Program 1:

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

```
#include<glut/glut.h>
#include<stdio.h>
int x1, y1, x2, y2;

// displays a point
void display(int x, int y){
      glBegin(GL_POINTS);
            glVertex2i(x, y);
      glEnd();
}
// implements bresenham's line drawing algorithm
void draw_line(){
      glClear(GL_COLOR_BUFFER_BIT);
      int dx, dy, i, p;
      int incx=1, incy=1, inc1, inc2;
      int x=x1, y=y1;

      dx = x2 - x1;
      dy = y2 - y1;

      if (dx<0){
            dx = -dx;
            incx = -1;
      }
      if (dy<0){
            dy = -dy;
            incy = -1;
      }

      // slope is less than 1
      if (dx>dy){
            p = 2 * dy - dx;                    // initial decision parameter
            inc1 = 2 * (dy - dx);
            inc2 = 2 * dy;
            for (i = 0; i<dx; i++){
                  display(x, y);
                  if (p >= 0){
                        y += incy;
                        p += inc1;
                  } else
                        p += inc2;
                  x += incx;

            }
      }
```


Date _____

Expt. No. 1 Page No. _____

PROGRAM 1

Write a program to generate a line using Bresenham's line drawing technique. Consider slope >1 and slope <1. User must be able to specify inputs through mouse/keyboard.

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

int x1, x2, y1, y2;

// displays a point
void display (int x, int y) {
      glBegin (GL_POINTS);
            glVertex2i (x, y);
      glEnd();
}

// implements Bresenham's line drawing algorithm
void draw_line() {
      glClear (GL_COLOR_BUFFER_BIT);

      int dx, dy, i, p;
      int incx = 1, incy = 1, inc1, inc2;
      int x = x1, y = y1;

      dx = x2 - x1;
      dy = y2 - y1;
```

Teacher's Signature : _____

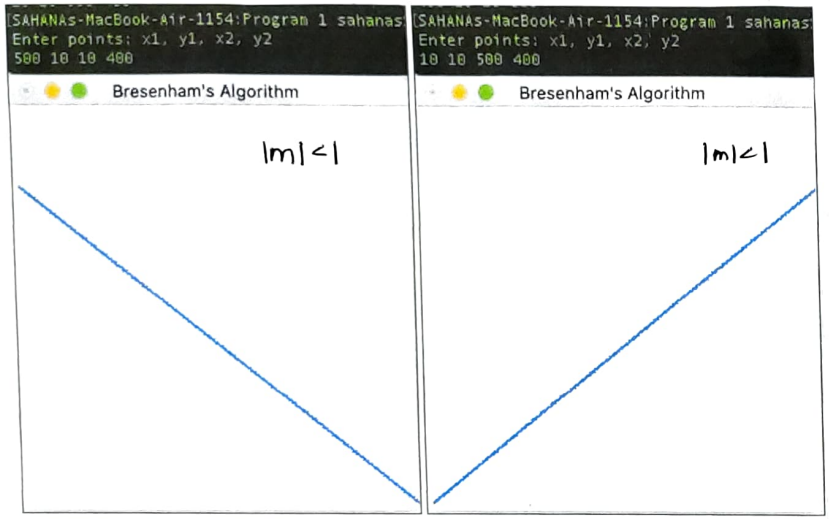
Date .....

Expt. No. ..... Page No. .....

```
if (dx<0) {
    dx = -dx;
    incx = -1;
}
if (dy<0) {
    dy = -dy;
    incy = -1;
}

if (dx > dy) {          // slope is less than 1
    p = 2 * dy - dx;    // initial decision parameter
    inc1 = 2 * (dy-dx);
    inc2 = 2 * dy;
    for (i = 0; i<dx; i++) {
        display (x, y);
        if (p>=0) {
            y+= incy;
            p += inc1;
        } else
            p += inc2;
        x+ = incx;
    }
} else {                // slope is greater than 1
    p=2 * dx - dy;
    inc1 = 2*(dx-dy);
    inc2 = 2 * dy;
    for (i=0; i<dy; i++) {
        display (x, y);
```

Teacher's Signature : _____

OUTPUT:



Date

Expt. No. Page No.

```
        if (p >= 0) {
            x += incx;
            p += inc1;
        }else
            p += inc2;
            y += incy;
        }
    }
    glFlush();
}

void myInit() {
    glClearColor(1.0, 1.0, 1.0, 1.0); // set a white bg
    glColor3f(0.0, 0.0, 1.0); // blue lines
    glPointSize(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 500, 0, 500);
}

int main(int argc, char* argv[]) {
    printf("Enter points: x1, y1, x2, y2\n");
    scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(300, 300);
    glutInitWindowPosition(5, 5);
    glutCreateWindow("Bresenham's algorithm");
    glutDisplayFunc(draw_line);
    myInit();
    glutMainLoop();
}
```

Teacher's Signature : _____

Program 2:

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

```
#include<glut/glut.h>
#include<stdio.h>
#include<math.h>
int xc, yc, r;
int rx, ry, xce, yce;
   plots the points of a circle according to 8-fold symmetry
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();
}
// implements Bresenham's circle drawing algorithm
void circle_bres(){
      glClear(GL_COLOR_BUFFER_BIT);
      int x = 0, y = r;
      //initial decision parameter
      int d = 3 - 2 * r;
      while (x <= y){
            draw_circle(xc, yc, x, y);
            if (d < 0) d = d + 4 * x + 6;
            else{
                  d = d + 4 * (x - y) + 10;
                  y--;
            } x++;
      }
      glFlush();
}
// plots the points of a ellipse according to 4-fold symmetry
void draw_ellipse(int xce, int yce, int x, int y){
      glBegin(GL_POINTS);
            glVertex2i(x + xce, y + yce);
            glVertex2i(-x + xce, y + yce);
            glVertex2i(x + xce, -y + yce);
            glVertex2i(-x + xce, -y + yce);
      glEnd();
}
// implements mid point ellipse drawing algorithm
void mid_point_ellipse(){
      glClear(GL_COLOR_BUFFER_BIT);
      float dx, dy, d1, d2, x=0, y=ry;
      // Initial decision parameter of region 1
      d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
      dx = 2 * ry * ry * x;
      dy = 2 * rx * rx * y;
      // For region 1
      while (dx < dy){
            draw_ellipse(xce, yce, x, y);
            if (d1 < 0){
                  x++;
                  dx = dx + (2 * ry * ry);
```


Date .....

Expt. No. 2 Page No. .....

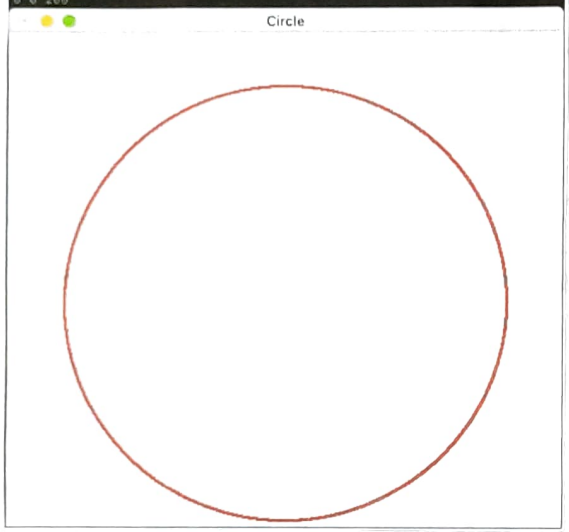
PROGRAM 2

Write a program to generate a circle and ellipse using Bresenham's circle drawing and ellipse drawing techniques using two windows, one to draw ellipse and other to draw circle. User can specify input keyboard or mouse.

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

int xc, yc, r;                    // circle parameters
int rx, ry, xce, yce;             // standard ellipse parameters
// plots the points of circle according to 8-fold symmetry
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();
}
```

Teacher's Signature : _____

```
SAHANAs-MacBook-Air-1154:Program 2 sahanas$ ./code2
Enter 1 to draw circle , 2 to draw ellipse
1
Enter coordinates of centre of circle and radius
0 0 200
```



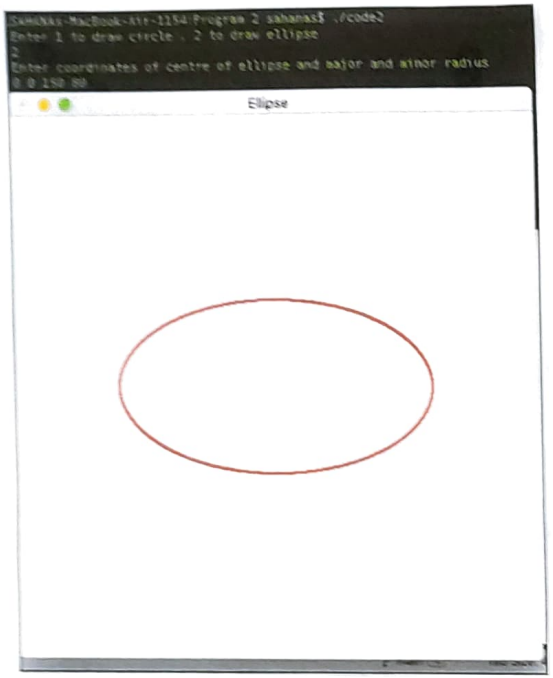
Date .....

Expt. No. .....

Page No. .....

```
// implements Bresenham's circle drawing algorithm
void circle_beres() {
    glClear(GL_COLOR_BUFFER_BIT);
    int x = 0, y = r;

    // initial decision parameter
    int d = 3 - 2 * r;
    while (x <= y) {
        draw_circle(xc, yc, x, y);
        if (d < 0)
            d += 4 * x + 6;
        else {
            d += 4 * (x - y) + 10;
            y--;
        }
        x++;
    }
    glFlush();
}

// Plots the points of ellipse according to 4-fold symmetry
void draw_ellipse(int xce, int yce, int x, int y) {
    glBegin(GL_POINTS);
        glVertex2i(xce + x, yce + y);
        glVertex2i(xce - x, yce + y);
        glVertex2i(xce + x, yce - y);
        glVertex2i(xce - x, yce - y);
    glEnd();
}
```

Teacher's Signature : _____



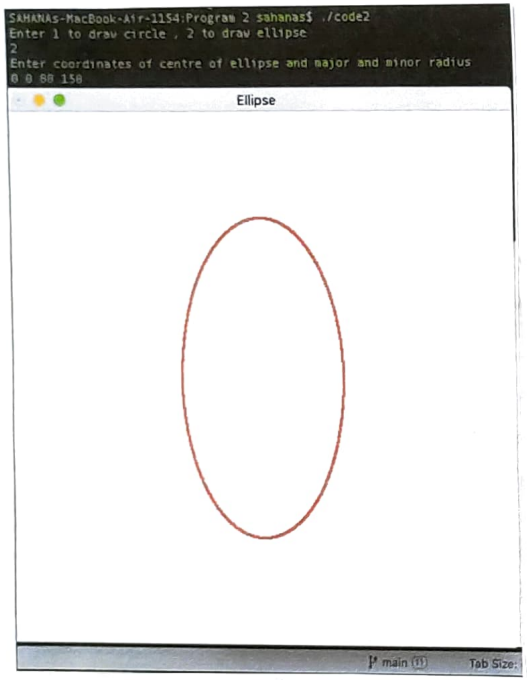
Date .....

Expt. No. .....

Page No. .....

```
//Implements mid point ellipse drawing algorithm
void mid_point_ellipse(){
    glClear(GL_COLOR_BUFFER_BIT);
    float dx, dy, d1, d2, x=0, y=ry;

    //Initial decision parameter of region 1
    d1 = (ry*ry) + (0.25*rx*rx) - (rx*rx*ry);
    dx = 2*ry*ry*x;
    dy = 2*rx*rx*y;

    //for region 1
    while(dx<dy){
        draw_ellipse(xce, yce, x, y);
        if(d1<0){
            x++;
            dx = dx + 2*(ry*ry);
            ~~dy = dy+~~ d1+= dx + (ry*ry);
        }else{
            xy++;
            y--;
            dx = dx + (2*ry*ry);
            dy = dy - (2*rx*rx);
            d1+ = dx - dy + (ry*ry);
        }
    }

//decision parameter for region 2
d2 = ((ry*ry) * ((x+0.5)*(x+0.5))) +
     ((rx*rx) * ((y-1)*(y-1))) - (rx*rx*ry*ry);
```

Teacher's Signature : _____



Expt. No.

Date

Page No.

```
// region 2
while (y>=0) {
    draw_ellipse (xce,yce,x,y);
    if (d2>0) {
        y--;
        dy -= 2*rx*rx;
        d2 += rx*rx -dy;
    }else {
        y--;
        x++;
        dy -= 2*rx*rx;
        dx += 2*ry*ry;
        d2 += dx-dy+rx*rx;
    }
}
glFlush();
}

void myInit() {
    glClearColor(1,1,1,1);
    glColor3f(1.0,0.0,0.0);
    glPointSize(3.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-250,250,-250,250);
}
```

Teacher's Signature : _____

Expt. No.

Date

Page No.

```
int main (int argc, char* argv[]) {

    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (0,0);

    printf (" enter 1 to draw circle, 2 to
                draw ellipse \n");
    int ch;
    scanf (&ch); scanf ("%d", &ch);
    switch (ch) {
        case 1:
            printf (" enter coordinates of center
                of circle and radius\n");
            scanf ("%d%d%d", &xc, &yc, &r);
            glutCreateWindow ("Circle");
            glutDisplayFunc (circle_bres);
            break;
        case 2:
            printf (" enter coordinates of center of
              standard ellipse and major axis and
              minor radius\n");
            scanf ("%d%d%d%d", &xce, &yce, &xx, &xy);
            glutCreateWindow ("Ellipse");
            glutDisplayFunc (mid_point_ellipse);
            break;
    }
    myInit();
    glutMainLoop();
}
```

Teacher's Signature : _____

Program 3:

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

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

int m;
typedef float point[3];
point tetra[4] = { {0,100,-100},{0,0,100},{100,-100,-100},{-100,-100,-100} };
// points of a tetrahedron

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

// Recursively divides the triangles till m reaches 0
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);
}

// Generate 2D Sierpenski's gasket for each face of a tetrahedron
void tetrahedron(){
    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();
}
```


Date .....

Expt. No. 3 .....

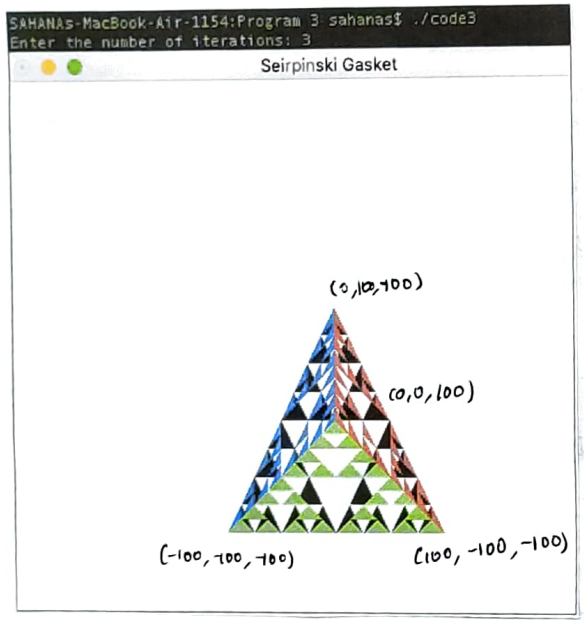
Page No. .....

PROGRAM 3

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

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

int m;
typedef float point[3];
point tetra[4] = {{0,100,-100},{0,0,100},{100,-100,-100},
                  {-100,-100,-100}}; //points of tetrahedron


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

//Recursively divides the triangles till m reaches 0
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;
```

Teacher's Signature : _____



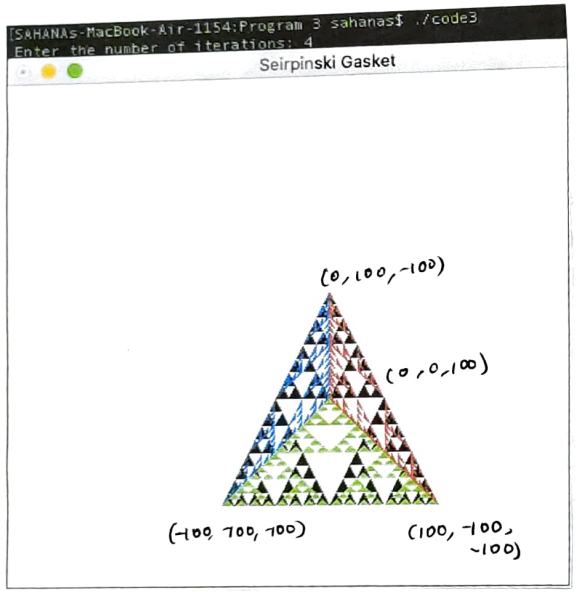
Date .....

Expt. No. .....

Page No. .....

```
    divide_triangle (a, v1, v2, m-1);
    divide_triangle (b, v2, v3, m-1);
    divide_triangle (b, v3, v1, m-1);
} else
    draw_triangle (a, b, c);
}

// Generate 2D-Sierpenski's gasket for each face of a
void tetrahedron() {                                    // tetrahedron
    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 myInit() {
    glClearColor (1,1,1,1);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
    glOrtho (-300.0, 300.0, -300.0, 300.0);
}
```

Teacher's Signature : _____



Date .....

Expt. No. .....

Page No. .....

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

    printf ("enter the number of iterations : ");
    scanf ("%d", &n);

    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition (100,200);
    glutInitWindowSize (500,500);
    glutCreateWindow ("S.ierpinski's gasket");
    glutDisplayFunc (tetrahedron);
    glEnable (GL_DEPTH_TEST);
    myInit();
    glutMainLoop();
}
```

Teacher's Signature : _____

Program 4:

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

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

int n, m;
int wx = 500, wy = 500;
float x[100], y[100];
static float intx[10] = {0}; // x intercepts of polygon with scanline


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

// finds the intersection of scanline with each edge of a polygon
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;
     }
     if (scanline > y1 && scanline < y2)
          intx[m++] = x1 + (scanline - y1) * (x2 - x1) / (y2 - y1);
}

// implements scan-line area fill algorithm
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);
          std::sort(intx, (intx + m));
          if (m >= 2)
               for (int i = 0; i < m; i = i + 2)
                    draw_line(intx[i], s1, intx[i + 1], s1);
     }
}
```


Date .....

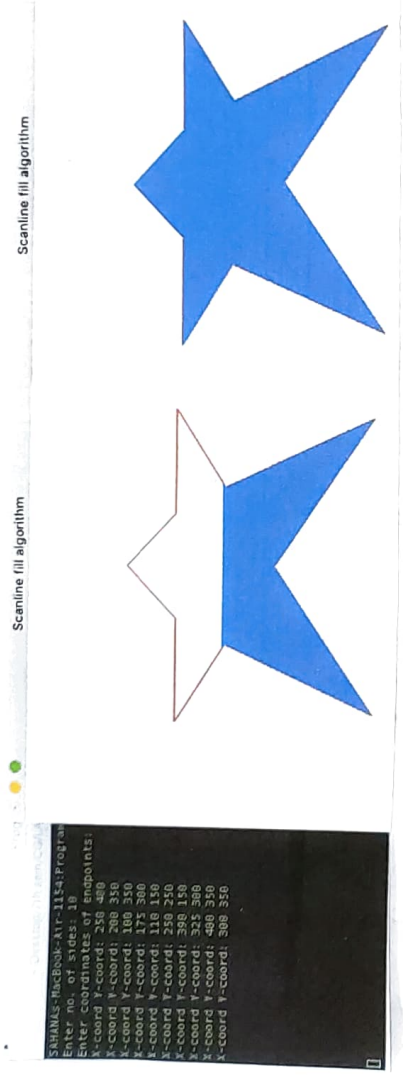
Expt. No. ..... Page No. .....

PROGRAM 4

Write a program to fill any polygon using scanline area fill algorithm

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

int n, m;
int wx = 500, wy = 500;
float x[100], y[100];
static float intx[10] = {0}; // x intercepts of polygon with scanline


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

//finds the intersection of scanline with every edge of polygon
void edgeDetect (float x1, float y1, float x2, float y2, int scanline) {
     float temp;
```

Teacher's Signature : _____



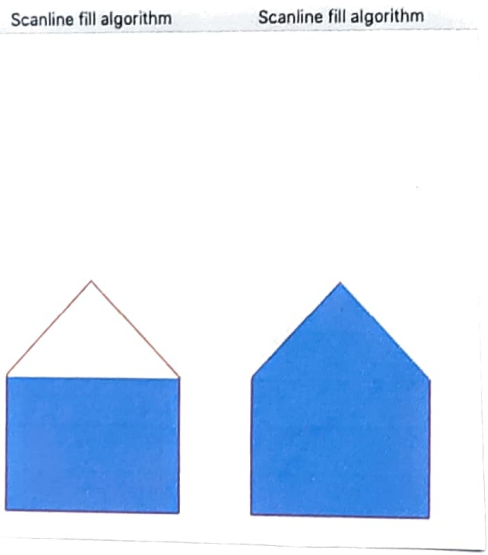
Date .....

Expt. No. ..... Page No. .....

```
if (y2<y1){
    temp = y1; y1 = y2; y2 = temp;
    temp = x1; x1 = x2; x2 = temp;
}
if (scanline > y1 && scanline < y2)
    intx[m++] = x1 + (scanline - y1) * (x2-x1) / (y2-y1);
}

//Implements scan-line area fill algorithm
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);
        std::sort (intx, (intx+m));
        if (m>=2)
            for(i=0; i<m; i+=2){
                draw_line (intx[i], s1, intx[i+1], s1);
    }
}

void display_filled_polygon(){
    glClear(GL_COLOR_BUFFER_BIT);
    glLineWidth(2);
    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);    // boundary of polygon
      for(int i=0; i<n; i++) glVertex2f(x[i], y[i]);
    glEnd();
    glColor(0,0,1);
    scanfill(x,y);
}
```

Teacher's Signature : _____

```
Enter no. of sides: 5
Enter coordinates of endpoints:
X-coord Y-coord: 250 320
X-coord Y-coord: 100 250
X-coord Y-coord: 100 150
X-coord Y-coord: 320 150
X-coord Y-coord: 320 250
```



Date .....

Expt. No. ..... Page No. .....

```
void myInit() {
    glClearColor (1,1,1,1);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D (0,wx,0,wy);
}


int main (int argc, char** argv) {
    printf ("Enter no. of sides: ");
    scanf ("%d", &n);
    printf ("Enter coordinates of endpoints");
    for (int i=0; i<n; i++) {
        printf ("X-coord Y-coord: ");
        scanf ("%f%f", &x[i], &y[i]);
    }


    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (wx,wy);
    glutInitWindowPosition (0,0);
    glutCreateWindow ("Scanline fill algorithm");
    myInit();
    glutMainLoop();
}
```

Teacher's Signature : _____

Program 5:

Write a program to create a house like 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<glut/glut.h>
#include<math.h>
#include<stdio.h>

// House coordinates
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 } };
int angle;
float m, c, theta;

// Display the initial house
void initialize(){
      glClearColor(0, 0, 0, 0);
      glClear(GL_COLOR_BUFFER_BIT);
      //NORMAL HOUSE
      glColor3f(1, 0, 0);
      glBegin(GL_LINE_LOOP);
      for (int i = 0; i < 11; i++)
            glVertex2fv(house[i]);
      glEnd();
      glFlush();
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glPushMatrix();
}
// Display the resultant house after required transformations
void result(){
      glColor3f(1, 1, 1);
      glBegin(GL_LINE_LOOP);
      for (int i = 0; i < 11; i++)
            glVertex2fv(house[i]);
      glEnd();
      glPopMatrix();
      glFlush();
}
// Rotating about (100,100) by the input angle
void rotate(){
      initialize();
      glTranslatef(100, 100, 0);
      glRotatef(angle, 0, 0, 1);
      glTranslatef(-100, -100, 0);
      result();
}
```


Date .....

Expt. No. 5 Page No. .....

PROGRAM 5

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

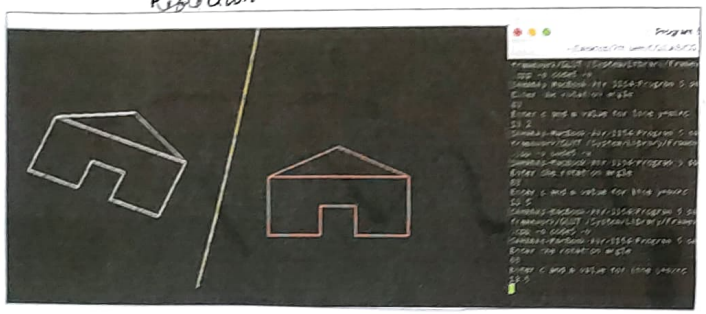
(i) Rotate it about a given fixed point using openGL transformation functions

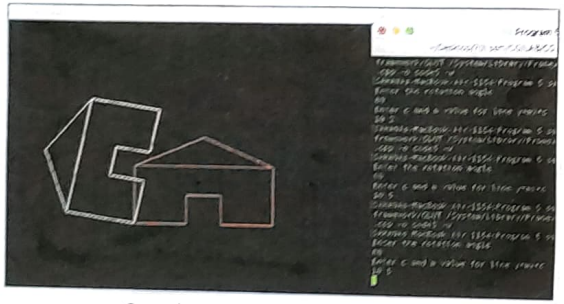
(ii) Reflect it about $y = mx+C$ axis using openGL transformation functions

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

// House coordinates
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}};
int angle;
float m, c, theta;

// Display the initial house
void initialize(){
   glClearColor (0, 0, 0, 0);
   glClear (GL_COLOR_BUFFER_BIT);

   //normal house
   glColor3f (1, 0, 0);
   glBegin (GL_LINE_LOOP);
   for (int i=0; i<11; i++)
      glVertex2fv (house[i]);
   glEnd();
   glFlush();
```

Teacher's Signature : _____

Reflection





$\theta = 80^\circ$
$c = 13$
$m = 5$

Rotation

Date .....

Expt. No. ..... Page No. .....

```
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity();
    glPushMatrix();
}

//Display the resultant house after required transformations
void result() {
    glColor3f(1, 1, 1);
    glBegin(GL_LINE_LOOP);
    for (int i=0; i<11; i++)
        glVertex2fv(house[i]);
    glEnd();
    glPopMatrix();
    glFlush();
}

// Rotation about (100,100) by the input angle
void rotate() {
    initilize();
    glTranslatef(100,100,0);
    glRotatef(angle, 0, 0, 1);
    glTranslatef(-100,-100, 0);
    result();
}

//Reflect about ~~now~~ y = mx + c
void reflect() {
    initialize();
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
// draw the axis  y = mx + c
float x1 = 0, x2 = 500;
float y1 = m * x1 + c;
float y2 = m * x2 + c;
glBegin (GL_LINES);
    glVertex2f (x1, y1);
    glVertex2f (x2, y2);
glEnd();
glFlush();

//Reflection
theta = atan(m);
theta = theta * 180/3.14;
glTranslate (0, c, 0);
glRotatef (theta, 0, 0, 1);
glScale (1, -1, 1);
glRotatef (-theta, 0, 0, 1);
glTranslate (0, -c, 0);
result();
}

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

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
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);
}

int main (int argc, char** argv) {
    printf ("Enter the rotation angle\n");
    scanf ("%d", &angle);
    printf ("enter c and m 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");
    glutDisplayFunc (rotate);
    glutMouseFunc (mouse);
    myInit ();
    glutMainLoop ();
}
```

Teacher's Signature : _____

Program 6:

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

```
#include<glut/glut.h>
#include<stdio.h>
#include<stdlib.h>
#define true 1
#define false 0
double xmin, ymin, xmax, ymax;  // clipping window coordinates
double xvmin, yvmin, xvmax, yvmax; // viewport coordinates

const int LEFT = 1;
const int RIGHT = 2;
const int BOTTOM = 4;
const int TOP = 8;
int n;
struct line_segment {
      int x1, y1;
      int x2, y2;
} ls[10];
// Computes code for each point wrt clipping window
int computeoutcode(double x, double y){
      int code = 0;

      if (y > ymax)      code |= TOP;
      else if (y < ymin) code |= BOTTOM;

      if (x > xmax)      code |= RIGHT;
      else if (x < xmin) code |= LEFT;

      return code;
}

//Implements Cohen-Sutherland line clipping algorithm
void cohensuther(double x0, double y0, double x1, double y1){
      int outcode0, outcode1, outcodeout;
      bool accept = false, done = false;

      outcode0 = computeoutcode(x0, y0);
      outcode1 = computeoutcode(x1, y1);
      do{
            // if both the endpoints are inside
            if (!(outcode0 | outcode1)){
                  accept = true;
                  done = true;
            }
            // if both the endpoints are outside
            else if (outcode0 & outcode1)
                  done = true;
            // line intersects the clipping window atleast once
```


Date .....

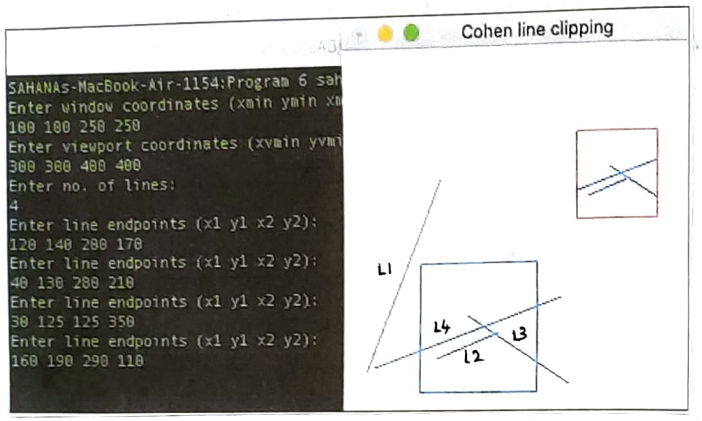
Expt. No. 6 Page No. .....

PROGRAM 6

Write a program to implement cohen-sutherland line clipping algorithm. Make provision to specify the input for multiple lines, window for clipping and viewport to display the clipped image.

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

#define true 1
#define false 0
double xmin, ymin, xmax, ymax; //clipping window
double xvmin, yvmin, xvmax, xvmin; //Viewport

const int LEFT = 1;
const int RIGHT = 2;
const int BOTTOM = 4;
const int TOP = 8;

int n;
struct line_segment {
      int x1, y1;
      int x2, y2;
} ls[10];
```

Teacher's Signature : _____



Date .....

Expt. No. ..... Page No. .....

```
//computes code for each point wrt clipping window
int computeoutcode (double x, double y) {
    int code = 0;
    if (y > ymax)   code |= TOP;
    else if (y < ymin) code |= BOTTOM;

    if (x > xmax)   code |= RIGHT;
    else if (x < xmin) code |= LEFT;
    return code;
}

//Implements Cohen-Sutherland line clipping algorithm
void cohensuther (double x0, double y0, double x1, double y1) {
    int outcode0, outcode1, outcodeout;
    bool accept = false, done = false;

    outcode0 = computeoutcode (x0, y0);
    outcode1 = computeoutcode (x1, y1);

    do {
      // if both the endpoints are inside
      if (!(outcode0 | outcode1)) {
            accept = true;
            accept = true;
            done
      }
      // if both the endpoints are outside
      else if (outcode0 & outcode1)
            done = true;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
// line intersects the clipping window atleast once
else {
    double x, y;
    outcode out = outcode0 ? outcode0 : outcode1;

    if (outcode out & TOP) {
        x = x0 + (x1 - x0) * (ymax - y0) / (y1 - y0);
        y = ymax;
    }
    else if (outcode out & BOTTOM) {
        x = x0 + (x1 - x0) * (ymin - y0) / (y1 - y0);
        y = ymin;
    }
    else if (outcode out & RIGHT) {
        y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
        x = xmax;
    }
    else {
        y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
        x = xmin;
    }
    // clipping
    if (outcode out == outcode0) {
        x0 = x; y0 = y;
        outcode0 = computeoutcode (x0, y0);
    }  ~~outcode1 = compute outcode (x1, y1);~~
    else {
        x1 = x; y1 = y;
        outcode1 = compute outcode (x1, y1);
    }
}
} while (!done);
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
if (accept) {
  // display the clipped vertices which are inside
  double sx = (vxmax - vxmin) / (xmax - xmin);
  double sy = (vymax - vymin) / (ymax - ymin);
  double vx0 = vxmin + (x0 - xmin) * sx;
  double vy0 = vymin + (y0 - ymin) * sy;
  double vx1 = vxmin + (x1 - xmin) * sx;
  double vy1 = vymin + (y1 - ymin) * sy;

  glColor3f(0,0,1);
  glBegin(GL_LINES);
    glVertex2d(vx0,vy0);
    glVertex2d(vx1,vy1);
  glEnd();
  }
}
void display(){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0,0,0);
    glBegin(GL_LINE_LOOP);
        glVertex2f(xmin, ymin);
        glVertex2f(xmax, ymin);
        glVertex2f(xmax, ymax);
        glVertex2f(xmin, ymax);
    glEnd();
    glBegin(GL_LINE_LOOP);
    for(int i=0; i<n; i++){
        glVertex2d(ls[i].x1, ls[i].y1);
        glVertex2d(ls[i].x0, ls[i].y2);
    glEnd();
```

Teacher's Signature : _____

Program 7:

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

```
#include <stdio.h>
#include <GLUT/glut.h>
double xmin, ymin, xmax, ymax;
double xvmin, yvmin, xvmax, yvmax;
int n;
struct line_segment {
      int x1, y1;
      int x2, y2;
} ls[10];
// check if the line is eligible for clipping
int cliptest(double p, double q, double* u1, double* u2){
      double r;
      if (p) r = q / p;

      if (p < 0.0){
            if (r > *u1) *u1 = r;
            if (r > *u2) return(false); // line portion is outside
      }else if (p > 0.0){
                  if (r < *u2)* u2 = r;
                  if (r < *u1) return(false); // line portion is outside
      }else
            if (q < 0.0) return(false); // line parallel to edge but outside
      return(true);
}
// Implements the liang barskey line clipping algorithm
void LiangBarsky(double x0, double y0, double x1, double y1){
      double dx = x1 - x0, dy = y1 - y0, u1 = 0.0, u2 = 1.0;
      if (cliptest(-dx, x0 - xmin, &u1, &u2))  // p1,q1
            if (cliptest(dx, xmax - x0, &u1, &u2)) // p2,q2
                  if (cliptest(-dy, y0 - ymin, &u1, &u2)) // p3,q3
                        if (cliptest(dy, ymax - y0, &u1, &u2)){ // p4,q4
                              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();
                        }
      }
```


Date _____

Expt. No. 7 Page No. _____

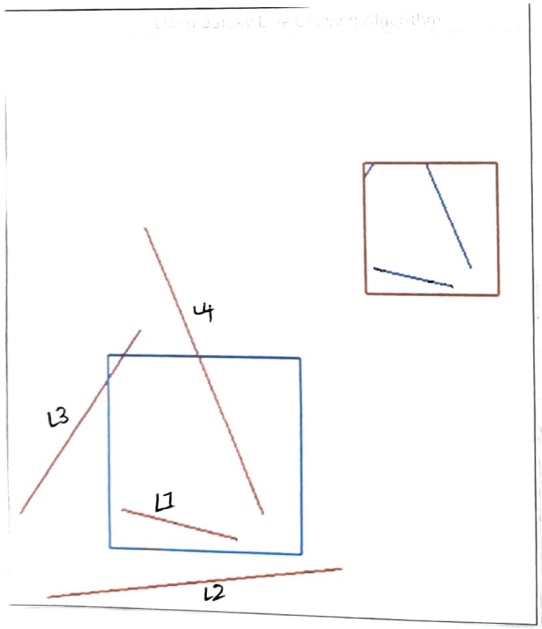
PROGRAM 7

Write a program to implement Liag Liang Barsky line clipping algorithm. Make provision to specify the input for multiple line, window for clipping & viewport.

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

double xmin, ymin, xmax, ymax;          //window
double xvmin, xvmin, xvmax, yvmax;      //viewport

int n;
struct line_segment {
      int x1, y1;
      int x2, y2;
} ls[10];
//check if the line is eligibe for clipping
int cliptest(double p, double q, double* u1, double* u2) {
      double r;
      if (p) r=q/p;
      if (p < 0.0) {
            if (r > *u1) *u1=r;
            if (r > *u2) return(false); //outside
      } else if (p>0.0) {
            if (r < *u2) *u2=r;
            if (r < *u1) return(false); //outside
      } else
            if (q < 0.0) return(false); //line parallel to edge
                                        but outside
      return(true);
}
```

Teacher's Signature : _____

```
Enter window coordinates: (xmin ymin xmax ymax)
100 100 250 250
Enter viewport coordinates: (xvmin yvmin xvmax yvmax)
300 300 400 400
Enter no. of lines:
4
Enter coordinates: (x1 y1 x2 y2)
110 130 200 110
Enter coordinates: (x1 y1 x2 y2)
90 125 125 270
Enter coordinates: (x1 y1 x2 y2)
50 60 280 90
Enter coordinates: (x1 y1 x2 y2)
130 350 220 130
```



Date .....

Expt. No. ..... Page No. .....

```
//Implements the Liang Barsky line clipping algorithm
void LiangBarsky(double x0, double y0, double x1, double y1){
  double dx=x1-x0, dy=y1-y0, u1=0.0, u2=1.0;

  if(cliptest(-dx,x0-xmin, &u1, &u2))  //p1, q1
    if(cliptest(dx, xmax-x0, &u1, &u2))  //p2, q2
      if(cliptest(-dy, y0-ymin, &u1, &u2))  //p3, q3
        if(cliptest(dy, ymax-y0, &u1, &u2))  //p4, q4
        {
          if(u1>0.0){
            x0 += u1*dx;
            y0 += u1*dy;
          }if(u2<1.0){
            x1 = x0+u2*dx;
            y1 = x0+u2*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);
          glVertex2f(vx0,vy0);
          glVertex2f(vx1,vy1);
        glEnd();
  }}
```

Teacher's Signature : _____

Date _____

Expt. No. _____ Page No. _____

```
void display () {
    glClear ( GL_COLOR_BUFFER_BIT);
    glColor3f (1.0, 0.0, 0.0);
    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();
    }
    glColor3f (0.0, 0.0, 1.0);
    glBegin (GL_LINE_LOOP);
        glVertex2f (xmin , ymin);
        glVertex2f (xmax , ymin);
        glVertex2f (xmax , ymax);
        glVertex2f (xmin , ymax);
    glEnd();

    glColor3f (1.0, 0.0, 0.0);
    glBegin (GL_LINE_LOOP);
        glVertex2f (xvmin, yvmin);
        glVertex2f (xvmax , yvmin);
        glVertex2f (xvmax, yvmax);
        glVertex2f (xvmin, yvmax);
    glEnd();
    for (int i=0 ; i <n ; i++)
        LiangBarsky (ls[i].x1, ls[i].y1, ls[i].x2, ls[i].y2);
    glFlush();
}
```

Teacher's Signature : _____

Date

Expt. No. Page No.

```
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 (0, 500, 0, 500);
}

int main (~~int~~ int argc, char** argv) {
   printf ("enter window coordinates :
                (xmin ,ymin , xmax, ymax \n");
   scanf ("%lf%lf%lf%lf", &xmin ,&ymin, &xmax, &ymax);
   printf ("enter ~~window~~ viewport coordinates : \n");
   scanf ("%lf %lf %lf %lf", &xvmin, &yvmin, &xvmax, &yvmax);
   printf ("enter no. of lines");
   scanf ("%d", &n);
   for (int i =0 ; i <n ; i ++) {
      printf ("enter coordinates : (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 (0,0);
   glutCreateWindow ("Liang Barsky");
   glutDisplayFunc (display);
   myInit();
   glutMainLoop();
}
```

Teacher's Signature :

Program 8:

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

```
#include<iostream>
#include<GLUT/glut.h>

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 = 20;

void drawPoly(int p[][2], int n) {
      glBegin(GL_POLYGON);
      for (int i = 0; i < n; i++)
            glVertex2f(p[i][0], p[i][1]);
      glEnd();
}

// Returns x-value of point of intersection
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;
}

// Returns y-value of point of intersection
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;
}

// clips all the edges w.r.t one clip edge of clipping area
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];
               // Calculating position of first and second point
            int i_pos = (x2 - x1) * (iy - y1) - (y2 - y1) * (ix - x1);
            int k_pos = (x2 - x1) * (ky - y1) - (y2 - y1) * (kx - x1);
            // Case 1 : When both points are inside
            if (i_pos >= 0 && k_pos >= 0){
                  new_points[new_poly_size][0] = kx;
                  new_points[new_poly_size][1] = ky;
                  new_poly_size++;
            }
            // Case 2: When only first point is outside
            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;
                  new_poly_size++;
```


Date _____

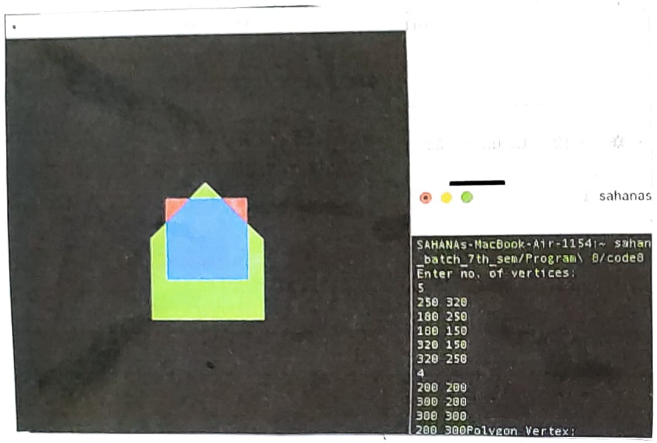
Expt. No. 8 Page No. _____

PROGRAM 8

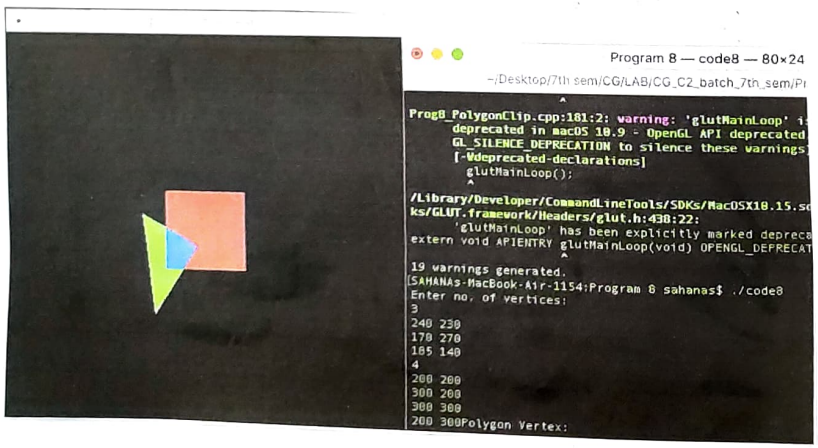
Write a program to implement the Sutherland-Hodgeman polygon clipping algorithm. Make provision to specify the input polygon and window for clipping.

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

int poly_size, poly_points[20][2], org_poly_size;
int org_poly_points[20][2], clipper size, clipper_points[20][2];
const int MAX_POINTS =20;

void drawPoly (int p[][2], int n) {
      glBegin (GL_POLYGON);
      for (int i=0; i<n; i++)
         glVertex2f( p[i][0], p[i][1]);
      glEnd();
}

int x_intersect (int x1, int y1, int x2, int y2, int x3, int y3, int x4,
int num = (x1*y2 - y1*x2) * (x3-x4) - (x1-x2)*(x3*y4 - y3*x4);   int y4){
int den = (x1-x2) * (y3 -y4) - (y1 -y2) * (x3-x4);
return num/den;
}

int y_intersect (int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
int num = (x1*y2-y1*x2) * (y3-y4) - (y1-y2) * (x3*y4 -y3*x4);
int den = (x1 - x2) * (y3 -y4) - (y1 -y2) * (x3-x4);
return num/den;
}
```

Teacher's Signature : _____



red - clipping window
blue - clipped polygon
green - polygon



Date .....

Expt. No. ..... Page No. .....

```
void clip(int poly_points[][2], int &poly_size, int x1, int y1, int x2, int y2)
{
    int new_points[MAXPOINTS][2], new_poly_size = 0;
    for (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);

        //case 1: When both points are inside
        if (i_pos >= 0 && k_pos >= 0) {
            new_points[new_poly_size][0] = kx;
            new_points[new_poly_size][1] = ky;
            new_poly_size++;
        }

        //case 2: When only first point is outside
        else if (i_pos < 0 && k_pos >= 0) {
            new_points[new_poly_size][0] = x_intersect(x1, y1,
                                           x2, y2, ix, iy, kx,
            new_points[new_poly_size][1] = y_intersect          ky);
            new_poly_size++;
            new_points[new_poly_size][0] = kx;
            new_points[new_poly_size][1] = ky;
            new_poly_size++;
        }

        //case 3: When only second point is outside
        else if (i_pos >= 0 && k_pos < 0) {
            new_points[new_poly_size][0] = x_intersect(x1, y1,
            new_points[new_poly_size][1]  x2, y2, ix, iy, kx,
            new_poly_size++;                = y_intersect(ky);
        }
```

Teacher's Signature : _____

Date

Expt. No. Page No.

```
// case 4: When both inputs are outside, do nothing
}
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 myInit(){
    glClearColor(0.0f,0.0f,0.0f,0.0f);
    glMatrixMode(GL_PROJECTION); glLoadIdentity();
    gluOrtho2D(0,500,0,500);
    glClear(GL_COLOR_BUFFER_BIT);
}
void display(){
    myInit();
    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][0],
            clipper_points[i][1], clipper_points[k][0],
            clipper_points[k][1]);
    }
    glColor3f(0,0,1);
    drawPoly(poly_points, poly_size);
    glFlush();
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
int main (int argc, char** argv) {
    printf ("enter no. of vertices: \n");
    scanf ("%d", &poly_size);
    org_poly_size = poly_size;
    for (int i = 0 ; i< clipper_size; i++) {
        printf ("Polygon Vertex:");
        scanf ("%d%d", &poly_points[i][0], &poly_points[i][1]);
        org_poly_points[i][0] = poly_points[i][0];
        org_poly_points[i][0] = poly_points[i][i];
    }
    printf ("enter no. of vertices of clipping window");
    scanf ("%d", &clipper_size);
    for (int i=0 ; i < clipper_size ; i++) {
        printf ("clip Vertex \n");
        scanf ("%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();
}
```

Teacher's Signature : _____

Program 9:

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

```
#include<GLUT/glut.h>
#include<stdio.h>
#include<stdlib.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();
}

void mykeyboard(unsigned char key, int x, int y) {
      switch (key) {
      case 't': glutPostRedisplay(); break;
      case 'q': exit(0);
      default: break;
      }
}

void myInit() {
      glClearColor(0, 0, 0, 0);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glOrtho(0, 600, 0, 600, 0, 600);
}

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


Date .....

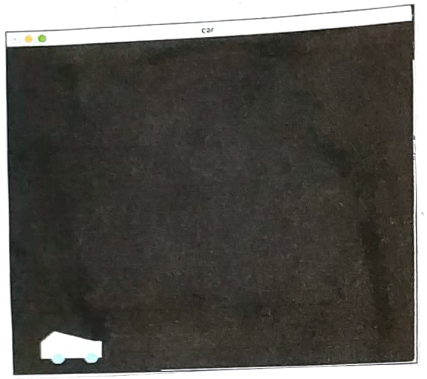
Expt. No. 9 Page No. .....

PROGRAM 9

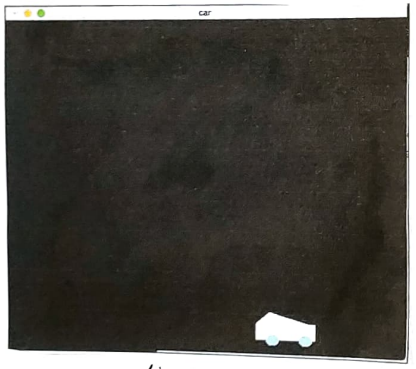
Write a program to model a car like figure using display lists and move a car from one end of screen to other end. User is able control speed with mouse.

```
#include <glut/glut.h>
#include <stdio.h>
#include <stdlib.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();
}
```

Teacher's Signature : _____



initial



final

Date .....

Expt. No. ..... Page No. .....

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

void mykeyboard (unsigned char key, int x, int y) {
    switch(key) {
        case 't': glutPostRedisplay(); break;
        case 'q': exit(0);
        default : break;
    } }

void myInit () {
    glClearColor (0,0,0,0);
    glMatrixMode (GL_PROJECTION); glLoadIdentity();
    glOrtho (0,600, 0,600, 0,600);
}
void draw_wheel () {
    glColor3f (0,1,1);
    glutSolidSphere (10,25,25);
}
void moveCar (float s) {
    glTranslatef (s, 0, 0);
    glCallList (CAR);
    glPushMatrix();
    glTranslate (75,25, 0);
    glCallList (WHEEL);
    glPopMatrix();
```

Teacher's Signature : _____

Date _____

Expt. No. _____ Page No. _____

```
    glPushMatrix();
    glTranslate(75, 27, 0);
    glCallList(WHEEL);
    glPopMatrix();
    glFlush();
}
void myDisp(){
    glClear(GL_COLOR_BUFFER_BIT);
    car list();
    move car(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);
    glutInitWindowPosition(100, 100);
    glutInitWindowSize(600, 300);
    glutCreateWindow("car");
    myInit(); glutDisplayFunc(myDisp);
    glutMouseFunc(mouse);
    glutKeyboardFunc(mykeyboard);
    glutMainLoop();
}
```

Teacher's Signature : _____

Program 10:

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

```
#include <stdlib.h>
#include <GLUT/glut.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 display(void){
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     glLoadIdentity();
     glRotatef(theta[0], 1.0, 0.0, 0.0);
     glRotatef(theta[1], 0.0, 1.0, 0.0);
     glRotatef(theta[2], 0.0, 0.0, 1.0);

     glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);

     glBegin(GL_LINES);
          glVertex3f(0.0, 0.0, 0.0);
          glVertex3f(1.0, 1.0, 1.0);
     glEnd();
     glFlush();
}

// Idle callback, spin cube 2 degrees about selected axis
void spinCube(){
     delay(0.01);
     theta[axis] += 2.0;
     if (theta[axis] > 360.0) theta[axis] -= 360.0;
     glutPostRedisplay();
}
void mouse(int btn, int state, int x, int y){
     if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
     if (btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
```


Date .....

Expt. No. 10 Page No. .....

PROGRAM 10.

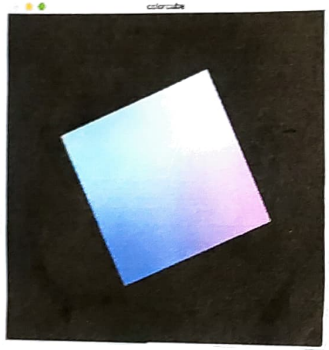
Write a program to create a color cube and spin it using OpenGL transformations.

```
#include <stdlib.h>
#include <GLUT/glut.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,-1.0,1.0,1.0};

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

Teacher's Signature : _____

about x axis

about z axis



about y axis

Date .....

Expt. No. ..... Page No. .....

```
void display (void) {
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity ();
    glRotatef (theta[0], 1, 0, 0);
    glRotatef (theta[1], 0, 1, 0);
    glRotatef (theta[2], 0, 0, 1);

    glDrawElements (GL_QUADS, 24, GL_UNSIGNED_BYTE,
                    cubeIndices);
    glBegin (GL_LINES);
        glVertex3f (0, 0, 0);
        glVertex3f (1, 1, 1);
    glEnd();
    glFlush();
}

// idle callback to spin 2° about selected axis
void spinCube () {
    delay (0.01);
    theta[axis] += 2.0;
    if (theta[axis] > 360.0) theta[axis] -= 360.0;
    glutPostRedisplay ();
}

void mouse (int btn, int state, int x, int y) {
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
    if (btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
    if (btn == GLUT_DOWN_BUTTON && state == GLUT_DOWN) axis = 2;
}
```

Teacher's Signature : _____

Date

Expt. No. Page No.

```
void myReshape (int w, int h) {
    glViewport (0,0,w,h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    if (w <= h)
        glOrtho(-2, 2, -2 * (GLfloat) h / (GLfloat) w,
                2 * (GLfloat) h / (GLfloat) w, -10, 10);
    else
        glOrtho (-2 * (GLfloat) w / (GLfloat) h, 2 * (GLfloat) w
                / (GLfloat) h, -2, 2, -10, 10);
    glMatrixMode (GL_MODELVIEW);
}
int main (int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500,500);
    glutCreateWindow ("color cube");
    glutReshapeFunc (myReshape); glutIdleFunc (spinCube);
    glutDisplayFunc (display); 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(1,1,1);
    glutMainLoop();
}
```

Teacher's Signature : _____

Program 11:

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

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

struct screenPt {
       int x, 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) {
       glLineWidth(3.0);
       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);
       int x0 = 200, y0 = 250;
       screenPt curvePt[2];
       glColor3f(red, green, blue);
       curvePt[0].x = x0; curvePt[0].y = y0;
       glClear(GL_COLOR_BUFFER_BIT);
       switch (curveNum) {
              case limacon: curvePt[0].x += a + b; break;
              case cardioid: curvePt[0].x += a + a; break;
              case threeLeaf: curvePt[0].x += a; break;
              case spiral: break;
              default: break;
       }
       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);
              lineSegment(curvePt[0], curvePt[1]);
              curvePt[0].x = curvePt[1].x;
              curvePt[0].y = curvePt[1].y;
```


Date _____

Expt. No. 11 Page No. _____

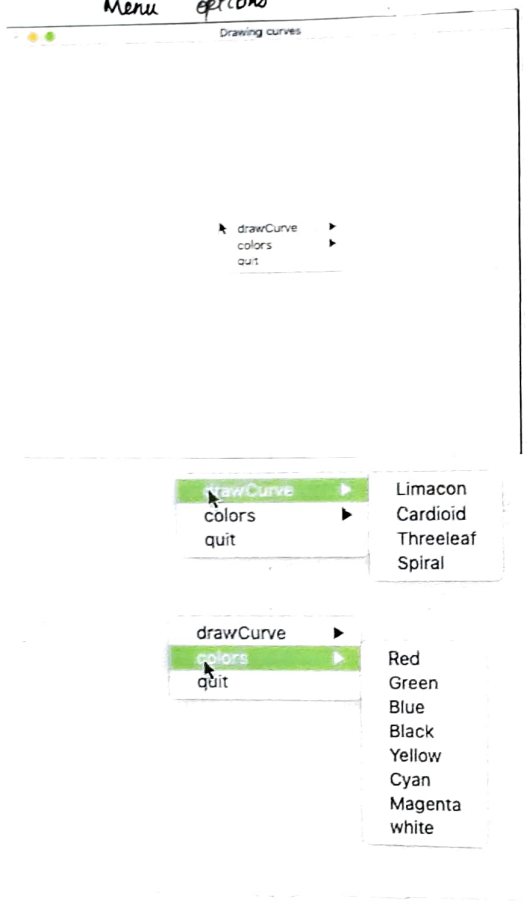
PROGRAM 11

Create a menu with three entries named curves, colors and quit. The entry curves has a submenu which has four entries named Limacon, Cardiod, Three-leaf and Spiral. The color menu has 8 colors of RGB model. Write a program to create the above hierarchical menu and attach appropriate services to each menu with mouse btn.

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

struct screenPt {
    int x, y
};

typedef enum {limacon=1, cardiod=2, threeleaf=3, spiral=4} curveName;
int w = 600, h = 500;
int curve = 1;
int red = 0, green = 0, blue = 0;

void myInit ( ) {
    glClearColor (1,1,1,1);
    gluMatrixMode (GL_PROJECTION);
    gluOrtho2D (0, 200, 0, 150);
}
```

Teacher's Signature : _____

Menu options



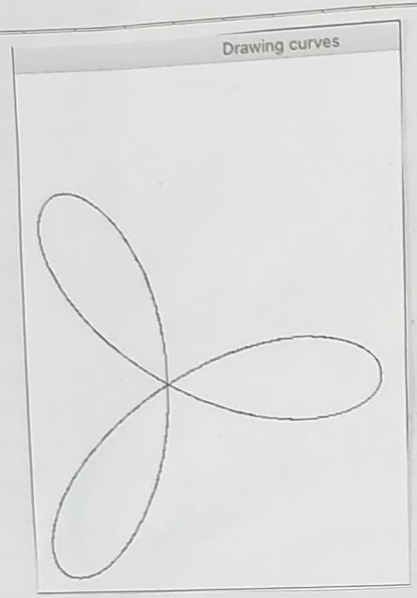
Date .....

Expt. No. .....

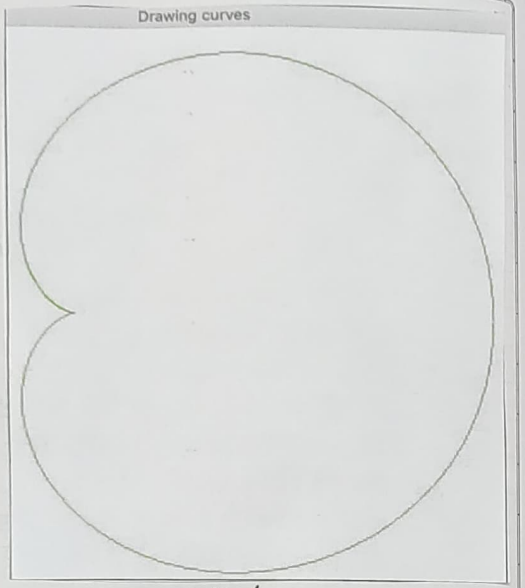
Page No. .....

```
void lineSegment (screenPt p1, screenPt p2) {
    glLineWidth (3.0);
    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);
    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 += 2 * a; break;
        case threeleaf : curvePt[0].x += a; break;
        case spiral : break;
        default : break;
    }
    theta = dtheta;
```

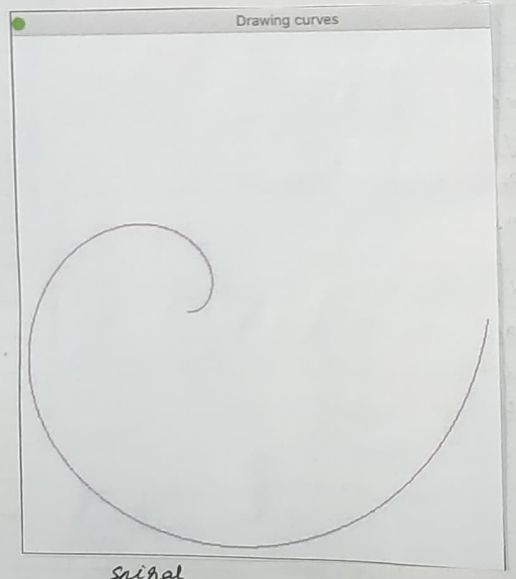
Teacher's Signature : _____



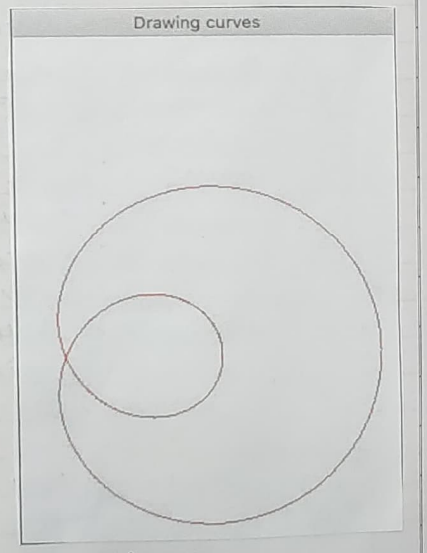
3 leaf



cardiod



spiral



limacon

Date

Expt. No. Page No.

```
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;
  }
}
void colorMenu (int id) {
    switch (id) {
        case 0 : break;
        case 1 : red = 0; ~~blue~~ 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);
}
```

Teacher's Signature :

Date .....

Expt. No. ..... Page No. .....

```
void main_menu (int id) {
    switch (id) {
        case 3: exit(0);
        }
    }
void mydisplay () {}
void myreshape (int nw, int nh) {
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrtho2D (0,(double)nw, 0, (double) nh);
    glClear (GL_COLOR_BUFFER_BIT);
}
int 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 ("Cardiod",2);
     glutAddMenuEntry("Threeleaf", 3);
     glutAddMenuEntry ("spiral", 4);
     glutAttachMenu (GLUT_LEFT_BUTTON);

    int colorId = glutCreateMenu (colorsMenu);
     glutAddMenuEntry ("Red", 4);
     glutAddMenuEntry ("Green", 2);
```

Teacher's Signature : _____

Date .....

Expt. No. .....

Page No. .....

```
glutAddMenuEntry ("Black",0);
glutAddMenuEntry ("Blue",1);
glutAddMenuEntry ("Yellow",6);
glutAddMenuEntry ("Cyan",3);
glutAddMenuEntry ("Magenta",5);
glutAddMenuEntry ("White",7);
glutAttachMenu (GLUT_LEFT_BUTTON);

glutCreateMenu (main_menu);
glutAddSubMenu ("drawCurve", curveId);
glutAddSubMenu ("colors", colorId);
glutAddMenuEntry ("quit", 3);
glutAttachMenu (GLUT_LEFT_BUTTON);

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

Teacher's Signature : _____

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

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

using namespace std;

float x1[4], ye[4];
int flag = 0,point_count=0;

void myInit() {
     glClearColor(1, 1, 1, 1);
     glColor3f(1, 1, 1);
     glPointSize(5);
     gluOrtho2D(0, 500, 0, 500);
}

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

void display() {
     glClear(GL_COLOR_BUFFER_BIT);
     int i;
     double t;
     glColor3f(0, 0, 0);
     glBegin(GL_POINTS);
     for (t = 0; t < 1; t = t + 0.005) {
          double xt = pow(1-t,3)*x1[0]+3*t*pow(1-t,2)*x1[1]
                      + 3 * pow(t, 2)*(1 - t)*x1[2] + pow(t, 3)*x1[3];
          double yt = pow(1 - t, 3)*ye[0] + 3 * t * pow(1 - t, 2) * ye[1]
                     +3*pow(t, 2) * (1 - t) * ye[2] + pow(t, 3) * ye[3];
          glVertex2f(xt, yt);
     }
     glColor3f(1, 1, 0);
     for (i = 0; i < 4; i++)
          glVertex2f(x1[i], ye[i]);
     glEnd();
     glFlush();
}
```


Date .....

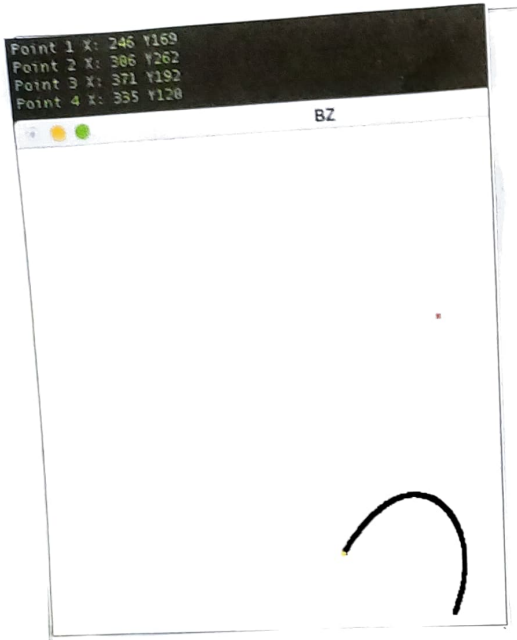
Expt. No. 12 Page No. .....

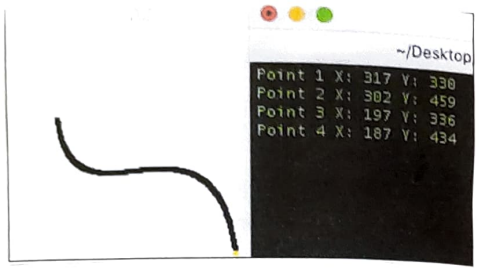
PROGRAM 12

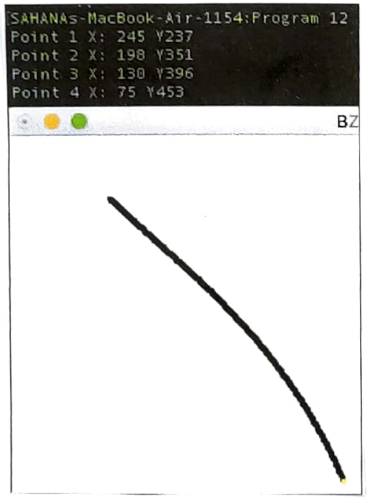
write a program to construct Bezier curve. control points are supplied through keyboard or mouse.

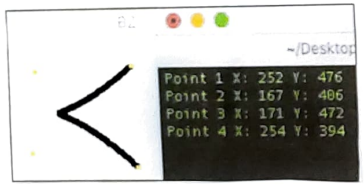
```
#include <iostream>
#include <math.h>
#include <glut/glut.h>

using namespace std;

float x1[4], ye[4];
int flag = 0, point_count = 0;

void myInit () {
     glClearColor (1,1,1,1);
     glColor3f (1,1,1);
     glpoint Size[5];
     gluOrtho 2D(0, 500, 0, 500);
}

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

Teacher's Signature : _____









Date

Expt. No. Page No.

```
void display() {
    glClear (GL_COLOR_BUFFER_BIT);
    int i; double t;
    glColor3f (0,0,0);
    glBegin (GL_POINTS);
    for (t=0; t<1; t+=0.005){
        double xt = pow (1-t,3) * x1[0] + 3 * t * pow(1-t,2)
                * x1[1] + 3 * pow (t,2)*(1-t)*x1[2] +
                    pow (t,3) * x1[3];
        double yt = pow (1-t,3) *yc[0]+3*t*pow(1-t, 2) * yc[1]
                + 3 *pow (t,2) *(1-t) * yc[2]+pow(t,3)*yc[3];
    }
    glColor3f (1, 1, 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<< "Point "<< ++point_count << " X: " <<x<< " Y: "<<
                    500 -y << endl;
        glPointSize (4);
        glColor3f (1,0,0);
        glBegin (GL_POINTS);
            glVertex2i(x, 500 -y);
        glEnd();
```

Teacher's Signature :

```
        glFlush();
        flag++;
    }
    if(flag>=4 && btn==GLUT_LEFT_BUTTON){
        glColor3f(0,0,1);
        display();
        flag=0;
        point_count=0;
    }
}

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