Date _____

Expt. No. 01 Page No. 01

Q) Write a program to generate a line using Brushnam's line drawing technique.

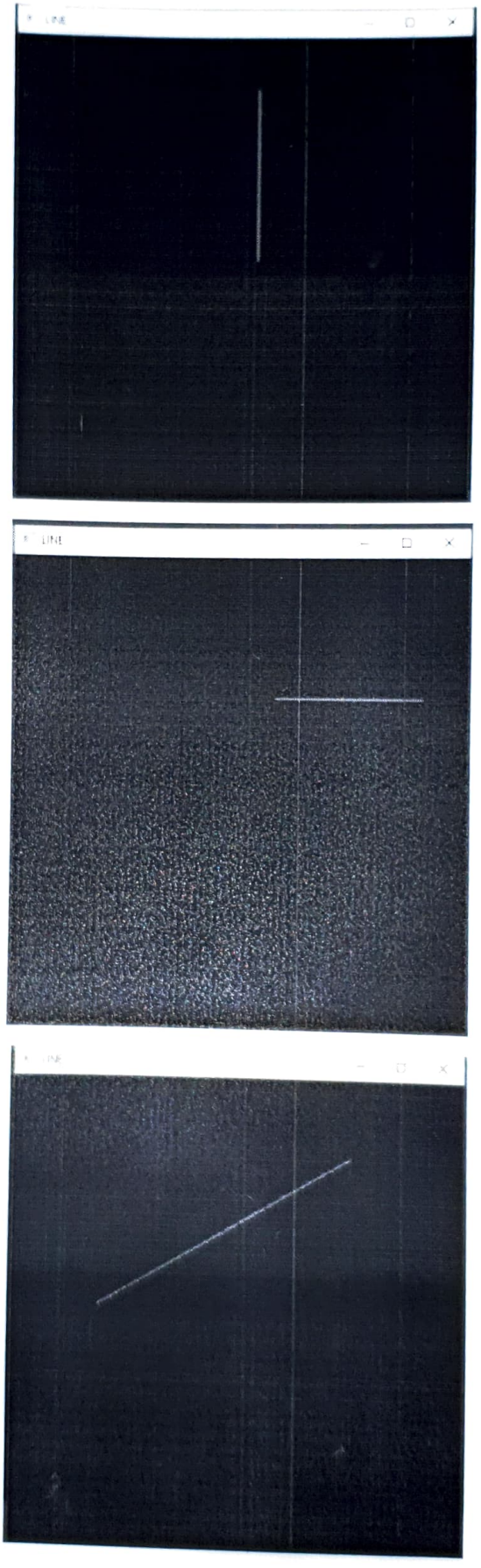
```
#include <iostream>
#include <gl/glut.h>
#include <time.h>
using namespace std;

int x1, x2, yc1, yc2;
int flag = 0;

void draw_pixel (int x, int y)
{
    glColor3f (1,0,0);
    glBegin (GL_POINTS);
    glVertex2i (x,y);
    glEnd();
    glFlush();
}
void draw_line ()
{
    int dx, dy, i, e;
    int incx, incy, inc1, inc2;
    int x,y;
    dx = x2-x1;
    dy = y2-yc1;
    if (dx<0) dx = -dx
    if (dy<0) dy = -dy;
    incx=1;
```

Teacher's Signature _____

```
if (x2 < xc1)
    incx = -1;
incy = 1;
 if (y2 < yc1)
   incy = -1;
x = x1;
y = yt1;
 if (dx > dy)
  {
     draw_pixel (x, y);
      e = 2*dy - dx;
      inc1 = 2*(dy-dx);
      inc2 = 2*dy;
     for (i = 0; i < dx ; i++)
      { if (e > 0)
           { y += incy;
             e += inc1;
           }
       else { e += inc2; }
       x += incx;
       draw_pixel (x, y);
     }
}
   else {
          draw_pixel (x, y);
           e = 2*dx - dy;
            x += incx ; inc1 = 2*(dx - dy);
            inc2 = 2*dx;
```

```
for (i=0; i<dy; i++)
    { if (e>0)
      { x+ = incx;
        e+ = inc1;
      }
    else
        e+ = inc2;
    y+ = incy;
    draw-pixel (x,y);
  }
}
  glFlush();
}
  void myinit ()
  { glClear (GL-COLOR-BUFFER-BIT);
    glClearColor (1,1,1,1);
    gluOrtho2D (-250, 250, -250, 250);
  }
  void MyMouse (int button, int state, int x, int y)
{
  Switch (button)
    {
      case GLUT-LEFT-BUTTON:
      if (state == GLUT_DOWN)
      { if (flag ==0) { printf ("Defining x1, y1");
        x1 = x-250;
        y<1 = 250-y;
        flag++;   cout <<x1<<" "<<y<1<<"\n";
      }
```

```
else
 { printf ("Defining x2 y2");
    x2 = x - 250;
    y2 = 250 - y;
   flag = 0;
    cout << x2 << " " << y2 << "\n";
    draw_line ();
 }
}
break;
 }}
void display()
 { }
 int main ( int ac, char * arg[] ),
   {
     glut init (&ac, av);
     glut InitDisplayMode ( GLUT_SINGLE | GLUT_RGB);
     glutInitWindowSize (500, 500);
      glut InitWindowPosition (100, 200);
       glut CreateWindow ("Line");
     myinit ();
       glut MouseFunc (MyMouse);
     glut DisplayFunc (display);
        glut MainLoop();
 }.
```



Program1: Line

Date _____

Expt. No. 02 Page No. 05

Q.) Write a program to generate a circle and ellipse using bresenham's circle and ellipse drawing technique.

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

using namespace std;

int xc, yc, r, rx, ry, xce, yce;
void draw_circle (int xc, int yc, int x, int y)
{ glBegin (GL_POINTS);
 glVertex2i(xc+x, yc+y); glVertex2i(xc-x, yc+y);
 glVertex2i (xc+x, yc-y); glVertex2i (xc-x, yc-y);
 glVertex2i (xc+y, yc+x); glVertex2i (xc-y, yc+x);
 glVertex2i (xc+y, yc-x); glVertex2i (xc-y, yc-x);
 glEnd();
}
void circlebres()
  { glClear (GL_COLOR_BUFFER_BIT);
 int x=0, y=r, d=3-2*r;
   while (x<=y)
    { draw_circle (xc, yc, x, y); x++;
        if (d<0)
            d = d+4*x+6;
      else { y--;
            d = d+4*(x-y)+10; }
```

Teacher's Signature _____

```
draw-circle (xc, yc, x, y); }
 glFlush(); }
int p1_x, p2-x, p1-y, p2-y, point1-done = 0;

void myMouseFunc(int button, int state, int x, int y)
 {
  if(button == GLUT-LEFT-BUTION && State == GLUT-DOWN &&
   point1-done == 0)
    { p1-x = x-250;
      p1-y = 250-y; point1-done = 1; }
else if (button == GLUT-LEFT-BUTTON && State == GLUT-DOWN)
  { p2-x = x-250; p2-y = 250-y;
     xc = p1-x; yc = p1-y;
 float exp = (p2-x-p1-x)*(p2-x-p1-x) + (p2-y-p1-y)*
          (p2-y-p1-y);
   r = (int) (sqrt(exp));
   circleBres();
   point1-done = 0;
 } }
/// Ellipse ///
void draw-ellipse (int xce, int yce, int x, int y)
 { glBegin(GL-POINTS);
 glVertex2i(x+xce, y+yce); glVertex2i(-x+xce, y+yce);
  glVertex2i(x+xce, -y+yce); glVertex2i(-x+xce, -y+yce);
  glEnd(); }
void midptellipse()
    { glClear(GL-COLOR-BUFFER-BIT);
  float dx, dy, d1, d2, x, y;
```

Teacher's Signature

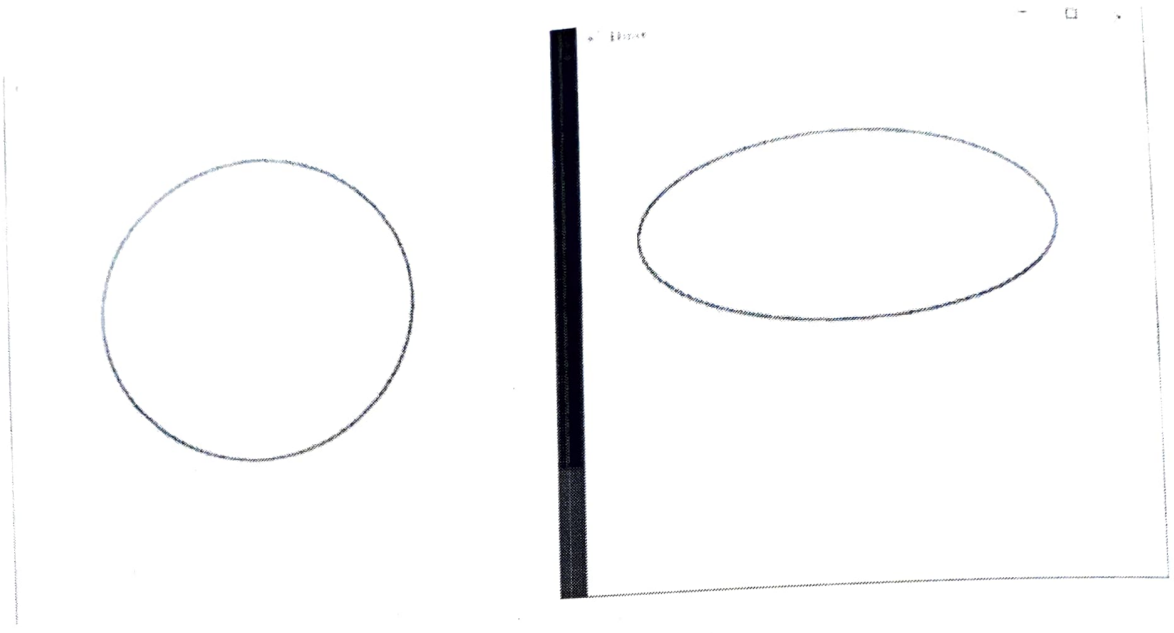
```
x=0; y= int (ry);
d1= (ry*ry) - (rx*rx*ry) + (0.25 *rx * rx);
dx = 2*ry * ry * x;
dy = 2* rx * rx * y;

while (dx<dy)
  {  draw-ellipse (xce, yce, x, y);
  if (d1<0) {
        x++;
        dx = dx + (2*ry * ry);
        d1 = d1 + dx + (ry * ry); }
  else {  x++; y--;
     dx = dx +(2 * ry * ry);
     dy = dy - (2*rx * rx);
      d1 = d1+dx-dy+(ry * ry);  } }

d2 = ((ry * ry)* ((x+0.5) * (x+0.5)))+
     ((rx * rx) * ((y-1) * (y-1))) - (rx*rx*ry*ry);

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

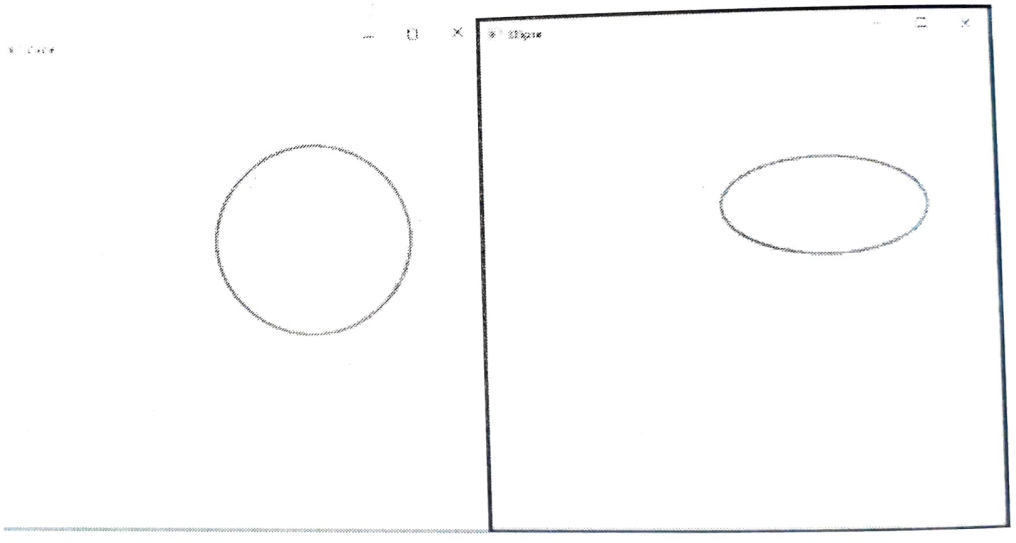
Teacher's Signature

```
int p1e-x, p2e-x, p1e-y, p2e-y, p3e-x, p3e-y, point1e-done=0;

void myMouseFunc(int button, int state, int x, int y)
{
 if (button== GLUT-LEFT-BUTTON && state == GLUT-DOWN &&
     point1e-done == 0)
  { p1e-x = x-250; p1e_y = 250-y;
    xce = p1e-x; yce = p1e_y; point1e-done=1; }

else if (button== GLUT-LEFT-BUTTON && state == GLUT-DOWN &&
      point1e_done == 1) {
   p2e-x = x-250;  p2e-y = 250-y;
 float exp = (p2e-x-p1e-x) * p2e-x-p1e-x) +
    (p2e-y-p1e-y) * (p2e-y-p1e-y);
   rx = (int) (sqrt(exp)); point1e_done = 2; }

else if (button== GLUT-LEFT-BUTTON && state == GLUT-DOWN
  && point1e-done == 2 ) {
     p3e-x = x-250; p3e-y = 250-y;
 float exp = (p3e-x-p1e-x) * (p3e-x-p1e-x) +
        (p3e-y-p1e-y) * (p3e-y-p1e-y);
    ry = (int) (sqrt(exp));
     midtellipse();
     point1e_done = 0; } }
     void myDrawing() { }
     void myDraning() { }
    void myinit() {
 glClearColor(1,1,1,1); glColor3f(1.0,0.0,0.0);
 glPointSize(3.0); gluOrtho2D(-250,250,-250,250); }
```

Teacher's Signature

```
void main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);

cout << "creating circle and ellipse on mouseclick \n";
    int id1 = glutCreateWindow("Circle");
    glutSetWindow(id1);
    glutMouseFunc(myMouseFuncCircle);
    glutDisplayFunc(myDrawingC);
    minit();
    glutInitWindowSize(500,500);
    glutInitWindowPosition(600,100);

    int id2 = glutCreateWindow("Ellipse");
    glutSetWindow(id2);
    glutMouseFunc(myMouseFunc);
    glutDisplayFunc(myDrawing);
    minit();
    glutMainLoop();
}
```





Program2: Circle and Ellipse

Date

Expt. No. 03 Page No. 10

8) Write a program to recursively subdivide a tetrahedron to form 3-D gasket Serpinski.

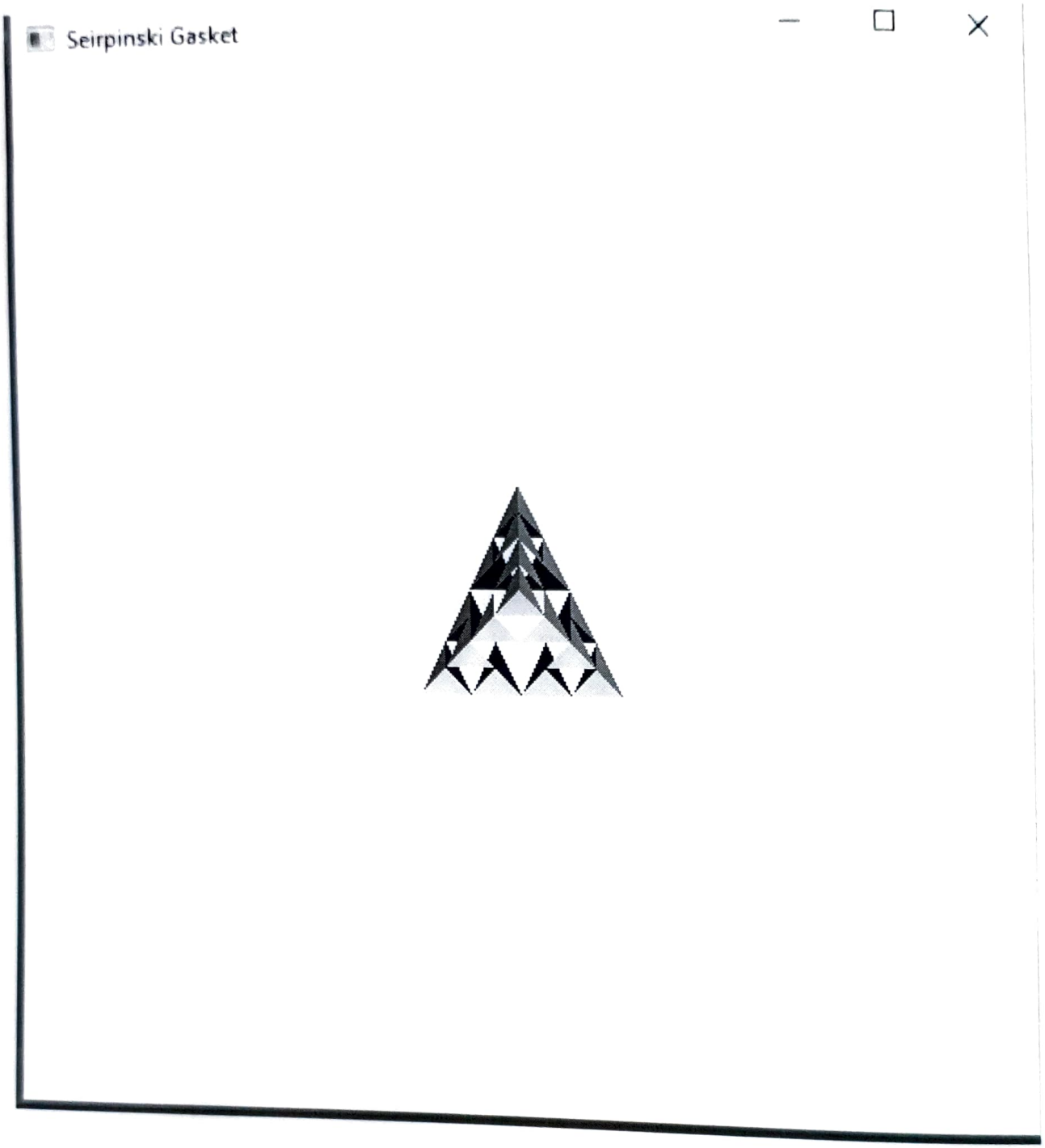
```
#include <gl/glut.h>
#include <stdio.h>

int m;
typedef float point[3];
point tetra[4] = {{0,100,-100}, {0,0,100}, {100,-100,-100},
                  {-100,-100,-100}};
void tetrahedron(void);
void myinit(void);
void divide_triangle(point a, point b, point c, int m);
void draw_triangle(point p1, point p2, point p3);
int main(int argv, char** argc)
{ printf("Enter the Number of iterations:");
  scanf_s("%d", &m);
  glutInit(&argv, argc);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB |
                      GLUT_DEPTH);
  glutInitWindowPosition(100,200);
  glutInitWindowSize(500,500);
  glutCreateWindow("Serpinski Gasket");
  glutDisplayFunc(tetrahedron);
  glEnable(GL_DEPTH_TEST);
  myinit();
  glutMainLoop(); }
```

Teacher's Signature

```
void divide_triangle (point a, point b, point c, int m)
{ point v1, v2, v3;
  int j;
  if (m>0) {
  for (j=0; j<3; j++)
        v1[j] = (a[j] + b[j])/2;
  for (j=0; j<3; j++)
        v2[j] = (a[j] + c[j])/2;
  for (j=0; j<3; j++)
        v3[j] = (b[j] + c[j])/2;

    divide_triangle (a, v1, v2, m-1);
    divide_triangle (c, v2, v3, m-1);
    divide_triangle (b, v3, v1, m-1);
  } else draw_triangle (a,b,c); }
void myinit () {
  glClearColor (1,1,1,1); glOrtho(-500.0, 500.0,
                                  -500.0, 500.0); }

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, 0.1);
  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);
```

Teacher's Signature

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



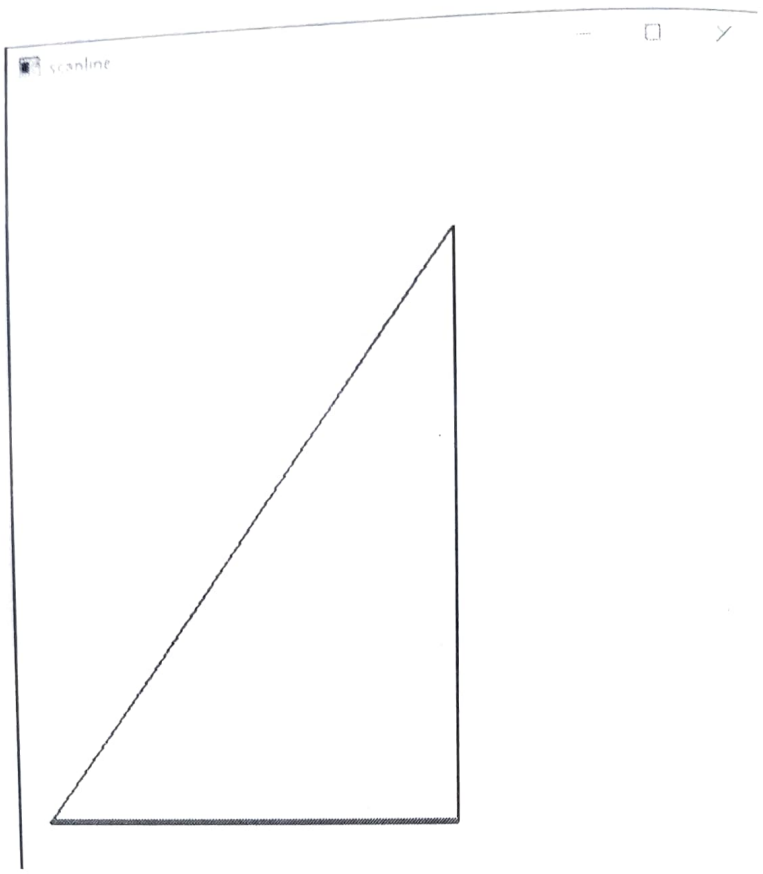
Program 3: Seirpinski Gasket

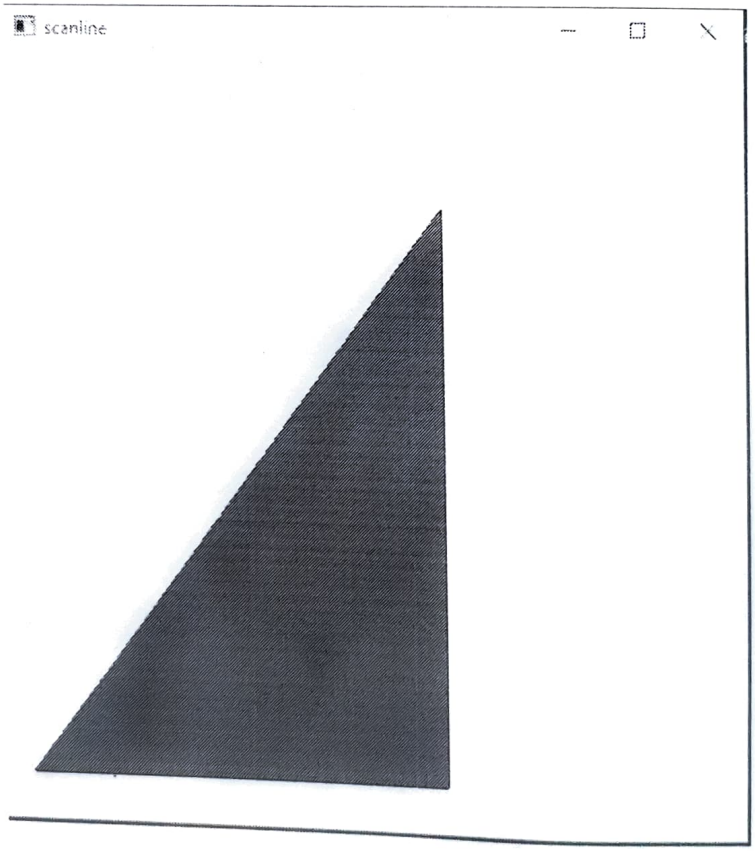
Q) Write a Program to fill any given polygon using Scan-line area filling algorithm.

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

void draw_line (float x1, float y1, float x2, float y2)
 { Sleep(100);
   glColor3f(1,0,0);
   glBegin (GL_LINES); glVertex2f (x1,y1);
   glVertex2f (x2,y2); glEnd();
   glFlush();
}

  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); }

void scanfill (~~int~~ 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 (int x, (intx+m));
    if (m>=2)
      for (int i = 0; i<m; i=i+2)
       draw_line (intx[i], s1, intx[i+1], s1);
    } }
 void display_filled_polygon() {
  glClear (GL_COLOR_BUFFER_BIT);
  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); }
```

```
void main (int ac , char * av[]) {
  glut Init (&ac, av);
   printf ("Enter no. of sides : \n");
   cin >> n;
printf ("Enter coordinates of endpoints : \n");
  for(int i=0; i<n; i++)
      {
          printf ("X-coord Y-coord for %d vertex : \n",
   cin >> x[i] >> y[i];                          i+1);
}
   glut Init Display Mode (GLUT-SINGLE | GLUT-RGB);
   glut Init Window Size (500, 500);
    glut Init Window Position (0,0);
     glut Creat Window ("scan line");
  glut Display Func (display_field_polygon);
     my Init-();
      glut Main Loop();
}.
```





Program4: Scan fill

Date _____

Expt. No. 05 Page No. 16

Q) Write a program to create a house like figure & perform the operations.

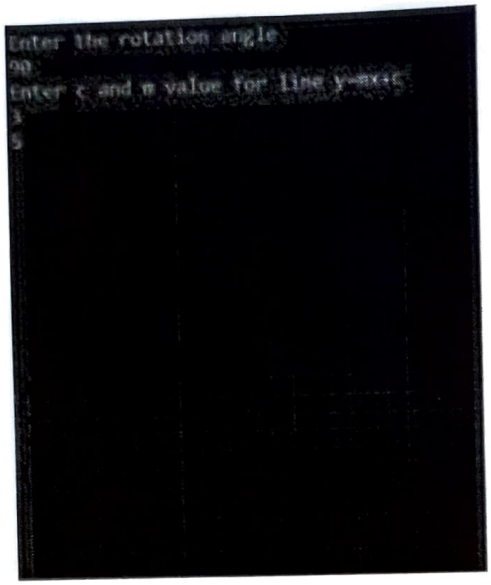
```
#include <gl/glut.h>
#include <math.h>
#include <iostream>
#include <stdio.h>
using namespace std;

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

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);
   glRotatef(angle,0,0,1);
   glTranslatef(-100,-100,0);
   glColor3f(1,1,0);
    glBegin(GL_LINE_LOOP);
```

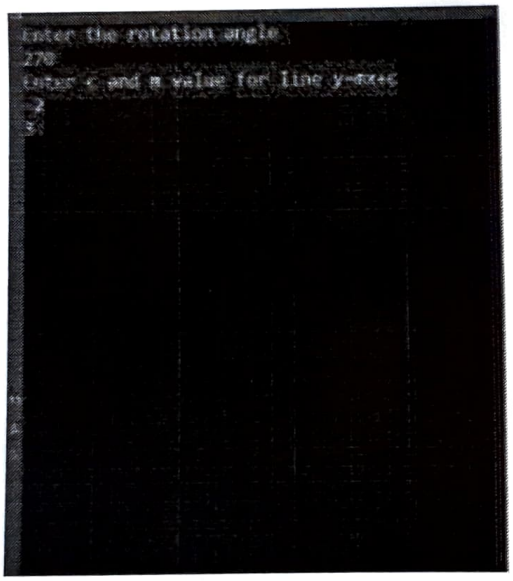
Teacher's Signature _____

```
for(int i=0; i<11; i++) glVertex2fv(house[i]);
glEnd(); glPopMatrix();
glFlush(); }
void display2(){
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity(); glOrtho2D(-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, y1=m*x1+c, y2=m*x2+c;
glColor3f(1,1,0);
glBegin(GL_LINES);
glVertex2f(x1,y1); glVertex2f(x2,y2);
glEnd(); glFlush();
glPushMatrix();
glTranslatef(0,c,0);
theta = atan(m);
theta = theta*180/3.14;
glRotatef(theta,0,0,1); glScalef(1,-1,1);
glRotatef(0,-c,0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++) glVertex2f(house[i]);
glEnd(); glPopMatrix(); glFlush(); }
```

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

void mouse (int btn, int stati , intx , int y) {
 if (btn == GLUT-LEFT-BUTTON && state == GLUT-DOW) {
    display (); }
else if (btn == GLUT-RIGHT-BUTTON  && state == GLUT-DOWN)
     { display2 (); } }

void main (int argc, char ** argv)
 { printf ("Enter the rotational angle \n");
    cin >>angle;
  printf ("Enter c and m value for line y = mx+c \n");
    cin >> c >> m;
 glutInit (&argc, argv);
 glutInitDisplay Mode (GLUT-SINGLE | GLUT-RGB);
    glutInitWindowSize (900,900);
  glutInitWindowPosition (100,100);
   glutCreatWindow ("House");
    glutDisplayFunc (display);
    glutMouse func (mouse);
    myInit ();
    glutMainLoop ();
}
```

Teacher's Signature _____





Program5: House Rotation

Date _____

Expt. No. 06 Page No. 19

8) Write a program to implement CohenSutherland line Clipping Algorithm.

```
#include<stdio.h>
#include<gl/glut.h>
#define outcode int
#define true 1
#define false 0
double xmin, ymin, xmax, ymax, xvmin, yvmin, xvmax, yvmax;
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, x2; y2; ; };
struct line_segment ls[10];
outcode computeoutcode (double x, double y)
{ outcode code=0;
  if (y>ymax) code.1=TOP;
  else if (y<ymin) code.1= BOTTOM;
 if (x>xmax) code.1 = RIGHT;
 else if (x<xmin) code.1 = LEFT;
  return code; }

void cohensuther (double x0, double y0, double x1, double y1)
{ outcode outcode0, outcode1, &outcodeout;
 bool accept = false, done = false;
 outcode 0= compute out code (x0, y0);
 outcode 1 = computeoutcode (x1, y1);
 do{ if (!(outcode0 | outcode1))
    { accept= true; done = true; }
```

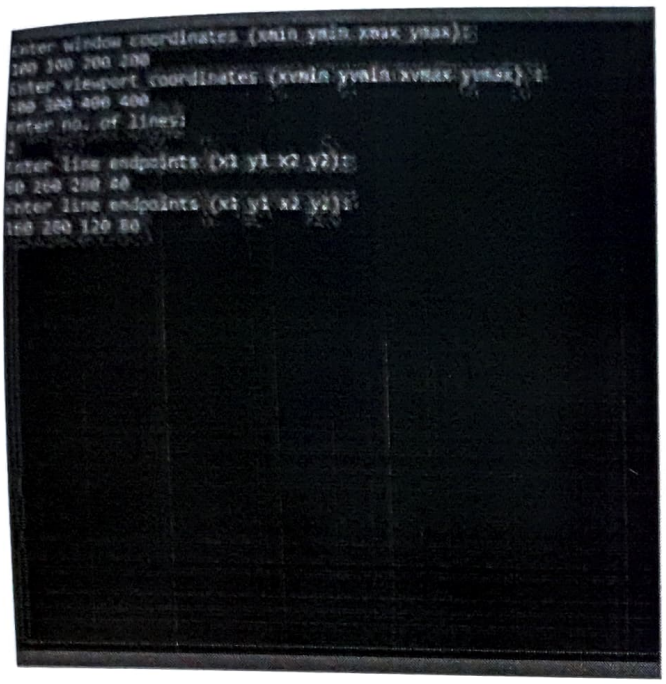
Teacher's Signature _____

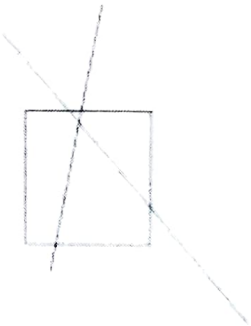
```
else if (outcode0 & outcode1)  done = true;
else { double x, y;
outcodeOut = outcode0 ? outcode0 : outcode1;
if (outcodeOut & TOP)
  { x = x0 + (x1-x0) * (ymax-y0) / (y1-y0);
    y = ymax; }
else if (outcodeOut & BOTTOM)
  { x = x0 + (x1-x0) * (ymin-y0) / (y1-y0);
    y = ymin; }
else if (outcodeOut & RIGHT)
{ y = y0 + (y1-y0) * (xmax-x0) / (x1-x0);
  x = xmax; }
else { y = y0 + (y1-y0) * (xmin-x0) / (x1-x0);
    x = xmin; }
if (outcodeOut == outcode0)
    { x0 = x; y0 = y; 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); glBegin(GL_LINE_LOOP); glVertex2f(xvmin, yvmin);
glVertex2f(xvmax, yvmax); glVertex2f(xvmin, yvmax);
glEnd(); glColor3f(0,0,1); glBegin(GL_LINES);
glVertex2d(vx0, vy0); glVertex2d(vx1, vy1); glEnd(); }}
```

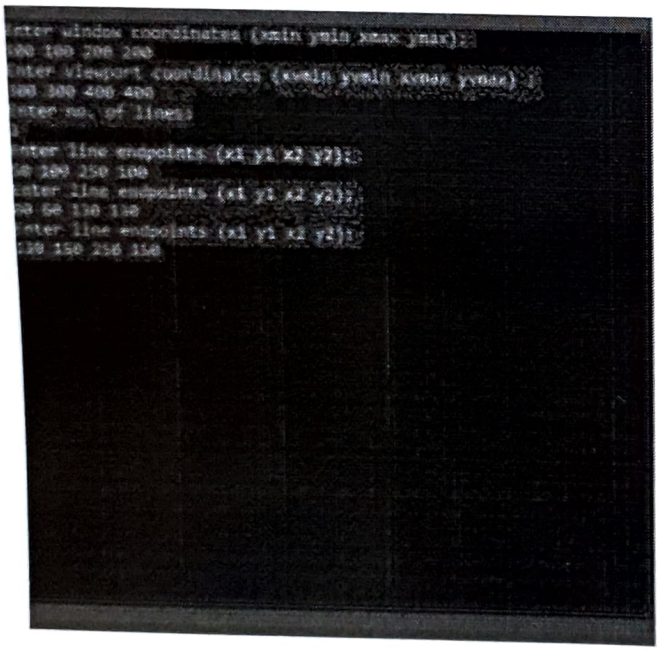
Teacher's Signature _____

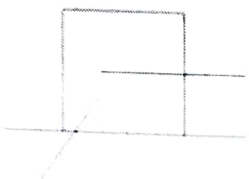
```
void display() {
  glClear (GL_COLOR_BUFFER_BIT); glColor3f (0,0,1);
  glBegin (GL_LINE_LOOP); glVertex2f (xmin, ymin);
   glVertex2f (xmax, ymin); glVertex2f (xmax, ymax);
   glVertex2f (xmin, ymax); glEnd();
   for (int i=0; i<n; i++) {
    glBegin (GL_LINES); glVertex2d (ls[i].x1, ls[i].y1);
    glVertex2d (ls[i].x2, ls[i].y2); glEnd(); }
 for (int i=0; i<n ; i++) cohensutherl (ls[i].x1, ls[i].x2,
   glFlush(); }                                ls[i].y2);
void init() { }
void main (int argc, char **argv)
{ printf-s("Window's coordinate");
  scanf-s("%lf %lf %lf %lf", &xmin, &ymin, &xmax, &ymax);
  printf ("Viewport coordinates");
  scanf-s("%lf %lf %lf %lf", &xmin, &ymin, &xmax, &ymax);
  printf ("Enter No. of lines:");
   scanf-s("%d", &n);
  for (int i=0; i<n; i++)
  { printf ("Enter line endpoints");
  scanf-s("%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);
    myinit();
     glutMainLoop();
}
```

Teacher's Signature _____









Program6: CohenSutherLand

Date

Expt. No. 07 Page No. 22

8) Write a program to implement Liang-Barsky line clipping Algorithm.

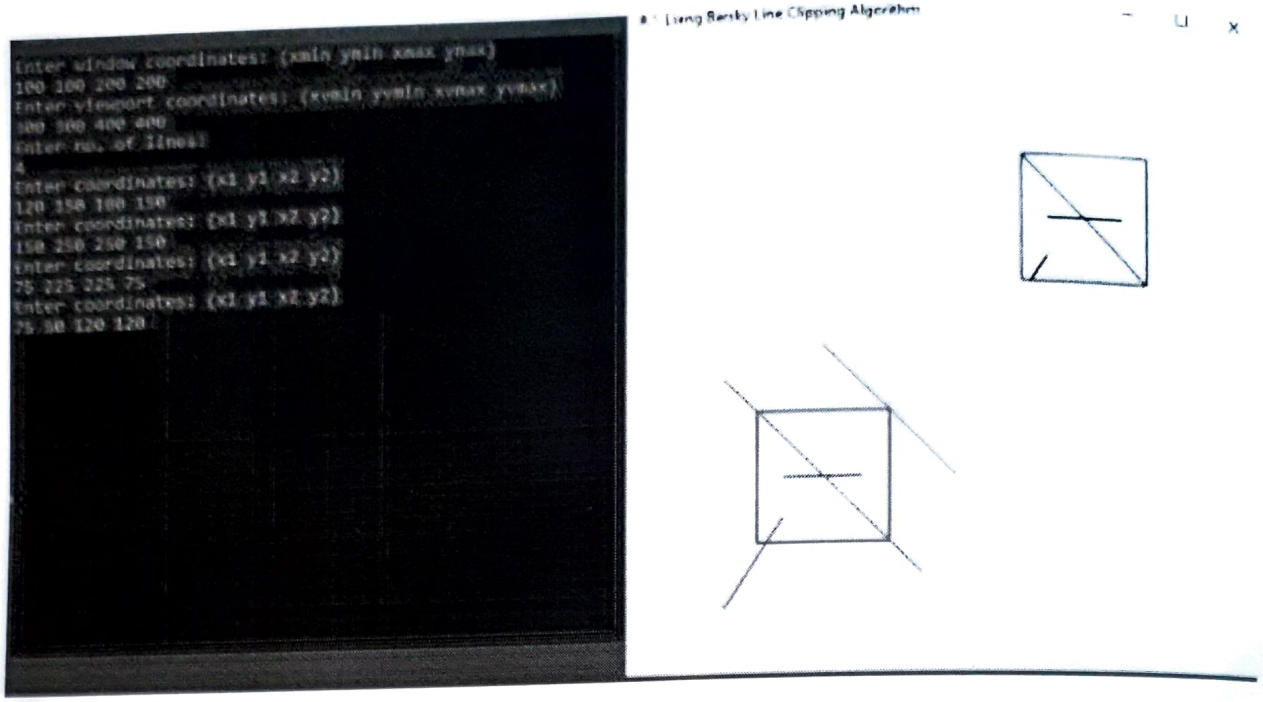
```
#include <gl/glut.h> #include<stdio.h>
double xmin, ymin, xmax, ymax, xvmin, xvmax, yvmin, yvmax;
int n;
struct line_segment{ int x1, x2, y1, y2; };
struct line_segment ls[10];

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); }
 else
     if (p>0.0) { if(r<*u2) *u2 = r; if(r<*u1)
      return (false); }
 else if (p==0.0) { if(q<0.0) return (false); }
 return (true); }

Void LiangBarskyLineClipAndDraw (double x0, double y0,
                                 double x1, double y1)
{ double dx = x1-x0; dy = y1-y0, u1 = 0.0, u2 = 1.0;
  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))
```

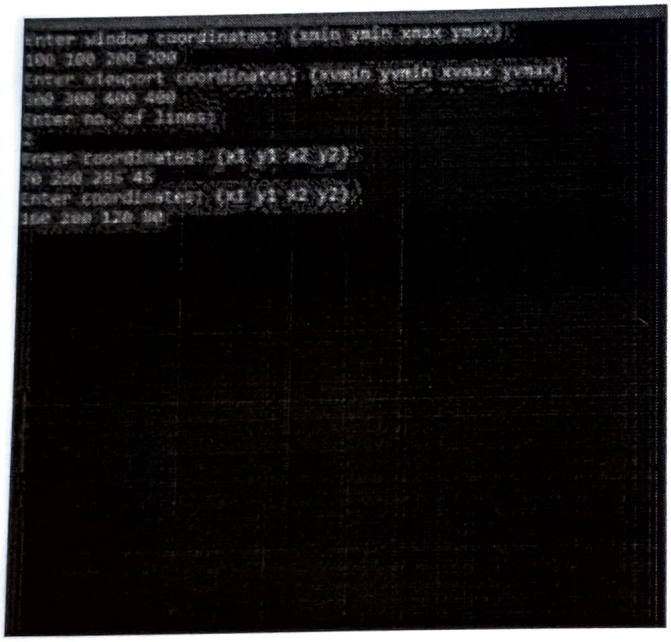
Teacher's Signature

```
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() {
 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(); for(int i=0; i<n; i++)
 LiangBarskyLineClipAndDraw(ls[i].x1, ls[i].y1,
       ls[i].x2, ls[i].y2);
  glflush(); }
   void myinit() { }
```

Teacher's Signature

```
int main(int argc, char **argv)
{ glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutInitWindowSize(500,500);
printf("Enter window coordinates");
scanf_s("%lf %lf %lf %lf",&xmin,&xymin,&xmax,&ymax);
printf("Enter viewport coordinates!");
 scanf_s("%lf %lf %lf %lf",&xvmin,&yvmin,&xvmax,&yvmax);
 printf("Enter no. of lines");
 scanf("%d",&n);
for(int i=0;i<n;++)
{ printf("Enter coordinates");
 scanf("%d %d %d %d",&ls[i].x1,&ls[i].y1,&ls[i].x2,&ls[i].y2);
 myinit();
glutMainLoop();
}.
```





Program7: LiangBarsky Line Clipping

Date _____

Expt. No. 08 Page No. 25

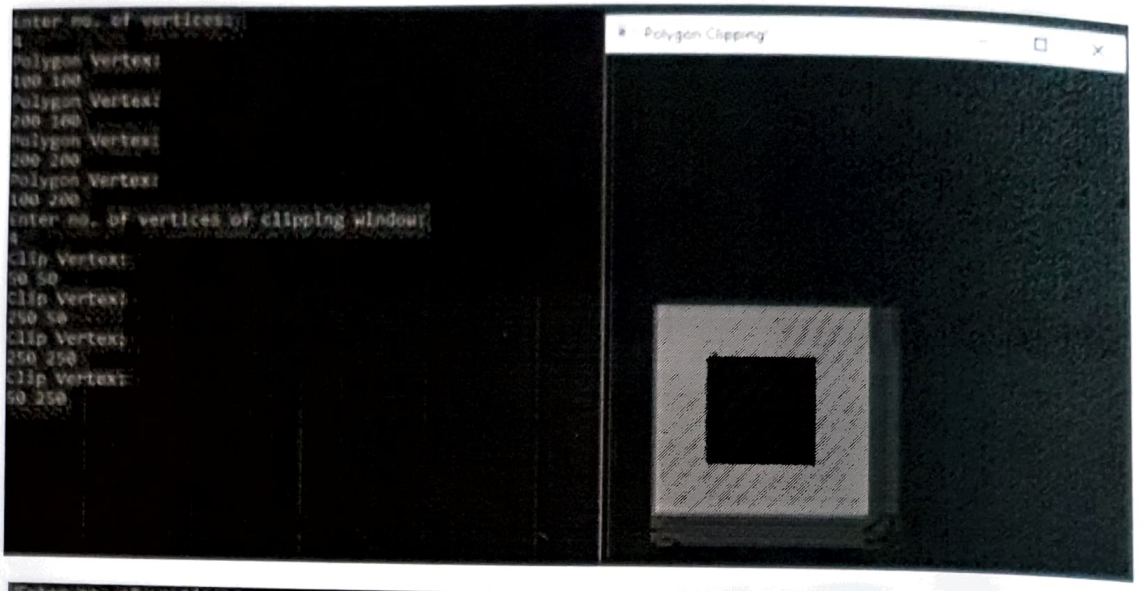
a) Write a program to implement Cohen-Hodgeman polygon clipping Algorithm.

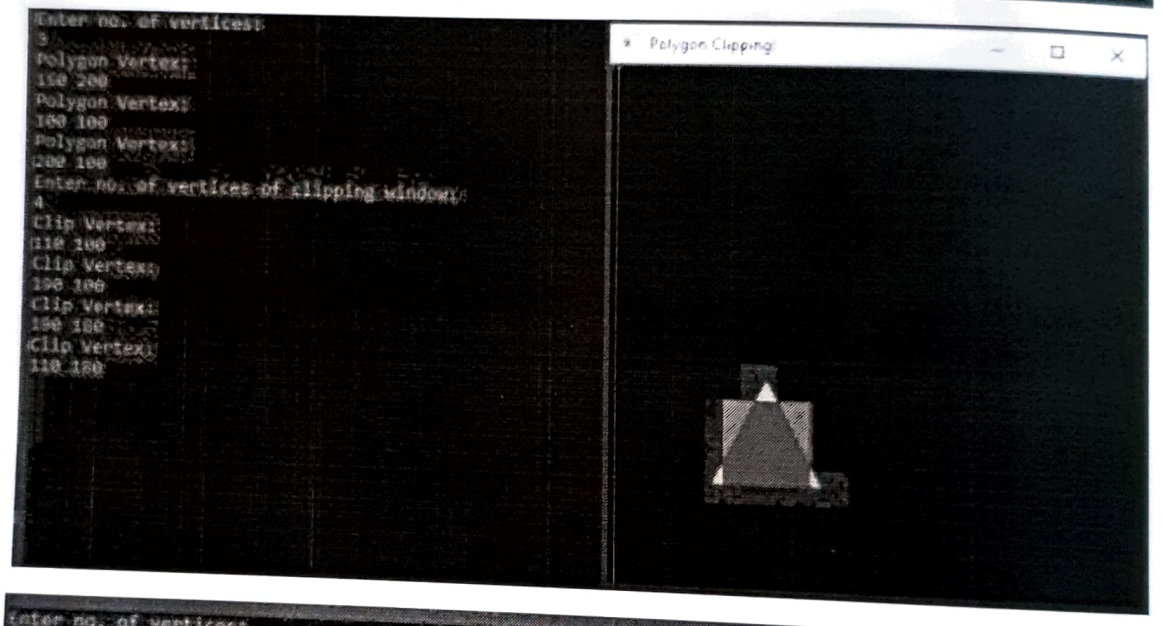
```
#include <stdio.h>
#include <gl/Glut.h>
#define using namespace std;
int poly_size, poly_points [20][2], org_poly_size,
org_poly_points [20][2], clipper_size, clipper_points [20][2];
const int MAX_POINTS = 20;

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

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

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

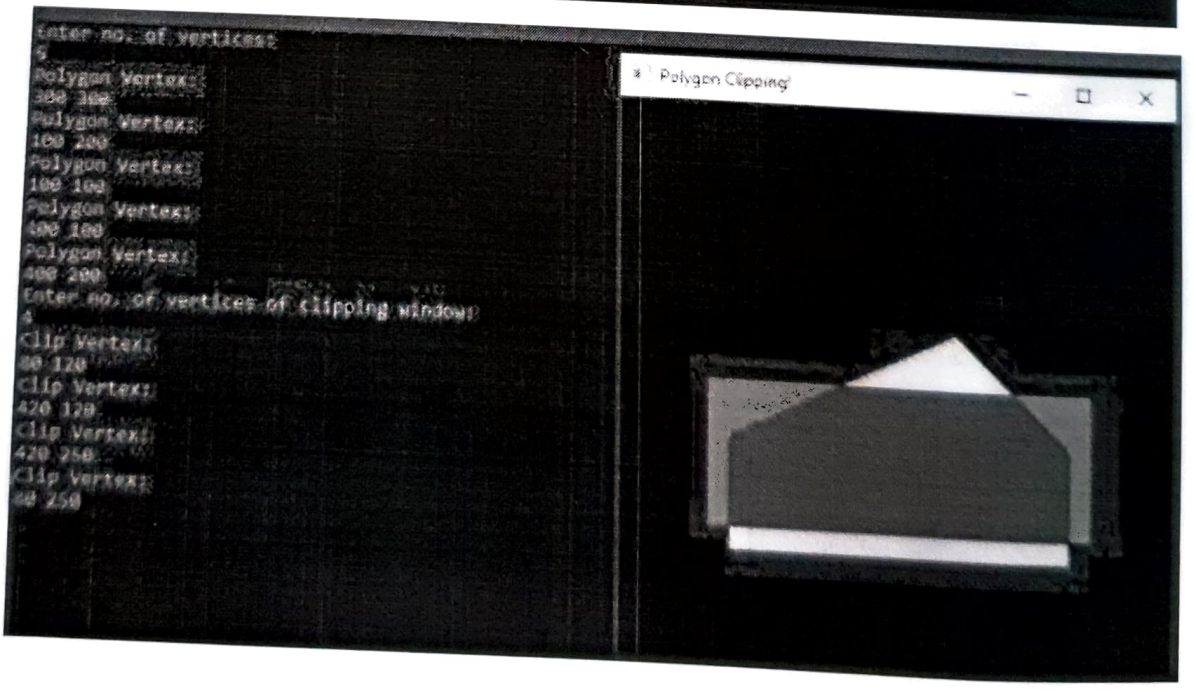
Teacher's Signature _____

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

```
void display () { init();
glColor3f (1.0f, 0.0f, 0.0f);
draw_poly (clipper_points, clipper_size);
glColor3f (0.0f, 1.0f, 0.0f);
draw_Poly ( 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];
drawPoly (poly_points, poly_size); glFlush(); }

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







Program8: Polygon Clipping

Expt. No. 09 Date

Page No. 28

9) Write a program to model a car like figure using display list and move a car.

```
#include <GL/glut.h>
#include <math.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 wheelist(){
   glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
  glColor3f(0,1,1); glutSolidSphere(10,25,25); glEndList();
   }
void mykeyboard(unsigned char key, int x, int y)
  { switch(key) {
  case't': glutPostRedisplay(); break;
  case'q': exit(0);
  default: break; } }

void movecar(float s) {
    glTranslate(s,0.0,0.0);
    glCallList(CAR); glPushMatrix();
    glTranslate(25,25,0.0);
```

Teacher's Signature

```
glCallList (WHEEL); glPopMatrix(); glPushMatrix();
glTranslatef (75, 25, 0.0); glCallList (WHEEL); glPopMatrix();
glFlush(); }

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, 500);
    glutDisplayFunc (myDisp);
    glutMouseFunc (mouse);
    glutKeyboardFunc (myKeyboard);
  glutMainLoop();
}
```

Program9: Car

Date

Expt. No. 10 Page No. 30

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

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

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

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

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() / CLOKS_PER_SEC + secs;
while((clock()/CLOCKS_PER_SEC) < end); }

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

Teacher's Signature

```
glRotatef(theta[0],1.0,0.0,0.0);
glRotatif(theta[1],0.0,1.0,0.0);
glRotatif(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();
}
void spinCube() {
    delay(0.01); theta[axis]+=2.0;
if (theta[axis]>360.0) theta[axis]-=360.0;
    glutPostRedisplay(); }

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

void myReshape(int w, int h)
 { glViewport(0,0,w,h); glMatrixMode(GL_PROJECTION);
glLoadIdentity();
    if(w<=h) glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,
          2.0*(GLfloat)h/(GLfloat)w,-10.0,10.0);
else
  glOrtho(-2.0*(GLfloat)w/(GLfloat)h, 2.0*(GLfloat)w/
   (glfloat)h, 2.0*(GL_float)w/(GL_float)h, -2.0,2.0,-10,10);
 glMatrixMode(GL_MODELVIEW); }
```

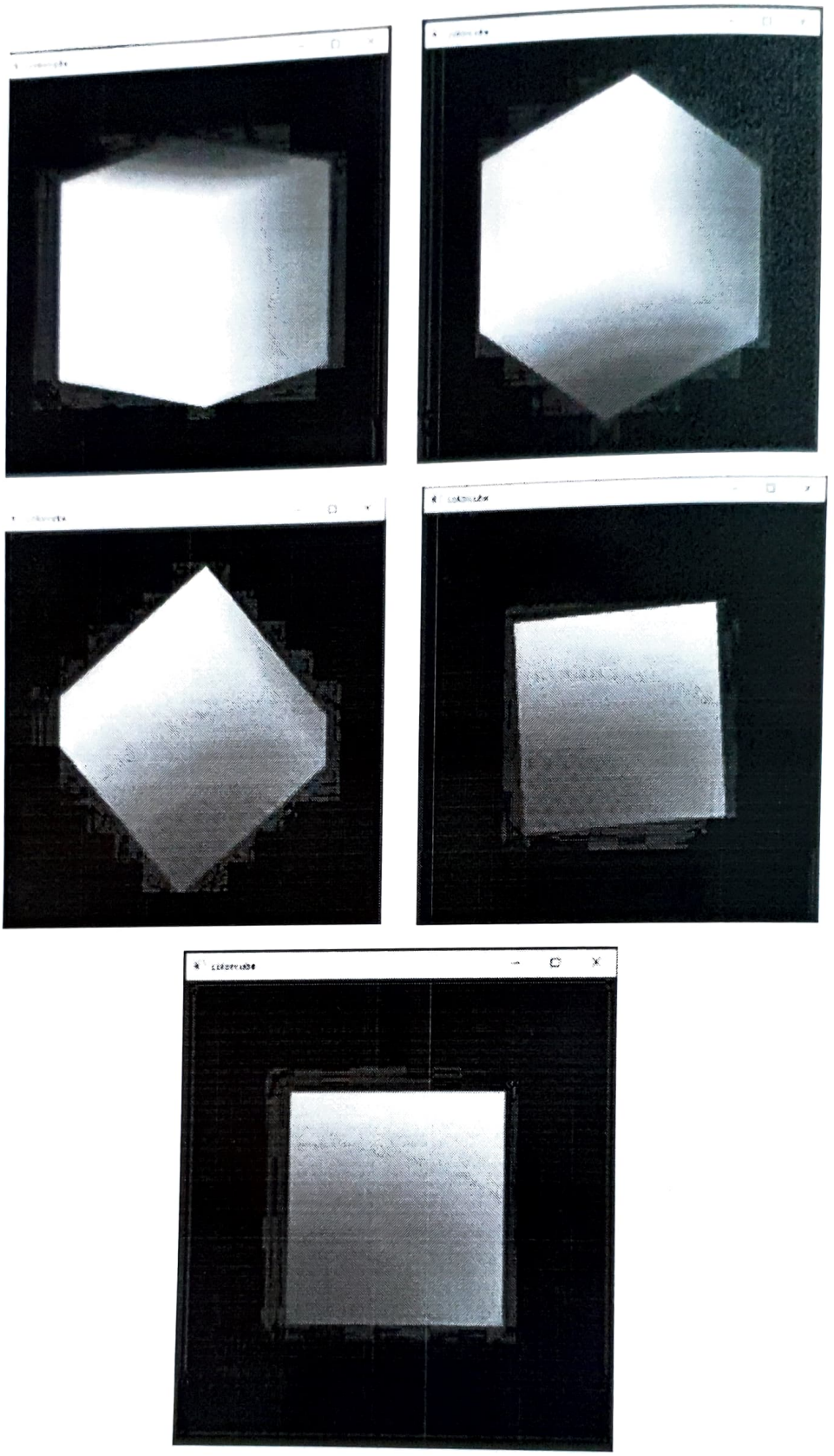
Teacher's Signature

Date _____

Expt. No. 10 Page No. 32

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

Teacher's Signature _____



Program10: Colour Cube

a) Create a menu with three entries named curve, colors and quit. The entry curve has a submenu : Limacon, Cardiod, Three leaf and Spiral. The color has submenu with all 8 colours of RGB color model.

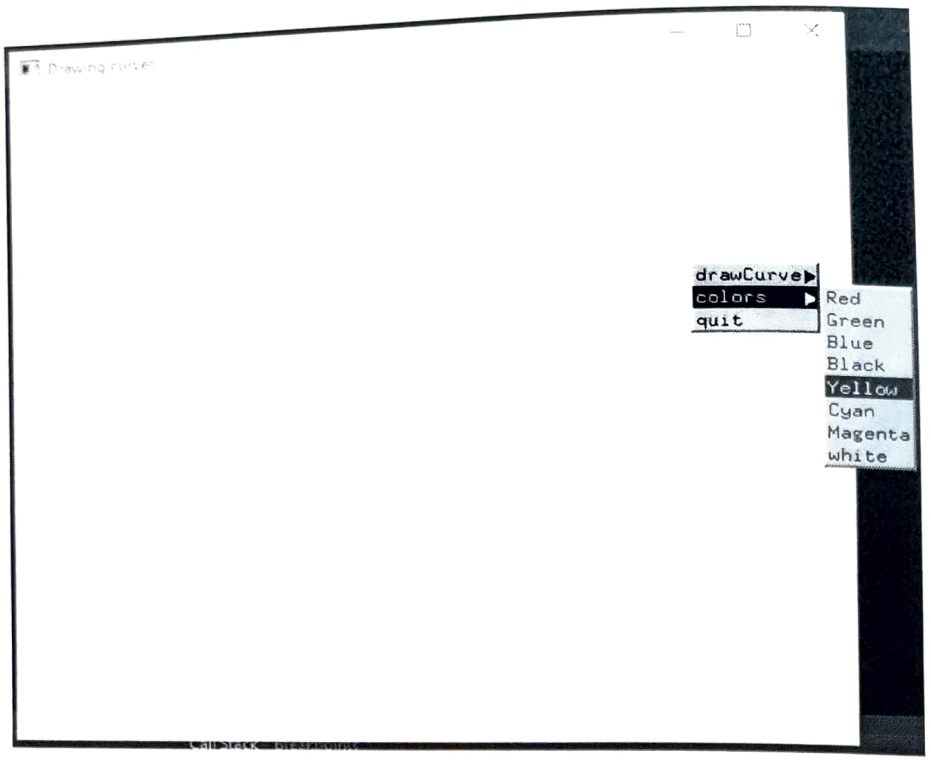
```
#include <gl/glut.h>
#include <math.h>
#include <stdio.h>
Struct screenPt {
        int x, y; };
typedef enum{limacon = 1, cardioid = 2, threeleaf = 3, spiral = 4}
        curveName;
int w=600, h=500,curve=1, red=0, green=0, blue=0;

void myinit(void) { }

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);
 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;
```

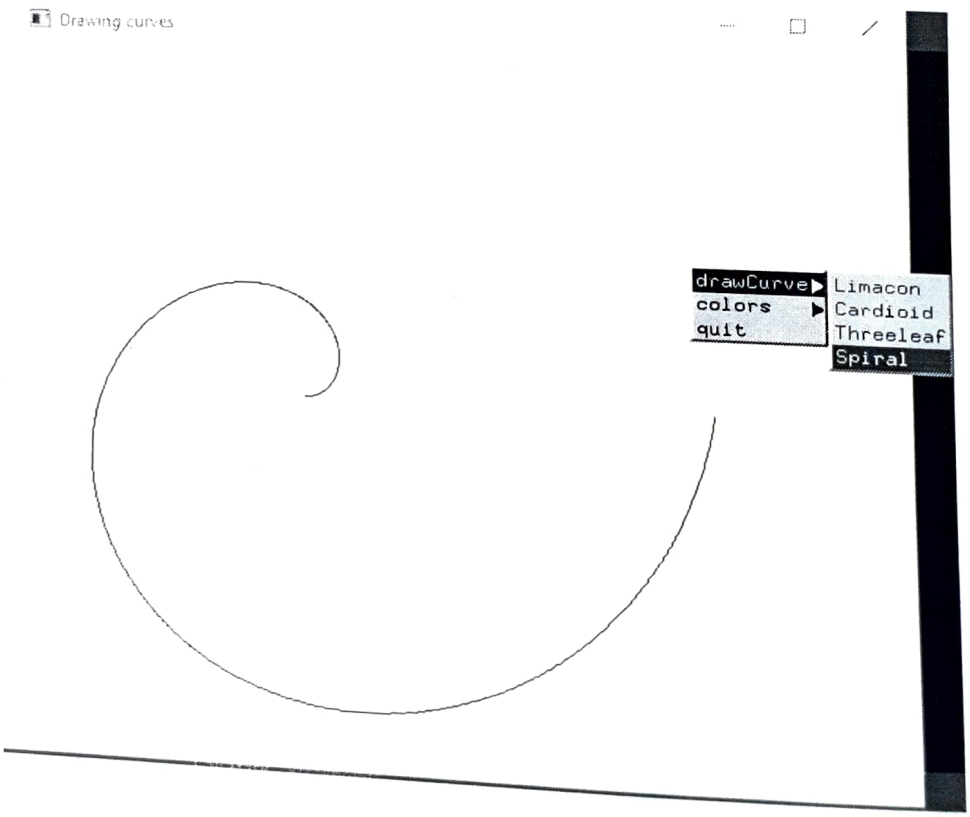
```
case threeleaf: curvePt[0].x += a; break;
case spiral: break;
  default : break;
} theta = dtheta;
while (theta < twopi) {
  switch (curveNum) {
case limacon: r = a * cos(theta) + b; break;
case Cardiod: r = a * (1 + cos(theta)); break;
case threeleaf: r = a * cos(3*theta); break;
case spiral: r = (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; } }
void colorMenu (int id) {
    switch (id) {
   case 0: break;
   case 1: red=0; green=0; blue=1; break;
   case 2: red=0; green=1; blue=0; break;
   case 3: red=0; green=1; blue=1; break;
   case 4: red=1; green=0; blue=0; break;
   case 5: red=1; green=0; blue=1; break;
   case 6: red=1; green=1; blue=0; break;
   case 7: red=1; green=1; blue=1; break;
    default: break; } drawCurve(curve); }
```

```
void main_menu(int id){
 switch(id){  case 3: exit(0);
              default: break; } }

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

void main (int argc, char **argv){
 glutInit(&argc, ~~char~~ argv);
 glutInitWindowSize(w,h);
 int curveId = glutCreateMenu(drawCurve);
glutAddMenuEntry("Limacon",1);
 glutAddMenuEntry("Cardoid",2);
glutAddMenuEntry("ThreeLeaf",3);
  glutAddMenuEntry("Spiral",4);
  glutHackMenu(GLUT_LEFT_BUTTON);
glutAddMenuEntry("Red",4);
      "          ("Green",2);
      "          ("Blue",1);
      "           ("Black",0);
      "           ("Yellow",6);
      "           ("Cyan",3)
      "           ("whit;7),'
 glutAHackMenu(GLUT_LEFT_BUTTON);
 glutCreateMenu(main_menu);
 glutAddSubMenu("drawCurve",curveId);
      "          ("colors",colorId);
```

Teacher's Signature

```
    glutAddMenuEntry("quit", 3);
    glutAttachMenu(GLUT_LEFT_BUTTON);
    myinit();
    glutMainLoop();
}
```





Program11: Drawing Curves

Expt. No. 12 Date

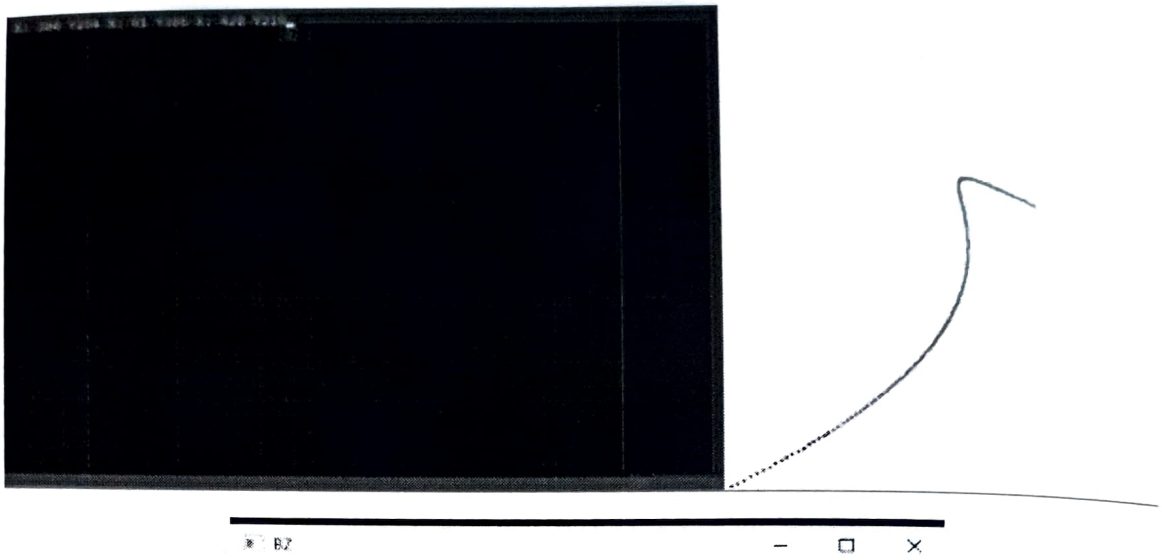
Page No. 37

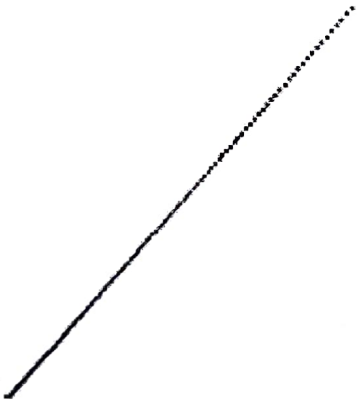
a) write a program to construct Beizer curve.

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

float f,g,r,x1[4],yc[4];
int flag=0;

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

void drawPixel(float x, float y){
      glBegin(GL_POINTS); glVertex2x(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)*yc[0]+3*t*pow(1-t,2)*yc[1]+3*
pow(t,2)*(1-t)*yc[2]+pow(t,3)*yc[3]; glVertex2f(xt,yt);
      }
```

Teacher's Signature

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

int main(int argc, char *argv[]) {
  glutInit(&argc, argv);

cout<<"Enter the X-coordinates";
cin >> x1[0] >> x1[1] >>x1[2] >> x1[3];
cout <<" Enter the y-coordinates";
cin >> yc[0] >>yc[1] >>yc[2] >>yc[3];
  glutInitDisplayMode(GLUT_SINGLE | GLUT-RGB);
  glutDisplayFunc(display);
  myInit();
glutMainLoop();
}
```





Program12: BZ