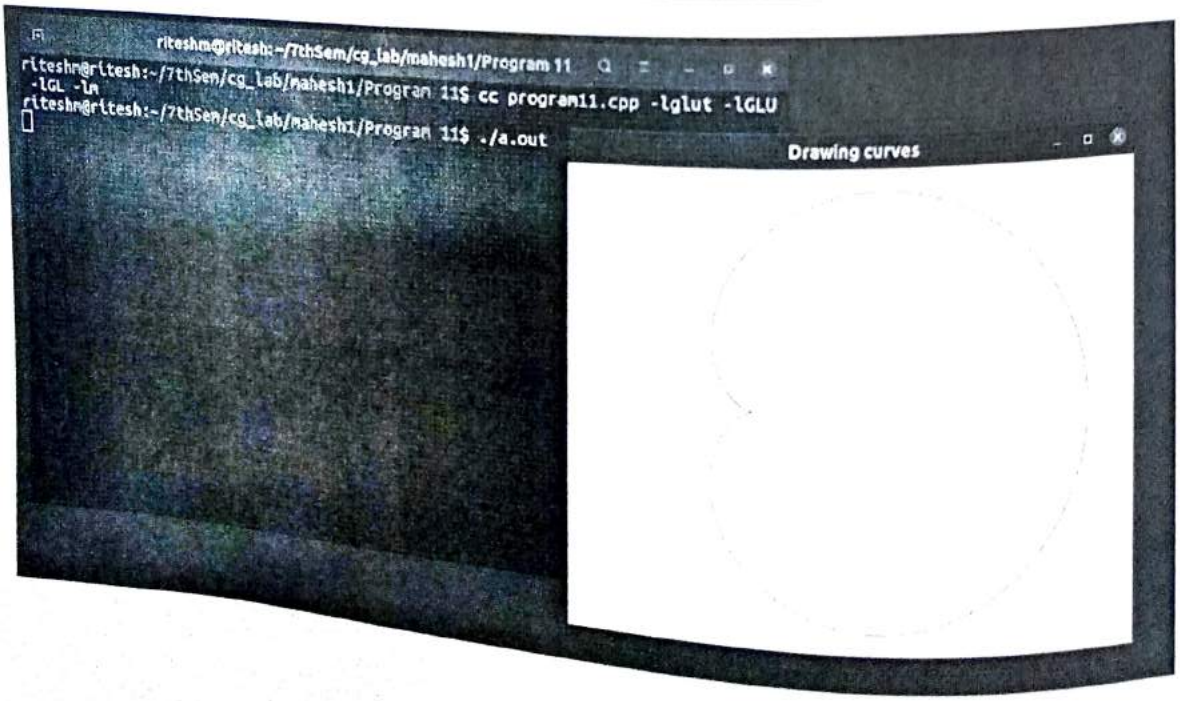
1.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 11
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ cc program11.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ ./a.out
Drawing curves

2.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 11
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ cc program11.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ ./a.out
Drawing curves

3.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 11
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ cc program11.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ ./a.out
Drawing curves

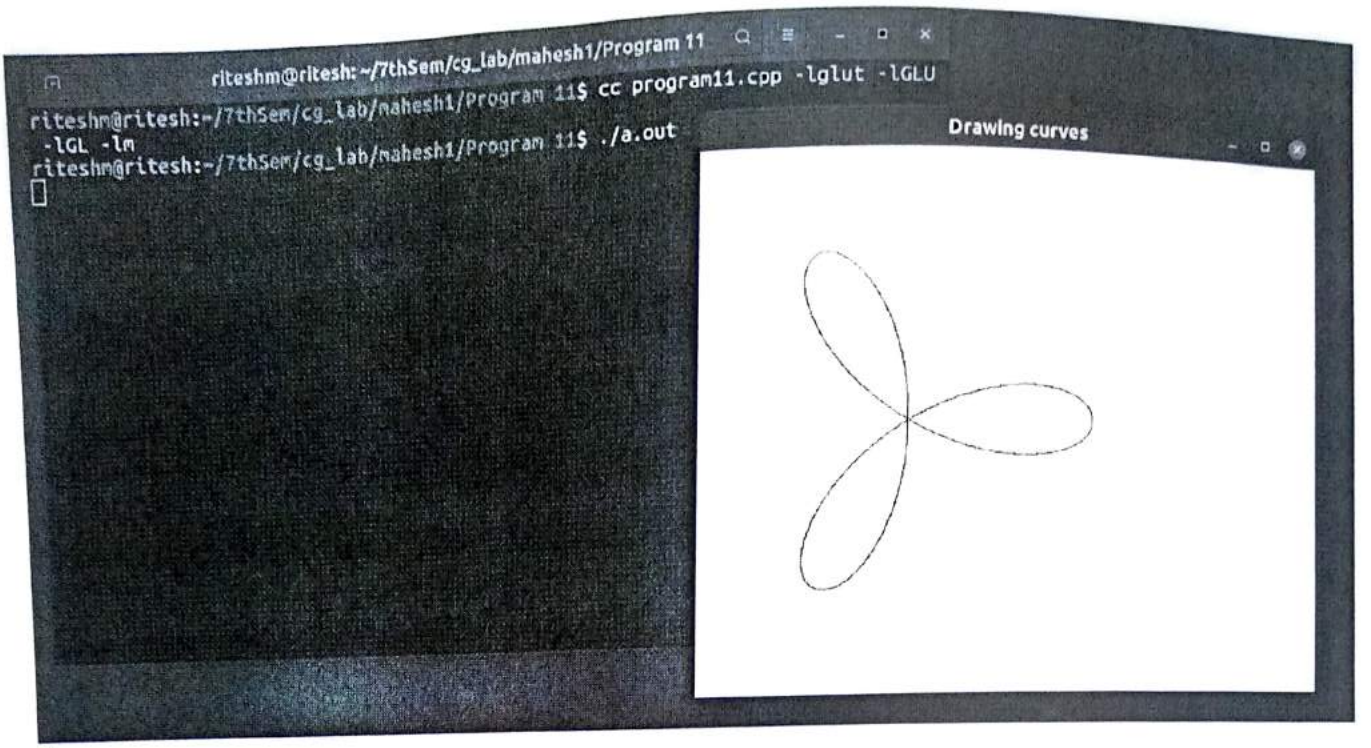
Name of Experiment ..... Curve Menu

Experiment No. ..... 11 ..... Date .....

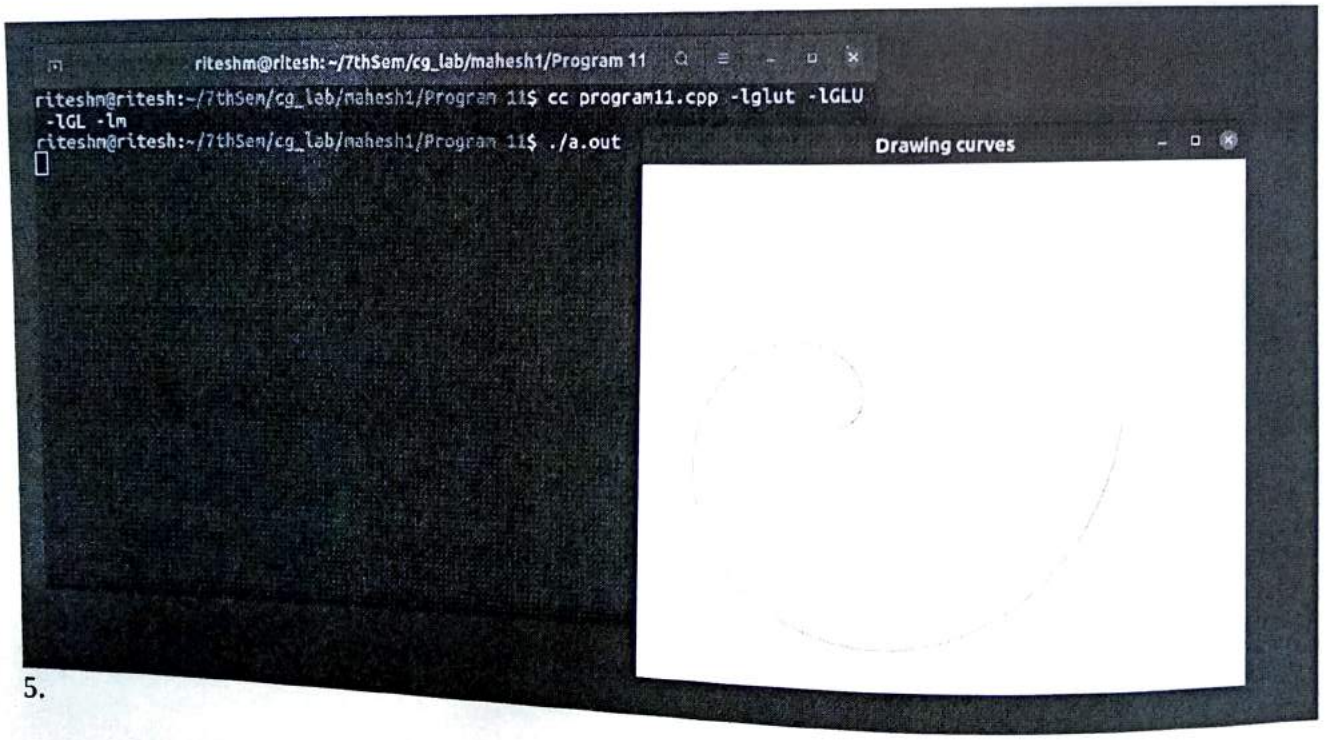
Page No. .....

11. Write a program to generate a Limacon, Cardioid, Three-Leaf, spiral

```
#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() {
    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();
}
void drawCurve(int curveNum) {
    const double twoPi = 6.283185;
    const int a = 175, b = 60;
    float r, theta, dtheta = 1.0 / float(a);
```


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 11
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ cc program11.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ ./a.out
Drawing curves

4.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 11
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ cc program11.cpp -lglut -lGLU
-lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 11$ ./a.out
Drawing curves

5.

```
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]);
curvePt[0].x = curvePt[1].x;
curvePt[0].y = curvePt[1].y;
theta += dtheta;
```

Experiment No. Date

Page No.

```
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 4:
        red=1;
        green=0;
        blue=0;
        break;
    case 3:
        red=0;
        green=1;
        blue=1; break;
    case 5:
        red=1;
        green=0;
        blue=1;
        break;
```

Experiment No.....

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

Name of Experiment ..... Date .....

Experiment No..... Page No.....

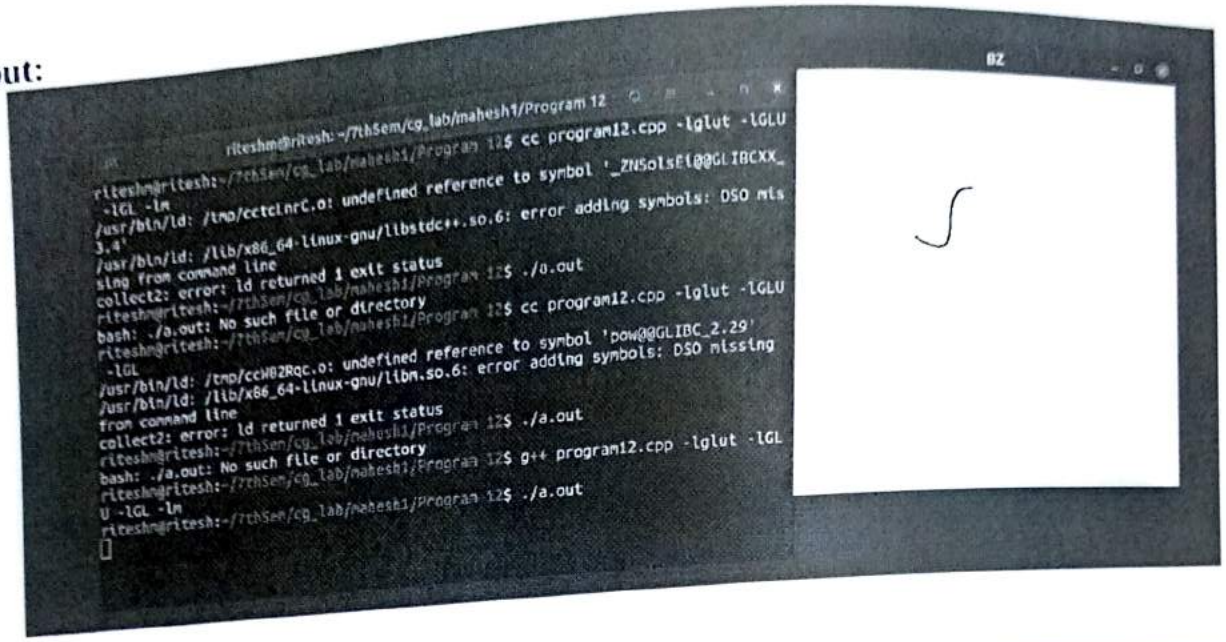
```
void main (int argc, char **argv){
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (w, h);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Drawing curves");
    int curveid = glutCreateMenu (drawcurve);
    glutAddMenuEntry ("Limacon", 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);
    glutAddMenuEntry ("White", 7);
    glutAttachMenu (GLUT_LEFT_BUTTON);
    glutCreateMenu (mainmenu);
    glutAddSubMenu ("drawcurve", curveId);
    glutAddSubMenu ("colors", colorId);
    glutAddMenuEntry ("quit", 3);
    glutAttachMenu (GLUT_LEFT_BUTTON);
    myinit ();
```

Name of Experiment ..... Date .....

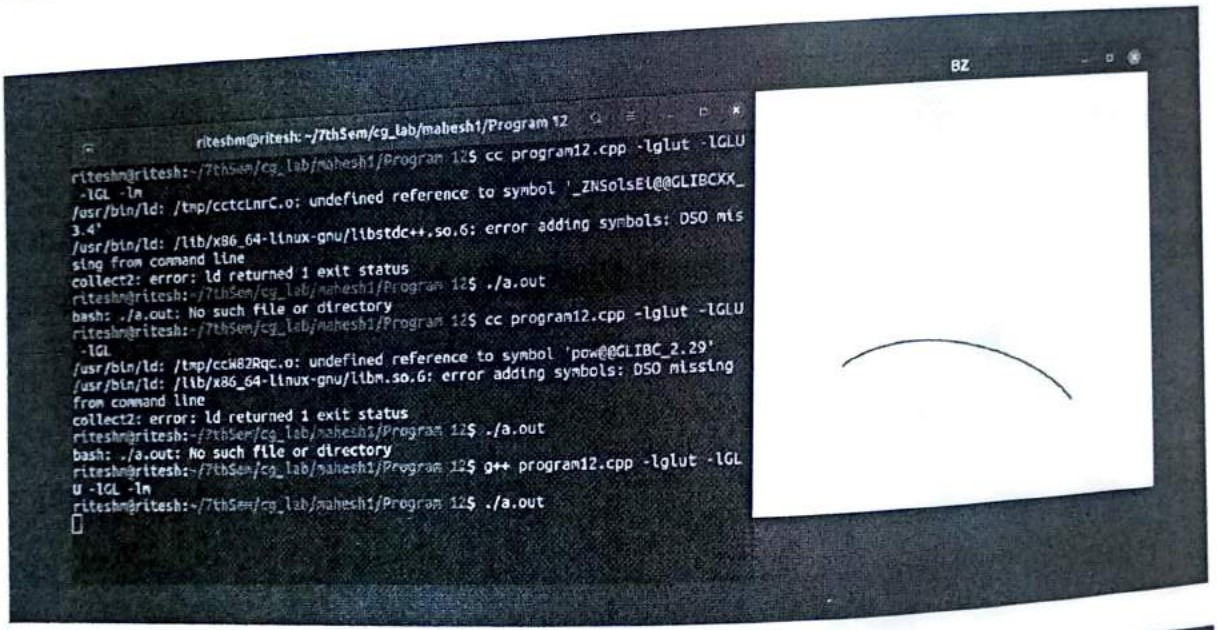
Experiment No..... Page No.....

```
glutDisplayFunc(mydisplay);
glutReshapeFunc(myreshape);
glutMainLoop();
}
```

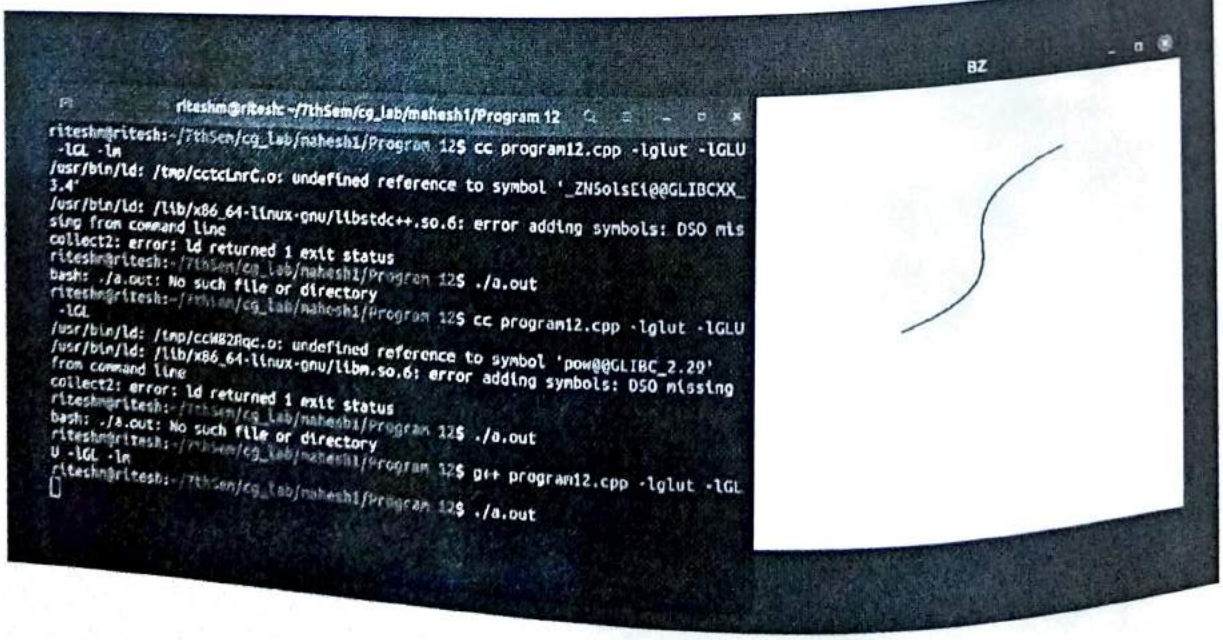
Output:


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 12
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ cc program12.cpp -lglut -lGLU
-lGL -lm
/usr/bin/ld: /tmp/cctcLnrC.o: undefined reference to symbol '_ZNSolsEl@@GLIBCXX_
3.4'
/usr/bin/ld: /lib/x86_64-linux-gnu/libstdc++.so.6: error adding symbols: DSO mis
sing from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
bash: ./a.out: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ cc program12.cpp -lglut -lGLU
-lGL
/usr/bin/ld: /tmp/ccW82Rqc.o: undefined reference to symbol 'pow@@GLIBC_2.29'
/usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing
from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
bash: ./a.out: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ g++ program12.cpp -lglut -lGL
U -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
BZ

1.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 12
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ cc program12.cpp -lglut -lGLU
-lGL -lm
/usr/bin/ld: /tmp/cctcLnrC.o: undefined reference to symbol '_ZNSolsEl@@GLIBCXX_
3.4'
/usr/bin/ld: /lib/x86_64-linux-gnu/libstdc++.so.6: error adding symbols: DSO mis
sing from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
bash: ./a.out: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ cc program12.cpp -lglut -lGLU
-lGL
/usr/bin/ld: /tmp/ccW82Rqc.o: undefined reference to symbol 'pow@@GLIBC_2.29'
/usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing
from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
bash: ./a.out: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ g++ program12.cpp -lglut -lGL
U -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
BZ

2.


riteshm@ritesh: ~/7thSem/cg_lab/mahesh1/Program 12
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ cc program12.cpp -lglut -lGLU
-lGL -lm
/usr/bin/ld: /tmp/cctcLnrC.o: undefined reference to symbol '_ZNSolsEl@@GLIBCXX_
3.4'
/usr/bin/ld: /lib/x86_64-linux-gnu/libstdc++.so.6: error adding symbols: DSO mis
sing from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
bash: ./a.out: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ cc program12.cpp -lglut -lGLU
-lGL
/usr/bin/ld: /tmp/ccW82Rqc.o: undefined reference to symbol 'pow@@GLIBC_2.29'
/usr/bin/ld: /lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing
from command line
collect2: error: ld returned 1 exit status
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
bash: ./a.out: No such file or directory
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ g++ program12.cpp -lglut -lGL
U -lGL -lm
riteshm@ritesh:~/7thSem/cg_lab/mahesh1/Program 12$ ./a.out
BZ

3.

Name of Experiment ..... Date .....

Experiment No. 12 Page No.....

2 Write a program to construct Bezier curve, control points are supplied through keyboard/mouse.

```
#include<iostream>
#include<math.h>
#include<glglut.h>
using namespace std;
float t,g,r, x1[4], yc[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];
```

Name of Experiment ..... Date .....

Experiment No. ..... Page No. .....

```
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);
}
glColor3f(1.0,0);
for(i=0; i<4; i++){
glVertex2f(x1[i], yc[i]);
glEnd();
glFlush();
}
}
void myMouse(int btn, int state, 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(x, 500-y);
glEnd();
glFlush();
flag++;
}
if(flag>=4 && btn==GLUT_LEFT_BUTTON){
glColor3f(0,0,1);
display();
flag=0;
```

Experiment No.

```
int main (int argc, char* argv[]){
    glut Init (&argc, argv);
    glut InitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glut InitWindowSize (500,500);
    glut InitWindowPosition (0,0);
    glut CreateWindow ("BZ");
    glut DisplayFunc (display);
    glut MouseFunc (mymouse);
    myinit();
    glutMainLoop;
}
```