Date

Expt. No. 1

Page No.

Prgl:- WAP to generate a line using Bresenham's line drawing technique. Consider slopes greater than one and slopes less than one. User must be able to draw as many lines and specify inputs through keyboard/mouse.

```
#include<iostream>
#include <GL/glut.h>
#include <time.h>
using namespace std;
int x1, x2, yc1, y2, flag=0;
void draw_pixel(int x, int y)
{ glColor3f(1,0,0);
  glBegin(GL_POINTS);
  glVertex2i(x,y);
  glEnd();
  glFlush(); }
void draw_line()
{ int dx, dy, i, e, incx, incy, inc1, inc2, x, y;
  dx = x2- x1;
  dy = y2-yc1;
  if (dx<0) dx = -dx;
  if (dy<0) dy = -dy;
  incx = 1;
  if (x2< x1)
     incx = -1;
  incy=1;
  if (y2 < yc1)
     incy =-1;
  x= x1;
  y= yc1;
```

Teacher's Signature :

Date .....

Expt. No. .....

Page No. .....

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

Teacher's Signature : _____

Date .....

Expt. No. .....

Page No. .....

```
void myinit()
{   glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1,1,1,1);
    glOrtho2D(-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: \t");
              x1= x-250;
              yc1= 250-y;
              flag++;
              cout << x1<< "  " <<yc1<< "\n";
           }else
           {  printf("\n Defining x2, y2: \t");
              x2= x-250;
              y2= 250-y;
              flag=0;
              cout << x2<< " " << y2<< "\n";
              draw-line();
           }
        }break;
    }
}
int main(int ac, char *av[])
{  // for keyboard
   cout << "Enter x1 and y1\n";
   cin >> x1>> yc1;
   cout<< "\n Enter x2 and y2 \n";
   cin >>x2>>y2;
```

Teacher's Signature : _____

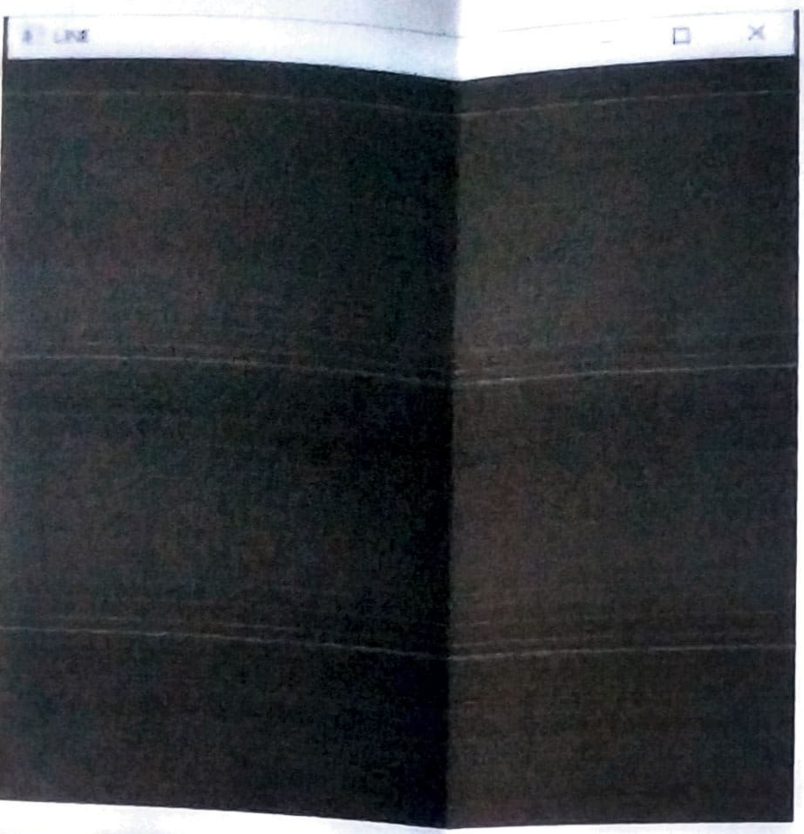
Date .....

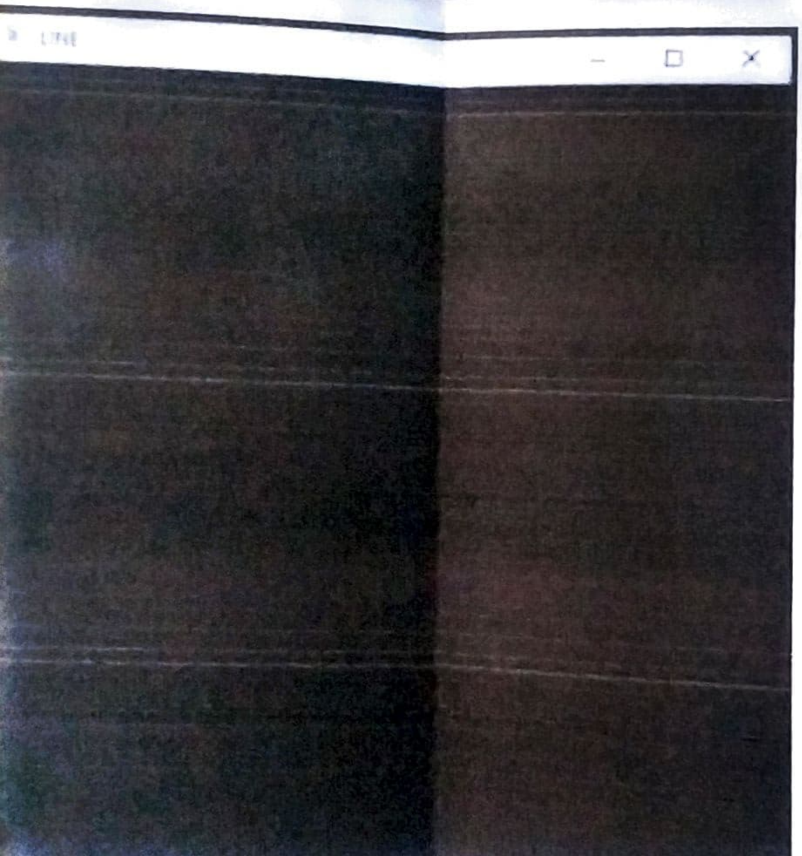
Expt. No. .....

Page No. .....

```
    glutInit(&ac, av);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 200);
    glutCreateWindow("LINE");
    myInit();
    glutMouseFunc(MyMouse);
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Teacher's Signature : _____


LINE


LINE

Date .....

Expt. No. .....

Page No. .....

2. WAP to generate a circle and ellipse using Bresenham's circle drawing and ellipse drawing techniques. Use 2 windows to draw circle in 1 window and ellipse in other window. User can specify inputs through keyboard/mouse.

```
#include <gl/glut.h>
#include <stdio.h>
#include <math.h>
#include <iostream>
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);
}
void circlebres()
{   glClear(GL_COLOR_BUFFER_BIT);
    int x=0, y=r;
    int d=3-2*r;
    while(x<=y)
    {   draw_circle(xc, yc, x, y);
        x++;
        if(d<0) d=d+4*x+6;
        else
        {   y--; d=d+4*(x-y)+10;
        }
```

Teacher's Signature : _____

```
        draw_circle (xc, yc, x, y);
    } glFlush();
}
int p1_x, p2_x, p1_y, p2_y, point1_done = 0;
void myMouseFunc circle (int button, int state, int x, int y)
{   if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN && point1_done == 0)
    {   p1_x = x - 250;   p1_y = 250 - y;   point1_done = 1;  }
    else if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {   p2_x = x - 250;  p2_y = 250 - y;   xc = p1_x;   yc = p1_y;
        float exp = (p2x - p1_x) * (p2_x - p1_x) + (p2_y - p1_y) * (p2_y - p1_y);
        r = (int) (sqrt(exp));
        circlebres();
        point1_done = 0;   }
}
void draw_ellipse (int xce, int yce, int x, int y)
{   glBegin (GL_POINTS);
    glVertex2i ( x + xce, y + yce);
    glVertex2i ( -x + xce, y + yce);
    glVertex2i ( x + xce, -y + yce);
    glVertex2i ( -x + xce, -y + yce);
    glEnd();
}
void midptellipse()
{   glClear(GL_COLOR_BUFFER_BIT);
    float dx, dy, d1, d2, x, y;
    x = 0;   y = (int) ry;
    d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
```

Teacher's Signature : _____

Date .....

Expt. No. .....

Page No. .....

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

~~void~~ void mint()
{ glClearColor (1,1,1,1);
  glColor3f (1,0,0);
  glPointSize (3);
  gluOrtho2D (-250, 250, -250, 250);
}

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

Teacher's Signature : _____

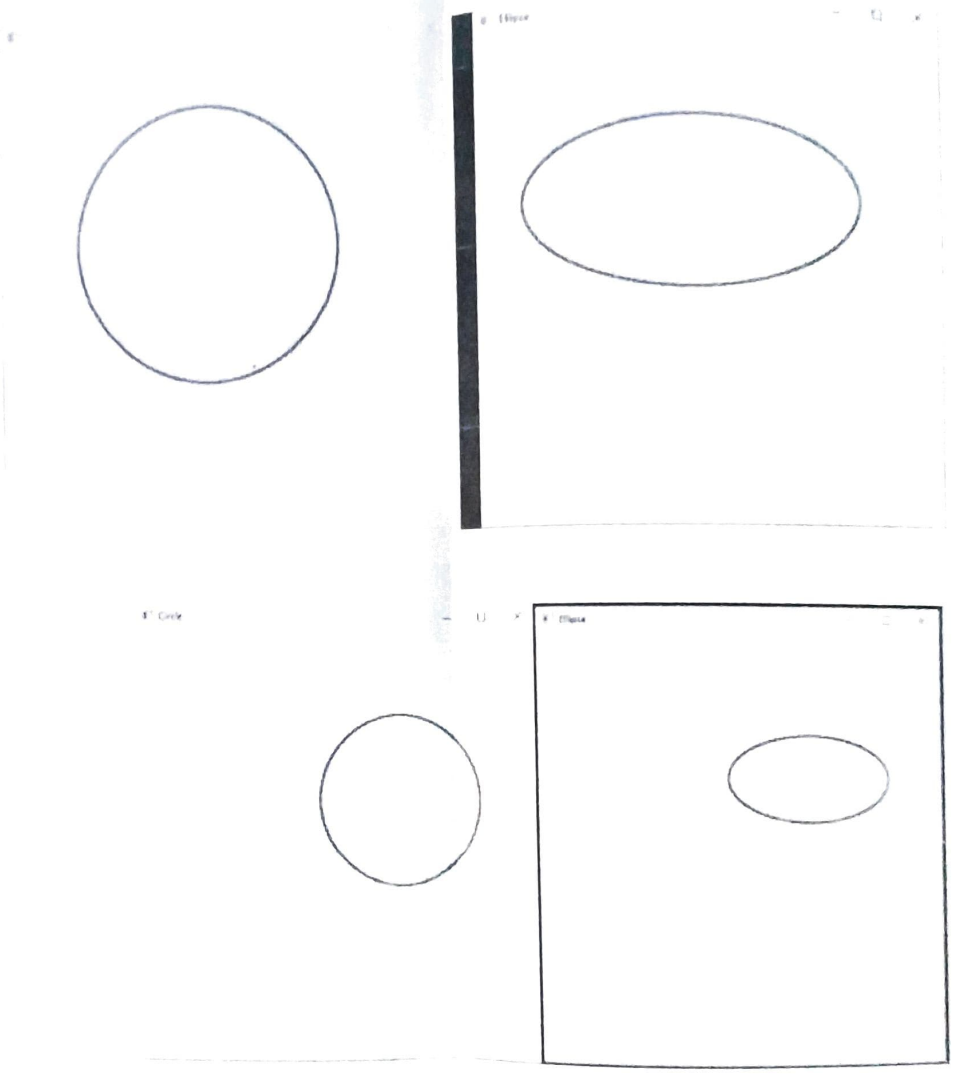
Date .....

Expt. No. .....

Page No. .....

```
glutInitWindowPosition(0,0);
cout << " Creating Circle and ellipse on mouse click\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();
}
```

Teacher's Signature : _____



***Program2:* Circle and Ellipse**

Date

Expt. No. Page No.

3 WAP to recursively subdivides a tetrahedron to form 3D Sierpinski gasket. The no. of recursive steps is to be specified at execution time.

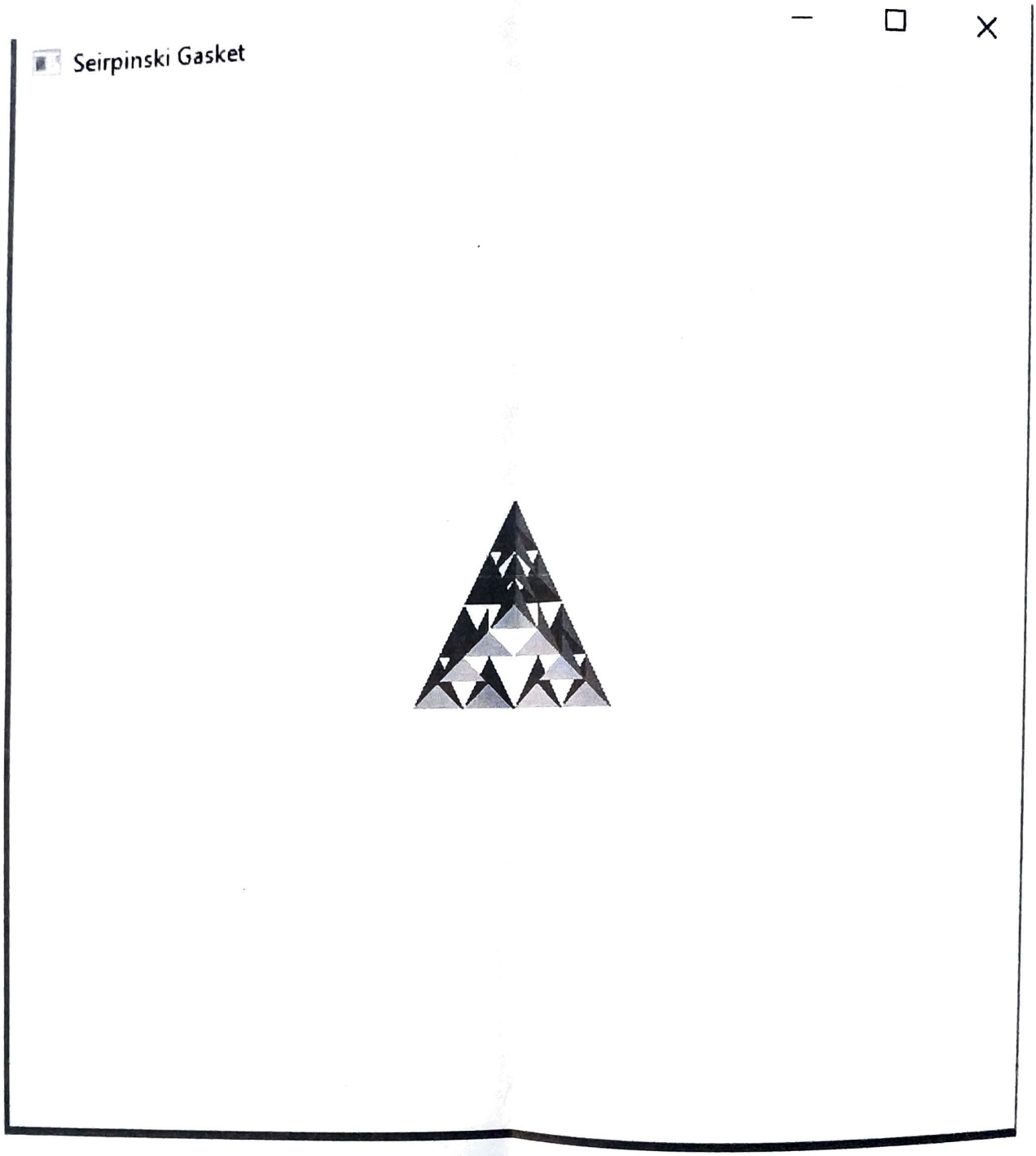
```
#include <gl/glut.h>
#include <stdio.h>
int m;
typedef float point[3];
point tetra[4] = { {0,100,-100}, {0,0,100}, {100,-100,-100}, {-100,-100,-100} };
int main(int argv, char** argc)
{   printf("Enter the no. of iterations: ");
    scanf_s("%d", &m);
    glutInit(&argv, argc);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(100, 200);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Sierpinski Gasket");
    glutDisplayFunc(tetrahedron);
    glEnable(GL_DEPTH_TEST);
    myinit();
    glutMainLoop();
}

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

Teacher's Signature :

```
        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, 500, -500, 500, -500, 500);
}
void tetrahedron(void)
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(1, 0, 0);
    divide_triangle(tetra[0], tetra[1], tetra[2], m);
    glColor3f(0, 1, 0);
    divide_triangle(tetra[3], tetra[2], tetra[1], m);
    glColor3f(0, 0, 1);
    divide_triangle(tetra[0], tetra[3], tetra[1], m);
    glColor3f(0, 0, 0);
    divide_triangle(tetra[0], tetra[2], tetra[3], m);
    glFlush();
}
void draw_triangle(point p1, point p2, point p3)
{   glBegin(GL_TRIANGLES);
    glVertex3fv(p1);
    glVertex3fv(p2);
    glVertex3fv(p3);
    glEnd();
}
```

Teacher's Signature : _____



Program 3: Seirpinski Gasket

Date .....

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

4. WAP to fill any given polygon using scan-line area filling algorithm.

```
#include <stdlib.h>
#include <gl/glut.h>
#include <algorithm>
#include <iostream>
#include <windows.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(scaline >y1 && scalline <y2)
        intx[m++]= x1 + (scanline-y1) *(x2-x1) / (y2-y1);
}
```

Teacher's Signature : _____

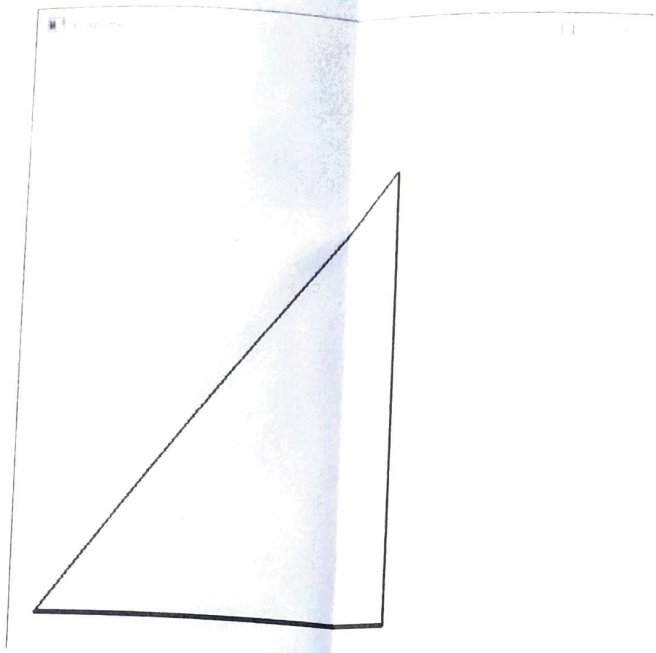
Date .....

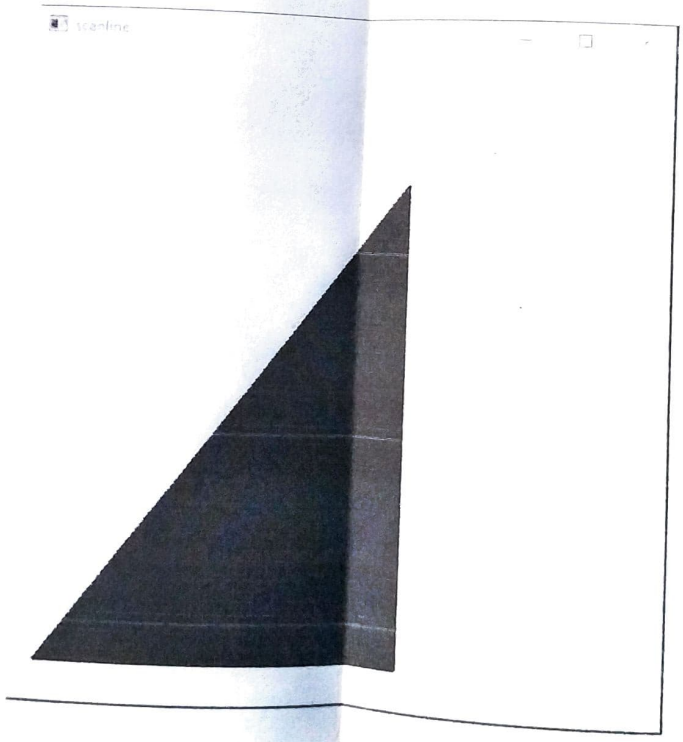
Expt. No. .....

Page No. .....

```
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)%on], y[(i+1)%on], s1);
      sort (intx, (intx+m));
      if (m >= 2)
          for(int 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); glBegin(GL_LINE_LOOP);
  for(int i= 0; i<n; i++)
      glVertex2f(x[i], y[i]);
  glEnd();
  scanfill(x,y); }
void main(int ac, char* av[])
{ glutInit(&ac,av);
  printf("Enter no. of sides:\n");
  cin >>n;
  for(i=0; i<n; i++)
      cin >> x[i] >>y[i];
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(500,500);
  glutInitWindowPosition(0,0);
  glutCreateWindow("scanline");
  glutDisplayFunc(display_filled_polygon);
  myInit();
  glutMainLoop();
}
```

Teacher's Signature : _____





Program4: Scan fill

Date .....

Expt. No. .....

Page No. .....

5. WAP 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 <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);
    for(int i=0; i<11; i++)
        glVertex2fv(house[i]);
    glEnd();
    glPopMatrix(); glFlush();
}
```

Teacher's Signature : _____

Date .....

Expt. No. .....

Page No. .....

```
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();                    //Reflected
    glTranslatef(0, c, 0);
    theta = atan(m); theta = theta * 180 / 3.14;
    glRotatef(theta, 0, 0, 1); glScalef(1, -1, 1);
    glRotatef(-theta, 0, 0, 1); glTranslatef(0, -c, 0);
    glBegin(GL_LINE_LOOP);
    for(int i = 0; i<11; i++)
        glVertex2fv(house[i]);
    glEnd(); glPopMatrix(); glFlush();
}

void myInit()
{   glClearColor(1, 1, 1, 1);
    glColor3f(1, 0, 0);
    glLineWidth(2.0);
```

Teacher's Signature : _____

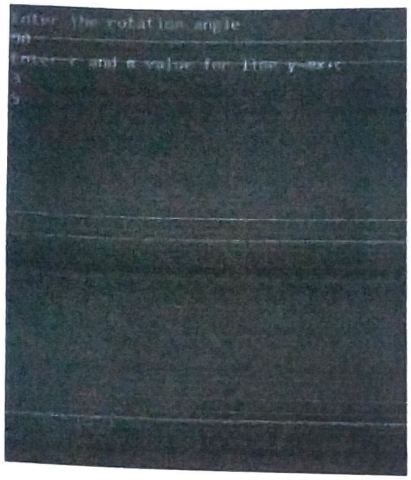
Date .....

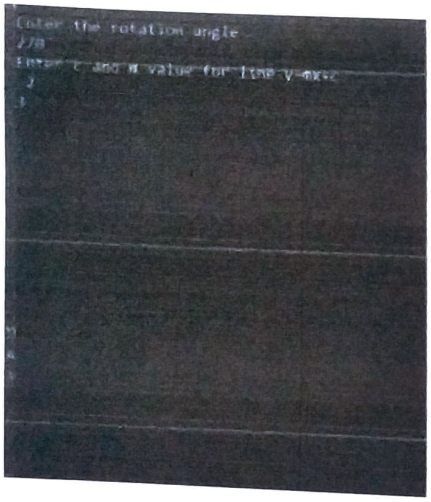
Expt. No. .....

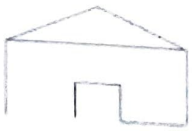
Page No. .....

```
        glMatrixMode(GL_PROJECTION); glLoadIdentity();
        glOrtho2D(-450, 450, -450, 450);
}
void mouse(int btn, int state, int x, int y)
{   if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    { display(); }
    else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    { display2(); }
}
void main(int argc, char** argv)
{   printf("Enter the rotation angle\n");
    cin >> angle;
    cin >> c >> n;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(900,900);
    glutInitWindowPosition(100,100);
    glutCreateWindow("House Rotation");
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    myInit();
    glutMainLoop();
}
```

Teacher's Signature : _____







Program5: House Rotation

Date .....

Expt. No. .....

Page No. .....

6. WAP 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<stdio.h>
#include<stdlib.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, LEFT=8, TOP=1, BOTTOM=2;
int n;
struct line_segment { int x1, y1, x2, y2; };
struct line_segment ls[10];
outcode computeoutcode(double x, double y)
{  outcode code=0;
   if (y>max) code |=TOP;
   else if (y<ymin) code |=BOTTOM;
   if (x>xmax) code |=RIGHT;
   else if (x<xmin) code |=LEFT;
   return code;
}
void cohensuthe(double x0, double y0, double x1, double y1)
{  outcode outcode0, outcode1, outcodeout;
   bool accept=false, done=false;
   outcode0=computeoutcode(x0, y0); outcode1=computeoutcode(x1,y1);
   do{
      if (!(outcode0 | outcode1)) {accept=true; done=true;}
      else if (outcode0 & outcode1) done=true;
```

Teacher's Signature : _____

Date .....

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

```
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, yvmin);
    glVertex2f(xvmax, yvmax); glVertex2f(xvmin, yvmax); glEnd();
    glColor3f(0,0,1); glBegin(GL_LINES);
    glVertex2d(vx0, vy0); glVertex2d(vx1, vy1); glEnd();
}}
void display()
{ glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(0,0,1); glBegin(GL_LINE_LOOP); glVertex2f(xmin, ymin);
  glVertex2f(xmax, ymin); glVertex2f(xmax, ymax); glVertex2f(xmin, ymax);
  glEnd();
```

Teacher's Signature : _____

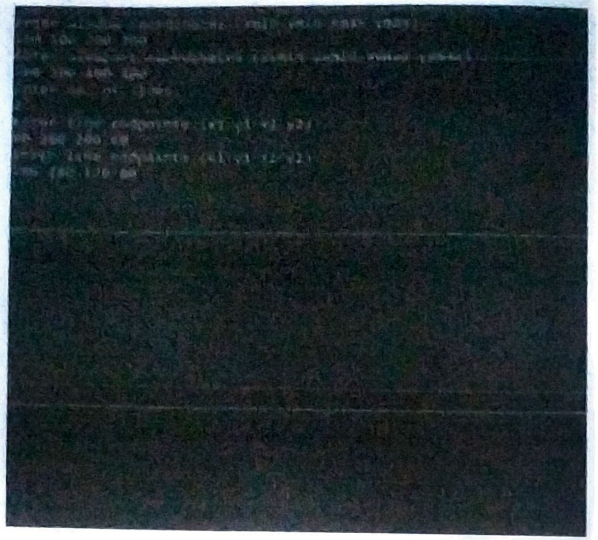
Date .....

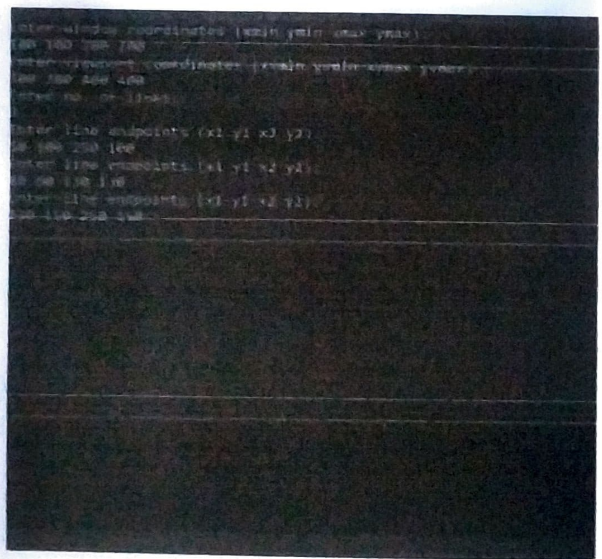
Expt. No. .....

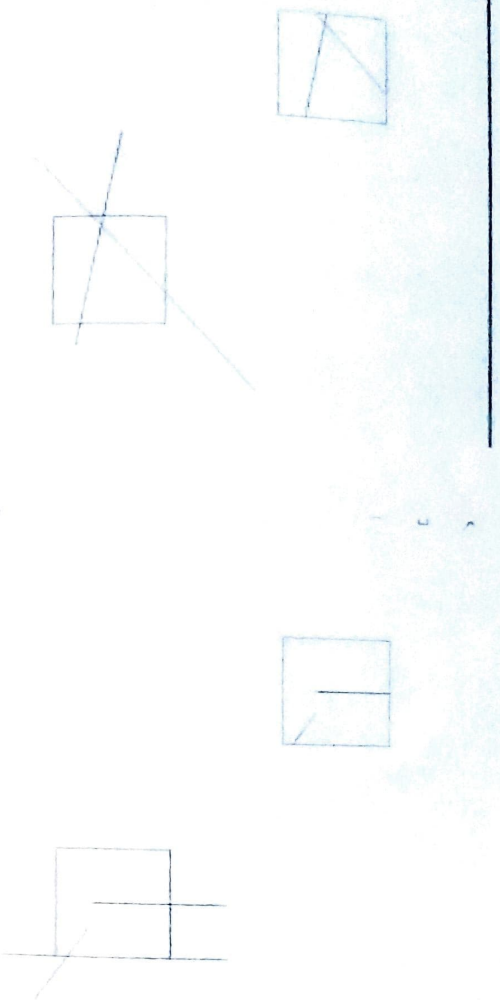
Page No. .....

```
    for(int i=0; i<n; i++)
    { glBegin(GL_LINES); glVertex2d(ls[i].x1, ls[i].y1);
      glVertex2d(ls[i].x2, ls[i].y2); glEnd(); }
    for(int i=0; i<n; i++)
        cohensuther(ls[i].x1, ls[i].y1, ls[i].x2, ls[i].y2);
    glFlush();
}
void myinit()
{   glClearColor(1,1,1,1); glColor3f(1,0,0); glPointSize(1);
    glMatrixMode(GL_PROJECTION); glLoadIdentity(); gluOrtho2D(0,500, 0,500);
}
void main(int argc, char **argv)
{   scanf_s("%lf%lf%lf%lf", &xmin, &ymin, &xmax, &ymax);
    scanf_s("%lf%lf%lf%lf", &xvmin, &yvmin, &xvmax, &yvmax);
    scanf_s("%d", &n);
    for(int i=0; i<n; i++)
        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);
    glutCreateWindow("clip");
    myinit();
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Teacher's Signature : _____







Program6: CohenSutherLand

Expt. No. Date

Page No.

7. WAP 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 <GL/glut.h>
double xmin, ymin, xmax, ymax, xvmin, yvmin, xvmax, yvmax;
int n;
Struct line_segment { int x1, y1, x2, 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)
    { if (r < *u2) *u2 = r; if (r < *u1) return false; }
    else if (p==0) { if (q<0) return false; }
    return true;
}
void LiangBarskyLineClipAndDraw (double x0, double y0, double x1, double y1)
{   double dx = x1-x0, dy = y1-y0, u1=0, u2=1;
    glColor3f(1, 0, 0); glBegin(GL_LINE_LOOP);
    glVertex2f(xvmin, yvmin); glVertex2f(xvmax, yvmin);
    glVertex2f(xvmax, yvmax); glVertex2f(xvmin, yvmax);
    glEnd();
    if (cliptest (-dx, x0-xmin, &u1, &u2))
        if (cliptest (dx, xmax-x0, &u1, &u2))
            if (cliptest (-dy, y0-ymin, &u1, &u2))
                if (cliptest (dy, ymax-y0, &u1, &u2))
                { if (u2<1) { x1 = x0+u2*dx; y1 = y0+u2*dy; }
```

Teacher's Signature :

Date .....

Expt. No. .....

Page No. .....

```
        if(u1>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(0,0,1); glBegin(GL_LINES);
    glVertex2d(vx0,vy0); glVertex2d(vx1,vy1); glEnd();
}
void display()
{   glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    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(0,0,1); glBegin(GL_LINE_LOOP); glVertex2f(xmin,ymin);
    glVertex2f(xmax,ymin); glVertex2f(xmax,ymax); glVertex2f(xmin,ymax); glEnd();
    for(int i=0;i<n;i++)
        LiangBarskyLineClipAndDraw(ls[i].x1,ls[i].y1,ls[i].x2,ls[i].y2);
    glFlush();
}
void myinit()
{   glClearColor(1,1,1,1); glColor3f(1,0,0);
    glLineWidth(2);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho2D(0,499,0,499);
}
```

Teacher's Signature : _____

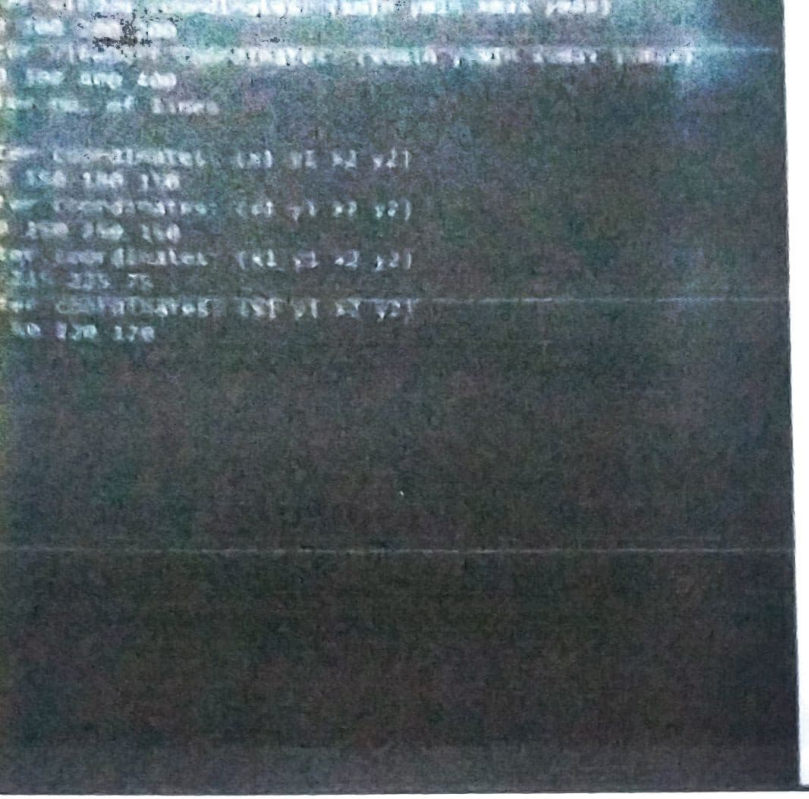
Date .....

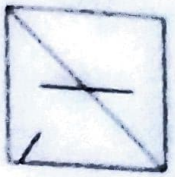
Expt. No. .....

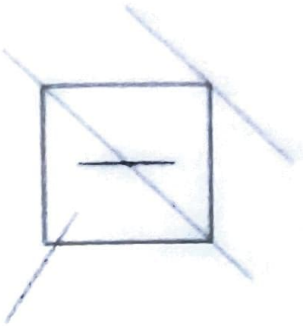
Page No. .....

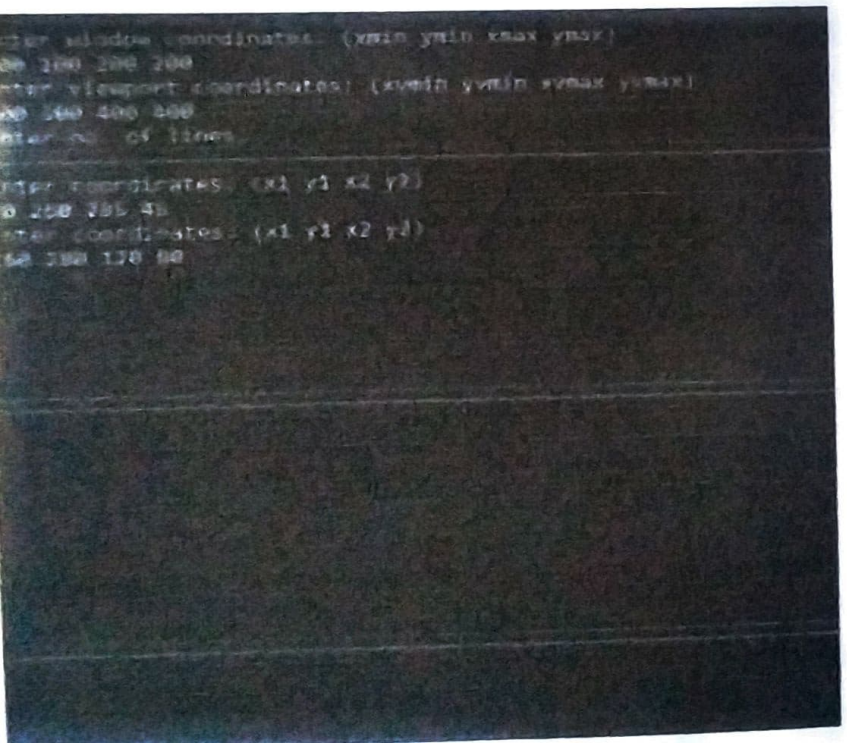
```
int main(int argc, char **argv)
{   glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0,0);
    printf("Enter window coordinates: (xmin ymin xmax ymax)\n");
    scanf("%lf %lf %lf %lf", &xmin, &ymin, &xmax, &ymax);
    printf("Enter viewport coordinates: (xvmin yvmin xvmax yvmax)\n");
    scanf("%lf %lf %lf %lf", &xvmin, &yvmin, &xvmax, &yvmax);
    printf("Enter no. of lines: \n");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {   printf("Enter coordinates: (x1 y1 x2 y2)\n");
        scanf("%d %d %d %d", &ls[i].x1, &ls[i].y1, &ls[i].x2, &ls[i].y2);
    }
    glutCreateWindow("Liang Barsky Line Clipping Algorithm");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

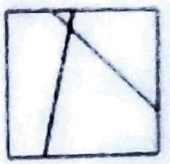
Teacher's Signature : _____

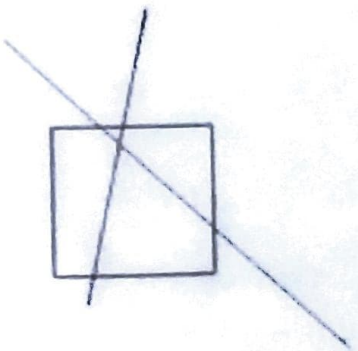












Date

Expt. No. Page No.

8. WAP to implement the Cohen-Hodgeman polygon clipping algorithm. Make provision to specify the input polygon and window for clipping.

```
#include <iostream>
#include <GL/glut.h>
using namespace std;
int poly_size, poly_points[20][2], org_poly_size, org_poly_points[20][2], clipper_size,
clipper_points[20][2];
const int MAX_POINTS = 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*x4 - y3*x4);
  int den = (x1-x2) * (y3-y4) - (y1-y2) * (x3-x4);
  return num/den;
}
void clip(int poly_points[][2], int &poly_size, int x1, int y1, int x2, int y2)
{ int new_points[MAX_POINTS][2], new_poly_size = 0;
  for (int i = 0; i < poly_size; i++)
  { int k = (i+1) % poly_size;
    int ix = poly_points[i][0], iy = poly_points[i][1];
    int kx = poly_points[k][0], ky = poly_points[k][1];
    int i_pos = (x2-x1) * (iy-y1) - (y2-y1) * (ix-x1);
```

Teacher's Signature :

Date .....

Expt. No. .....

Page No. .....

```
int k_pos = (x2-x1) * (ky - y1) - (y2 - y1) * (kx - x1);
if (i_pos >=0 && k_pos >=0)
{ new_points[new_poly_size][0] = kx; new_points[new_poly_size][1] = ky;
  new_poly_size++; }
else if (i_pos<0 && k_pos >=0)
{ new_points[new_poly_size][0] = x_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
  new_points[new_poly_size][1] = y_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
  new_poly_size++; new_points[new_poly_size][0] = kx;
  new_points[new_poly_size][1] = ky; new_poly_size++;
}
else if( i_pos >=0 && k_pos<0)
{ new_points[new_poly_size][0] = x_intersect(x1,y1, x2, y2, ix, iy, kx, ky);
  new_points[new_poly_size][1] = y_intersect(x1, y1, x2, y2, ix, iy, kx, ky);
  new_poly_size++;
}}
poly_size = new_poly_size;
for(int i=0; i< poly_size; i++)
{ poly_points[i][0] = new_points[i][0]; poly_points[i][1] = new_points[i][1]; }
}
void init()
{ glClearColor(0, 0, 0, 0); glMatrixMode(GL_PROJECTION); glLoadIdentity();
  glOrtho(0, 500, 0, 500, 0, 500); glClear(GL_COLOR_BUFFER_BIT);
}
void display()
{ init(); glColor3f(1, 0, 0);
  drawPoly(clipper_points, clipper_size); glColor3f(0, 1, 0);
  drawPoly(org_poly_points, org_poly_size);
  for(int i=0; i< clipper_size; i++)
  { int k = (i+1)% clipper_size;
```

Teacher's Signature : _____

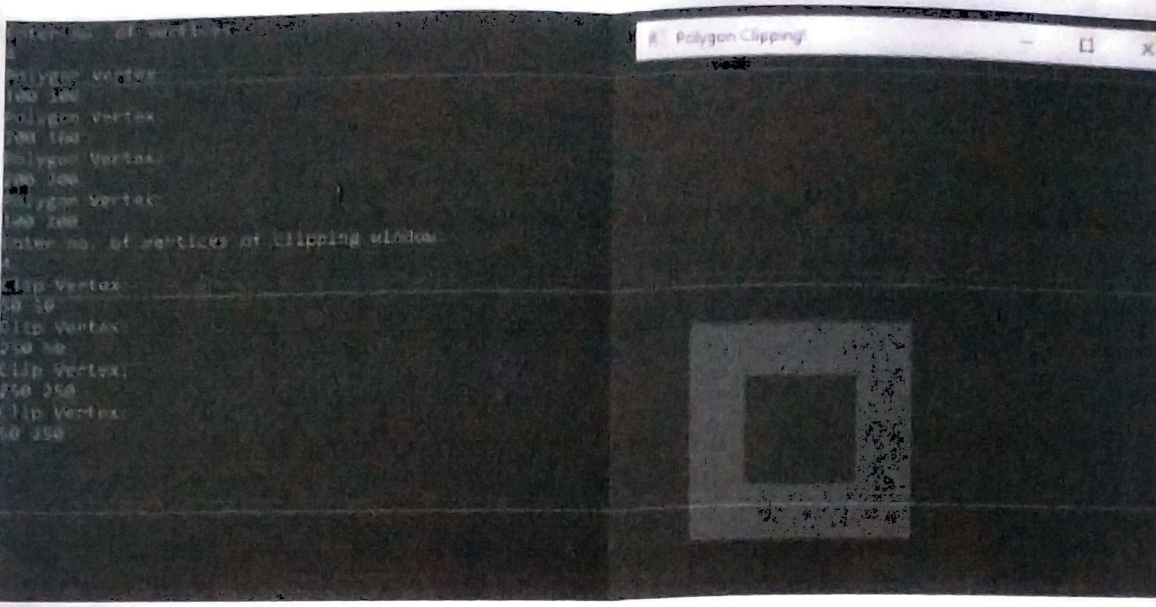
Date .....

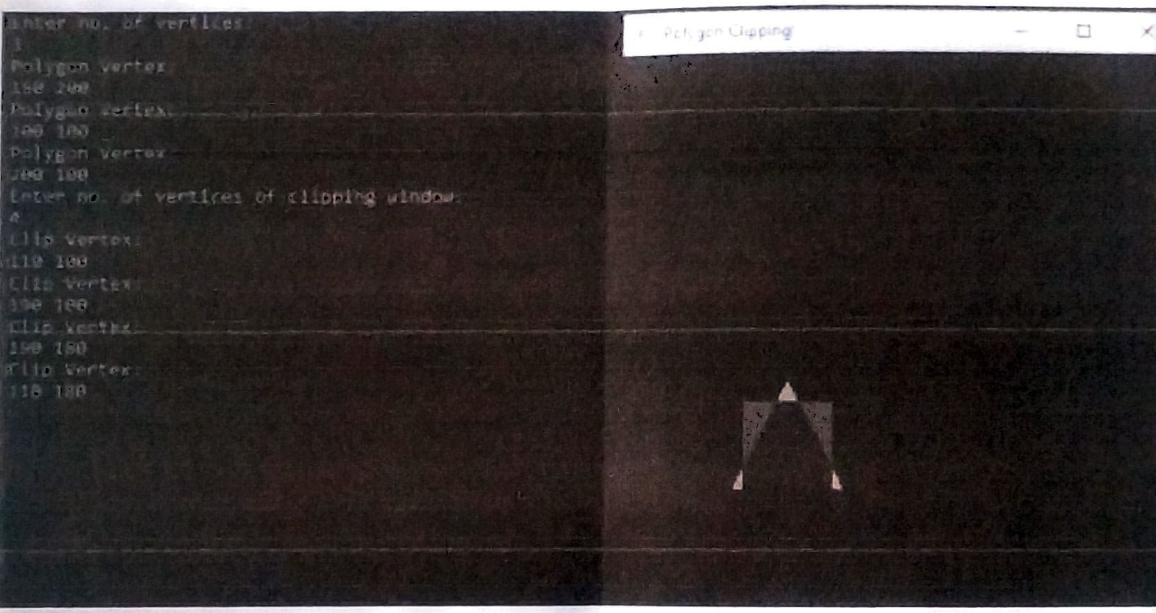
Expt. No. .....

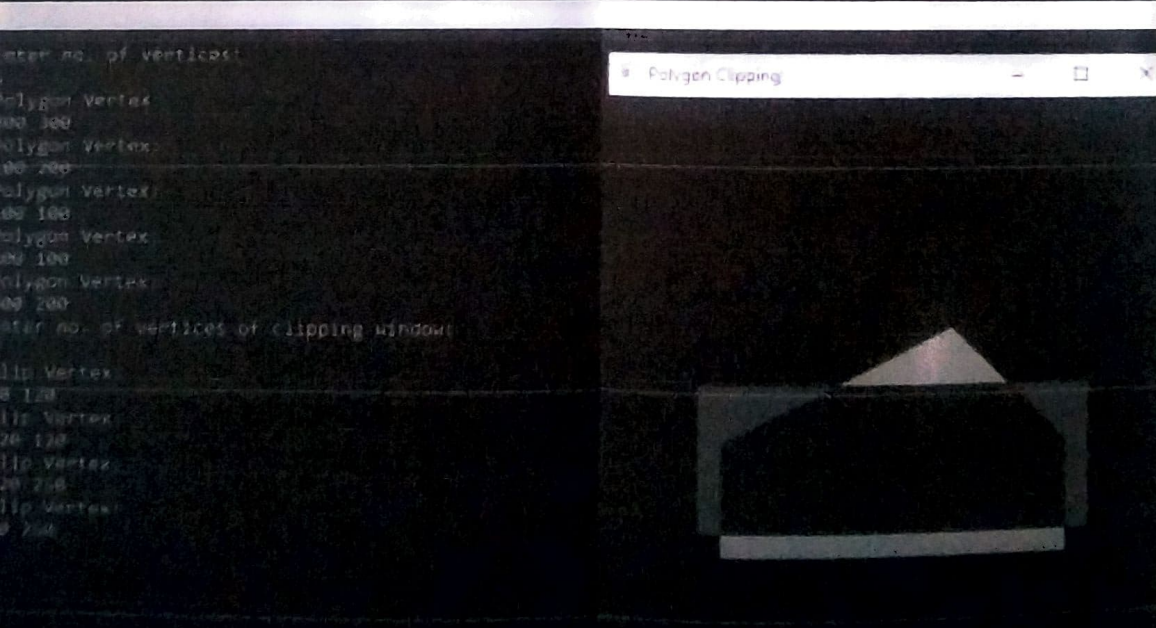
Page No. .....

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

Teacher's Signature : _____


Polygon Clipping


Polygon Clipping


Polygon Clipping

Date .....

Expt. No. .....

Page No. .....

9. WAP to model a car like figure using display lists and move a car from one end of the screen to othe end. User is able to control the speed using mouse.

```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
#define CAR 1
#define WHEEL 2
float s= 1;
void carlist()
{   glNewList (CAR, GL_COMPILE);
    glColor3f(1,1,1);
    glBegin (GL_POLYGON);
    glVertex3f(0,25,0); glVertex2f (90,25,0); glVertex2f (90,55,0);
    glVertex3f (80,55,0); glVertex2f (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;
   }
}
```

Teacher's Signature : _____

Date .....

Expt. No. .....

Page No. .....

```
void myInit()
{   glClearColor(0,0,0,0);
    glOrtho(0, 600, 0, 600, 0, 600);
}
void drawwheel()
{   glColor3f(0,1,1);
    glutSolidSphere(10, 25, 25);
}
void movecar(float s)
{   glTranslatef(s, 0, 0);
    glCallList(CAR); glPushMatrix(); glTranslatef(25, 25, 0);
    glCallList(WHEEL); glPopMatrix(); glPushMatrix(); glTranslatef(75, 25, 0);
    glCallList(WHEEL); glPopMatrix(); glFlush();
}
void myDispl()
{   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)
    { st = 5;
      myDispl();
    }
    else if(btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    { st = 2;
      myDispl();
    }
}
```

Teacher's Signature : _____

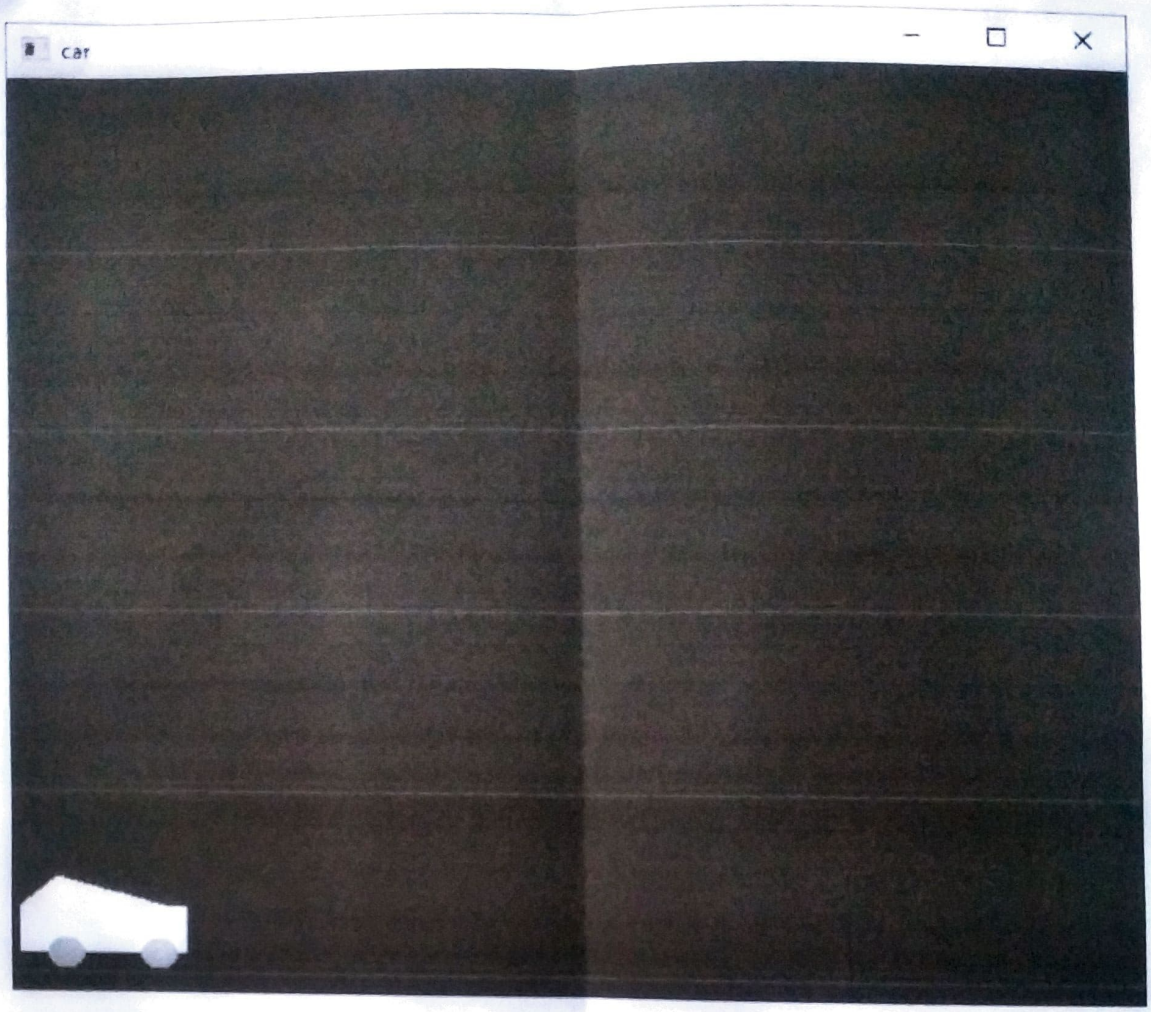
Date

Expt. No.

Page No.

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

Teacher's Signature :



Program 9: Car

Date .....

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

10. WAP to create a color cube and spin it using OpenGL transformations.

```
#include <stdlib.h>
#include <GL/glut.h>
#include <gl/GL.h>
#include <gl/GLU.h>
#include <time.h>
GLfloat vertices[] = { -1,-1,-1, 1,-1,-1, 1, 1,-1,-1, 1,-1,-1,-1, 1, 1,-1, 1,
                       1, 1,-1, 1, 1 };
GLfloat normals[] = { -1,-1,-1,1, -1,-1, 1, 1,-1,-1,1, -1,-1,-1,1, 1,-1,1, 1, 1,
                      1,-1, 1, 1 };
GLfloat colors[] = { 0,0,0, 1,0,0, 1,1,0, 0,1,0, 0,0,1, 1,0,1, 1,1,1,
                     1,0,1, 1 };
GLubyte cubeIndices[] = { 0,3,2,1, 2,3,7,6, 0,4,7,3, 1,2,6,5, 4,5,6,7,
                          0,1,5,4 };
static GLfloat theta[] = {0,0,0};
static GLfloat beta[] = {0,0,0};
static GLint axis = 2;
void delay(float secs)
{   float end = clock() / CLOCKS_PER_SEC + secs;
    while((clock() / CLOCKS_PER_SEC) < end);
}
void displaySingle(void)
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glRotatef(theta[0], 1, 0, 0);
    glRotatef(theta[1], 0,1,0);
    glRotatef(theta[2], 0,0,1);
```

Teacher's Signature : _____

```
    glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);
    glBegin(GL_LINES);
    glVertex2f(0,0,0);
    glVertex3f(1,1,0);
    glEnd();
    glFlush();
}
void spinCube()
{   delay(0.01);
    theta[axis] += 2;
    if(theta[axis] > 360) theta[axis] -= 360;
    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, 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);
}
```

Teacher's Signature : _____

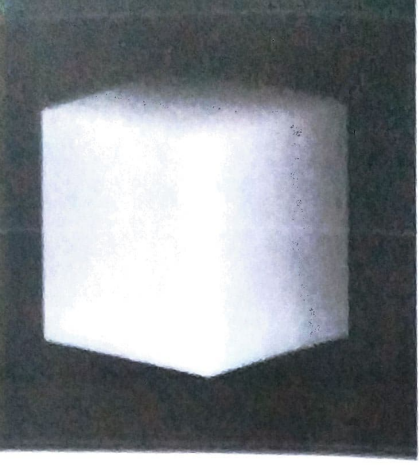
Date .....

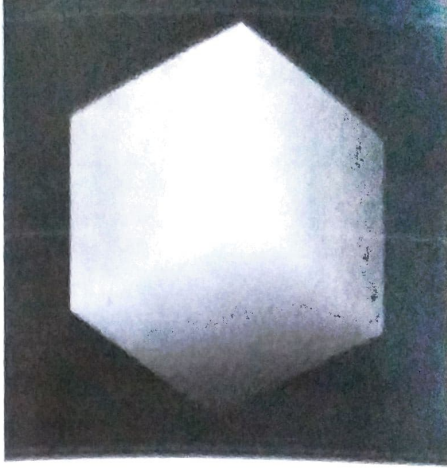
Expt. No. .....

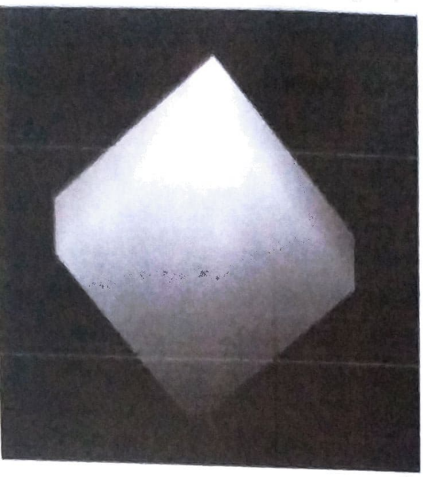
Page No. .....

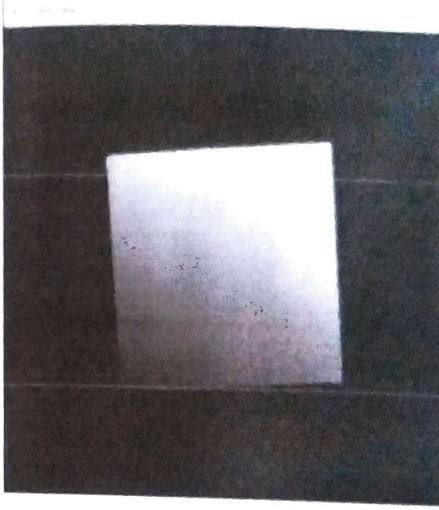
```
void main (int argc, char ** argv)
{   glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition (100, 100);
    glutInitWindowSize (500, 500);
    glutCreateWindow (" colorcube ");
    glutReshapeFunc (myReshape);
    glutDisplayFunc (displaySingle);
    glutIdleFunc (spinCube);
    glutMouseFunc (mouse);
    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();
}
```

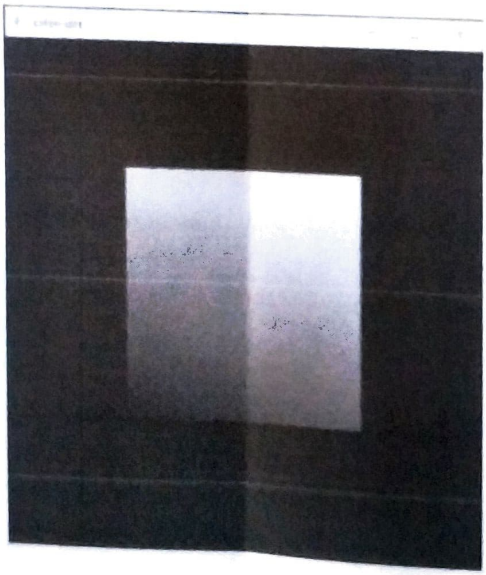
Teacher's Signature : _____











Program10: Colour Cube

Date

Expt. No.

Page No.

11. WAP to create a hierarchial menu and attach appropriate services to each menu entries with mouse buttons.

```
#include <gl/glut.h>
#include <math.h>
#include <stdio.h>
struct screenPt { int x, y; };
typedef enum { limacon = 1, cardroid = 2, threeleaf = 3, spiral = 4 } curveName;
int w = 600, h = 500;
int curve = 1, red = 0, green = 0, blue = 0;
void myinit(void)
{   glClearColor (1,1,1,1);
    glMatrixMode (GL_PROJECTION);
    gluOrtho2D (0, 200, 0, 150);
}
void lineSegment (screenPt p1, screenPt p2)
{   glBegin (GL_LINES);
    glVertex2i ( p1.x, p1.y);
    glVertex2i ( p2.x, p2.y);
    glEnd();
    glFlush();
}
void drawCurve (int curveNum)
{   const double twoPi = 6.283185;
    const int a = 175, b = 60;
    float r, theta, dtheta = 1/ float(a);
    int x0 = 200, y0 = 250;
    screenPt curvePt[2];
    curve = curveNum;
```

Teacher's Signature :

```
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) * 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 mydisplay()
{ }
void myreshape(int nw, int nh)
{  glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho2D(0, (double)nw, 0, (double)nh);
   glClear(GL_COLOR_BUFFER_BIT);
}
void main(int argc, char** argv)
{  glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

Teacher's Signature : _____

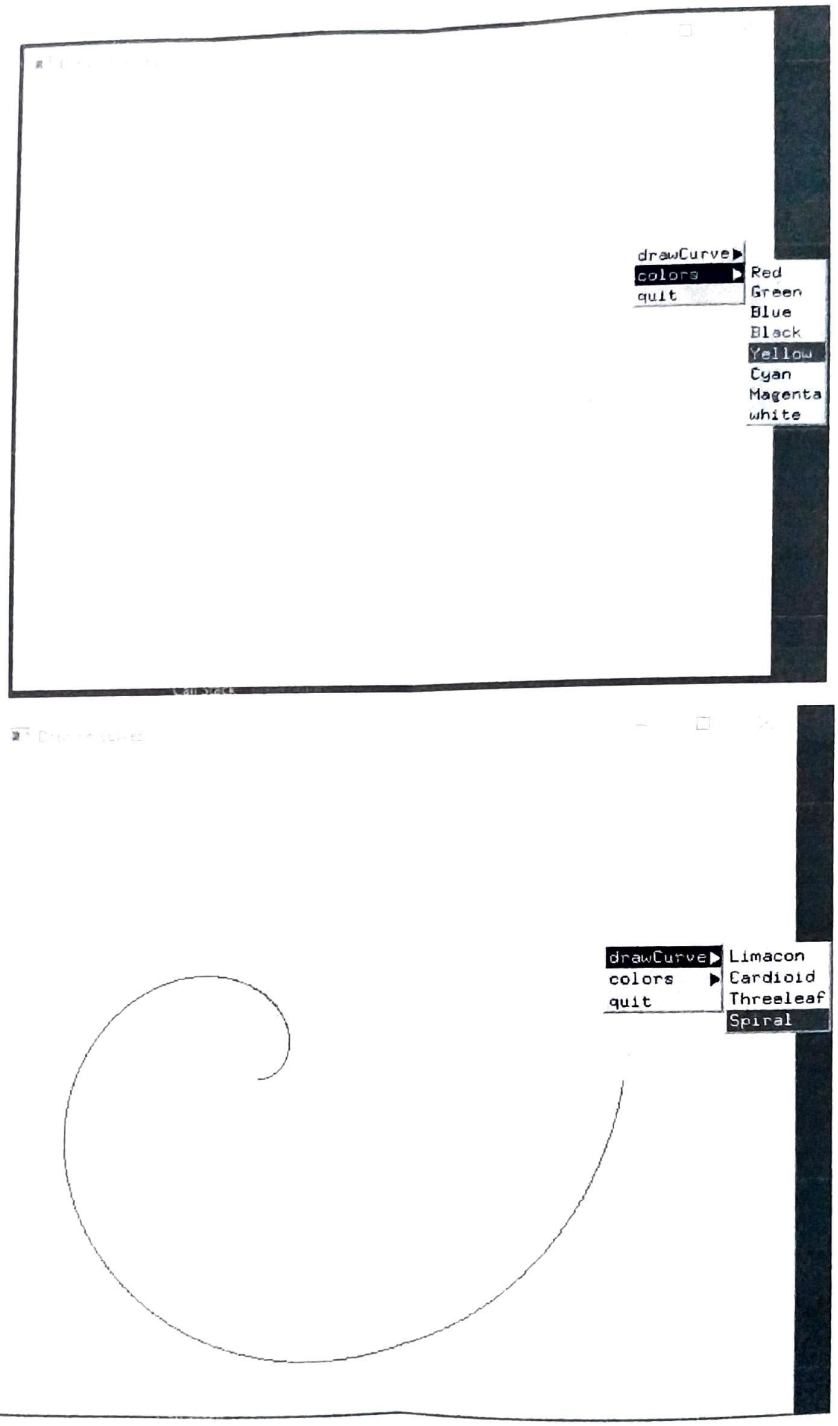
Date .....

Expt. No. .....

Page No. .....

```
glutInitWindowSize (w,h);
glutInitWindowPosition (100, 100);
glutCreateWindow ("Drawing curves");
int curveId = glutCreateMenu (drawCurve);
glutAddMenuEntry ("Limacon", 1);
glutAddMenuEntry ("cardioid", 2);
glutAddMenuEntry ("Threeleaf", 3);
glutAddMenuEntry ("Spiral", 4);
glutAttachMenu (GLUT_LEFT_BUTTON);
int colorId = glutCreateMenu (colorMenu);
glutAddMenuEntry ("Red", 4);
glutAddMenuEntry ("Green", 2);
glutAddMenuEntry ("Blue", 1);
glutAddMenuEntry ("Black", 0);
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 : _____



Program11: Drawing Curves

Date

Expt. No. Page No.

12. WAP to construct Bezier curve. Control points are supplied through keyboard / mouse.

```
#include <iostream>
#include <math.h>
#include <gl/glut.h>
using namespace std;
float f, g, gn, x1[4], yc[4];
int flag = 0;
void myInit()
{   glClearColor(1,1,1,1);
    glColor3f(1,1,1);
    glPointSize(5);
    gluOrtho2D(0, 500, 0, 500);
}
void drawPixel(float x, float y)
{   glBegin(GL_POINTS);
    glVertex2f(x,y);
    glEnd();
}
void display()
{   glClear(GL_COLOR_BUFFER_BIT);
    int i;
    double t;
    glColor3f(0,0,0);
    glBegin(GL_POINTS);
    for(t=0; t<1; t=t+0.005)
    {   double xt = pow(1-t,3) * x1[0] + 3*t*pow(1-t,2)* x1[1] +
                3*pow(t,2)*(1-t)*x1[2] + pow(t,3)* x1[3];
```

Teacher's Signature :

```
double yt = pow(1-t,3) * yc[0] + 3* t * pow(1-t, 2)* yc[1] +3*
  pow (t,2) * (1-t) * yc[2] + pow(t,3) * yc[3];
  glVertex2f (xt, yt);
}
glColor3f (1,1,0);
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 << " x: " << x << " y" << 500-y;
      glPointSize (3);
      glColor3f(1,1,0);
      glBegin (GL_POINTS);
      glVertex2i (x, 500-y);
      glEnd();
      glFlush();
      flag++;
  }
  if (flag >= 4 && btn == GLUT_LEFT_BUTTON)
  {   glColor3f (0,0,1);
      display();
      flag=0;
  }
```

Teacher's Signature : _____

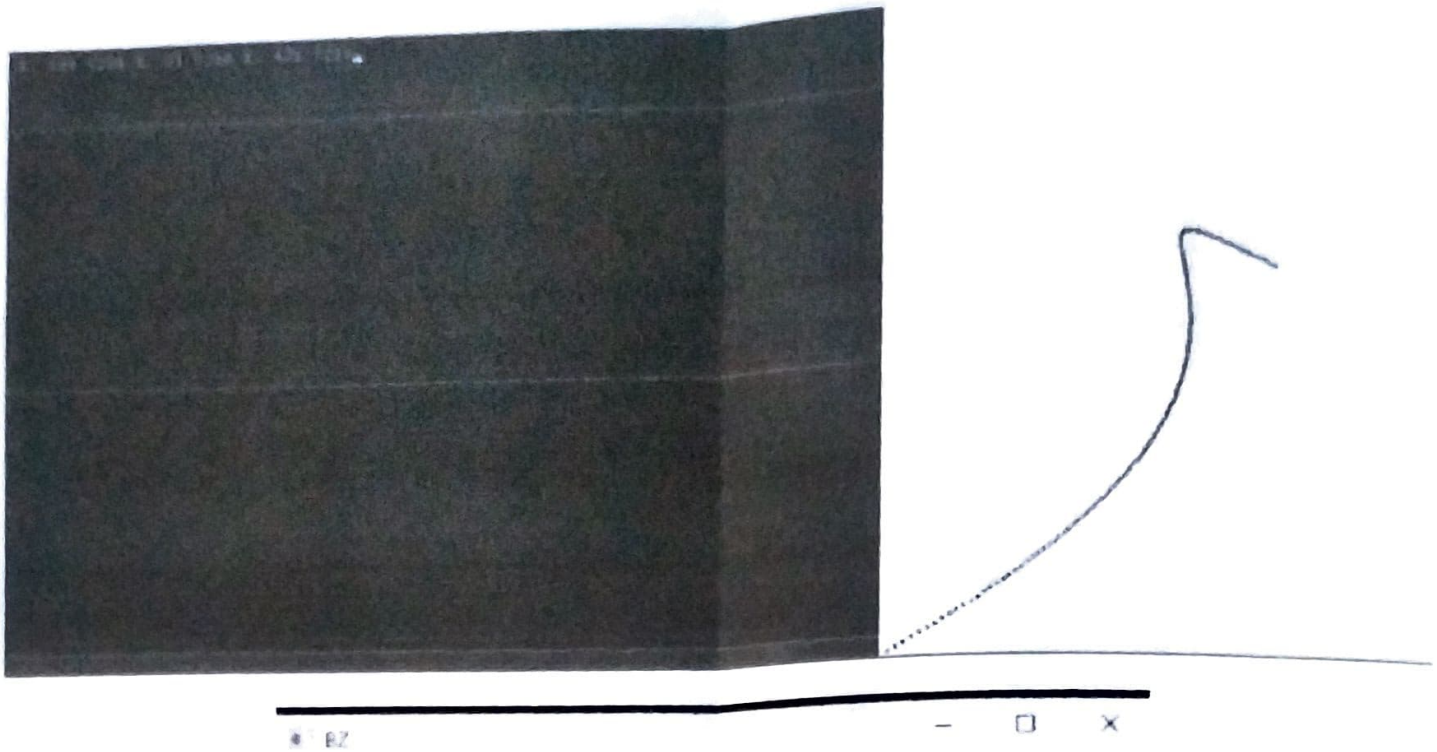
Date ----------

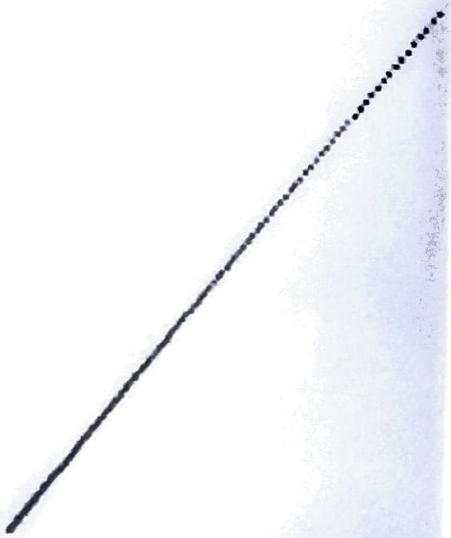
Expt. No. ----------

Page No. ----------

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

Teacher's Signature : _____





Program12: BZ