Lab Program - 1

Implement Bresenham's line drawing algorithm for all types of slope.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<math.h>

void Draw()
{
    GLfloat x1=450,y1=350,x2=150,y2=350;
    GLfloat M,p,dx,dy,x,y,t;
    glClear(GL_COLOR_BUFFER_BIT);

    if(x1==x2)
        M = (y2-y1);
    else
        M = (y2-y1)/(x2-x1);

    if(fabs(M)<1)
    {
        if(x1>x2)
        {
            t = x1;
            x1 = x2;
            x2 = t;

            t = y1;
            y1 = y2;
            y2 = t;
        }

        dx = fabs(x2-x1);
        dy = fabs(y2-y1);

        p = 2*dy-dx;

        x=x1;
        y=y1;

        glBegin(GL_POINTS);
            while(x<=x2)
            {
                glVertex2f(x,y);
                x=x+1;

                if(p>=0)
                {
                    if(M>0)
                        y=y+1;
                    else
                        y=y-1;
                    p = p+2*dy-2*dx;
                }
                else
                {
                    y=y;
                    p = p+2*dy;
```

```
            }
          }
       glEnd();
    }

    if(fabs(M)>=1)
    {
       if(y1>y2)
       {
          t = x1;
          x1 = x2;
          x2 = t;

          t = y1;
          y1 = y2;
          y2 = t;
       }

       dx = fabs(x2-x1);
       dy = fabs(y2-y1);

       p = 2*dx-dy;

       x=x1;
       y=y1;

       glBegin(GL_POINTS);
          while(y<=y2)
          {
           glVertex2f(x,y);
           y=y+1;

           if(p>=0)
           {
             if(M>0)
                x=x+1;
             else
                x=x-1;
             p = p+2*dx-2*dy;
           }
           else
           {
             x=x;
             p = p+2*dx;
           }
          }
       glEnd();
    }

    glFlush();
}

void MyInit()
{
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0,500,0,500);
   glMatrixMode(GL_MODELVIEW);
}
```

```
int main(int argC,char *argV[])
{
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Brenham's Line Drawing Algo");
    MyInit();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

https://youtu.be/H1RtMA3XV3k

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Lab Program - 2

Create and rotate a triangle about the origin and a fixed point.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

GLfloat R,px,py;

void Draw()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0,0,1);
    glBegin(GL_LINE_LOOP);
        glVertex2f(0.1,0.4);
        glVertex2f(0.7,0.4);
        glVertex2f(0.4,0.8);
    glEnd();

    glLoadIdentity();

    glTranslatef(px,py,0);
    glRotatef(R,0,0,1);
    glTranslatef(-px,-py,0);

    glColor3f(1,0,0);
    glBegin(GL_LINE_LOOP);
        glVertex2f(0.1,0.4);
        glVertex2f(0.7,0.4);
        glVertex2f(0.4,0.8);
    glEnd();

    glFlush();
}

int main(int argC,char *argV[])
{
    printf("Enter the Rotation Reference Point [Pivot Point] : ");
    scanf("%f%f",&px,&py);
    printf("Enter the Rotation Degree : ");
    scanf("%f",&R);
    glutInit(&argC,argV);
    glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(500,500);
    glutCreateWindow("Triangle Rotation");
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```
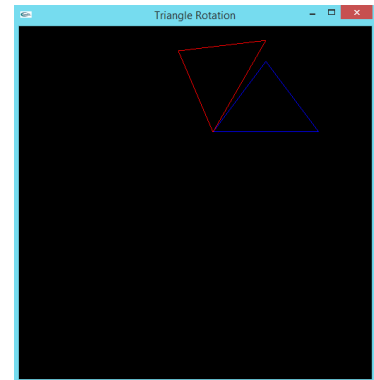
https://youtu.be/apF4MUgp6k8

	Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Lab Program - 3

Draw a colour cube and spin it using OpenGL transformation matrices.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

GLfloat d = 0;
int a=0;

void MyInit()
{
   glClearColor(0,0,0,1);
   glEnable(GL_DEPTH_TEST);
}

void Spin()
{
   d = d + 0.25;
   if(d > 360)
      d = 0;
   glutPostRedisplay();
}

void Face(GLfloat A[],GLfloat B[],GLfloat C[],GLfloat D[])
{
   glBegin(GL_POLYGON);
      glVertex3fv(A);
      glVertex3fv(B);
      glVertex3fv(C);
      glVertex3fv(D);
   glEnd();
}

void Cube(GLfloat V0[],GLfloat V1[],GLfloat V2[],GLfloat V3[],GLfloat V4[],GLfloat V5[],GLfloat V6[],GLfloat V7[])
{
   glColor3f(1,0,0);
   Face(V0,V1,V2,V3); //Front
   glColor3f(0,1,0);
   Face(V4,V5,V6,V7); //Back
   glColor3f(0,0,1);
   Face(V0,V4,V7,V3); //Left
   glColor3f(1,1,0);
   Face(V1,V5,V6,V2); //Right
   glColor3f(1,0,1);
   Face(V2,V3,V7,V6); //Bot
   glColor3f(0,1,1);
   Face(V0,V1,V5,V4); //Top
}

void Draw()
{
   GLfloat V[8][3] =  {
               {-0.5, 0.5, 0.5},
               { 0.5, 0.5, 0.5},
               { 0.5,-0.5, 0.5},
               {-0.5,-0.5, 0.5},
```

```
                            {-0.5, 0.5,-0.5},
                            { 0.5, 0.5,-0.5},
                            { 0.5,-0.5,-0.5},
                            {-0.5,-0.5,-0.5},
                     };
    GLfloat rV[8][3],r;
    int i;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    r = d*3.14/180;
    if(a == 1)
    {
       for(i=0;i<8;i++)
       {
          rV[i][0] = V[i][0];
          rV[i][1] = V[i][1]*cos(r)-V[i][2]*sin(r);
          rV[i][2] = V[i][1]*sin(r)+V[i][2]*cos(r);
       }
    }

    if(a == 2)
    {
       for(i=0;i<8;i++)
       {
          rV[i][0] = V[i][2]*sin(r)+V[i][0]*cos(r);
          rV[i][1] = V[i][1];
          rV[i][2] = V[i][2]*cos(r)-V[i][0]*sin(r);
       }
    }

    if(a == 3)
    {
       for(i=0;i<8;i++)
       {
          rV[i][0] = V[i][0]*cos(r)-V[i][1]*sin(r);
          rV[i][1] = V[i][0]*sin(r)+V[i][1]*cos(r);
          rV[i][2] = V[i][2];
       }
    }

    Cube(rV[0],rV[1],rV[2],rV[3],rV[4],rV[5],rV[6],rV[7]);

    glutSwapBuffers();
}

int main(int argc, char *argv[])
{
    printf("Enter the Axis of Rotation [ 1->Xaxis | 2->Yaxis | 3->Zaxis ]: ");
    scanf("%d",&a);
    glutInit(&argc,argv);
    glutInitWindowSize(600,600);
    glutInitWindowPosition(50,150);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutCreateWindow("Cube Spin with Matrices");
    MyInit();
    glutDisplayFunc(Draw);
    glutIdleFunc(Spin);
    glutMainLoop();
    return 0;
}
```
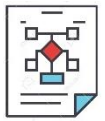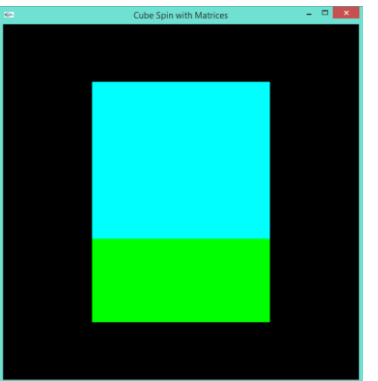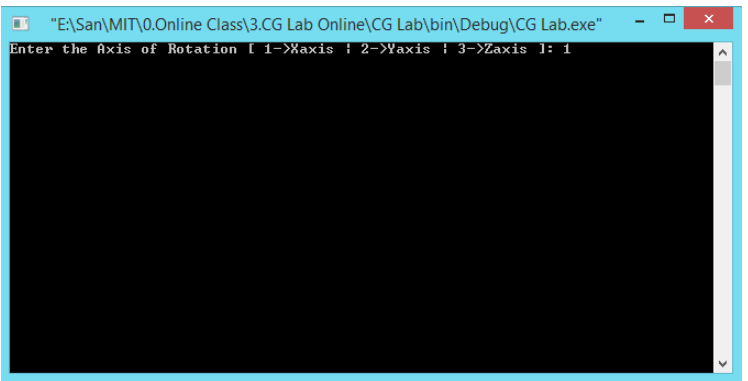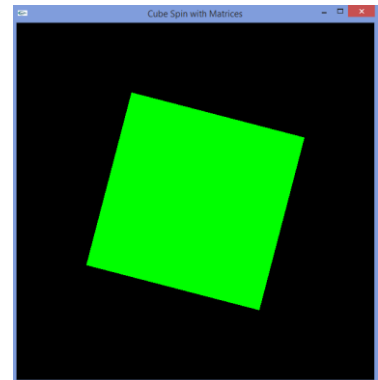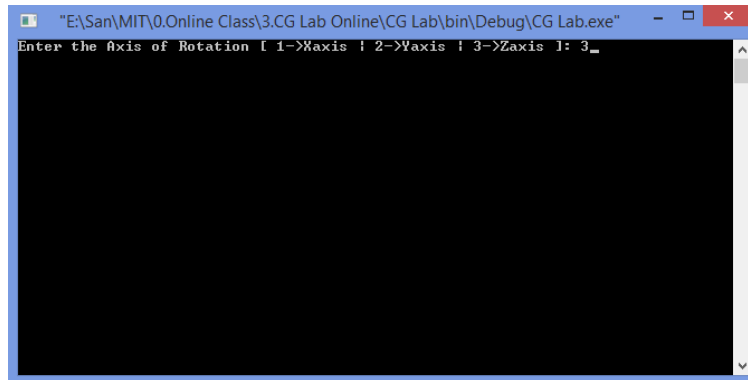
▶	**https://youtu.be/YKX2Np7omJQ**
📄	**Design a Flow Chart or Write the Algorithm (Need to do it yourself)**
💻	

Lab Program - 4

Draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

GLfloat Cx=0,Cy=0,Cz=3;

void MyInit()
{
    glClearColor(0,0,0,1);
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1,1,-1,1,2,10);
    glMatrixMode(GL_MODELVIEW);
}

void Square(GLfloat A[],GLfloat B[],GLfloat C[],GLfloat D[])
{
    glBegin(GL_POLYGON);
        glVertex3fv(A);
        glVertex3fv(B);
        glVertex3fv(C);
        glVertex3fv(D);
    glEnd();
}

void Cube(GLfloat V0[],GLfloat V1[],GLfloat V2[],GLfloat V3[],GLfloat V4[],GLfloat V5[],GLfloat V6[],GLfloat V7[])
{
    glColor3f(1,0,0);
    Square(V0,V1,V2,V3);
    glColor3f(0,1,0);
    Square(V4,V5,V6,V7);
    glColor3f(0,0,1);
    Square(V0,V4,V7,V3);
    glColor3f(1,1,0);
    Square(V1,V5,V6,V2);
    glColor3f(1,0,1);
    Square(V3,V2,V6,V7);
    glColor3f(0,1,1);
    Square(V0,V1,V5,V4);
}

void Draw()
{
    GLfloat V[8][3] =  {
                {-0.5, 0.5, 0.5},
                { 0.5, 0.5, 0.5},
                { 0.5,-0.5, 0.5},
                {-0.5,-0.5, 0.5},
                {-0.5, 0.5,-0.5},
                { 0.5, 0.5,-0.5},
                { 0.5,-0.5,-0.5},
                {-0.5,-0.5,-0.5}
```

```
                };
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();
    gluLookAt(Cx,Cy,Cz,0,0,0,0,1,0);

    Cube(V[0],V[1],V[2],V[3],V[4],V[5],V[6],V[7]);

    glutSwapBuffers();
}

void Key(unsigned char ch,int x,int y)
{
    switch(ch)
    {
        case 'x' : Cx = Cx - 0.5;   break;
        case 'X' : Cx = Cx + 0.5;   break;

        case 'y' : Cy = Cy - 0.5;   break;
        case 'Y' : Cy = Cy + 0.5;   break;

        case 'z' : Cz = Cz - 0.5;   break;
        case 'Z' : Cz = Cz + 0.5;   break;
    }
    glutPostRedisplay();
}

int main(int argC,char *argV[])
{
    glutInit(&argC,argV);
    glutInitWindowSize(600,600);
    glutInitWindowPosition(100,150);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutCreateWindow("Color Cube with Camera");
    MyInit();
    glutDisplayFunc(Draw);
    glutKeyboardFunc(Key);
    glutMainLoop();
    return 0;
}
```
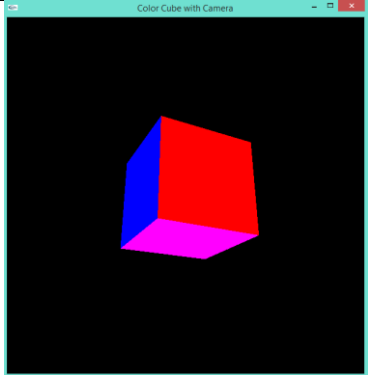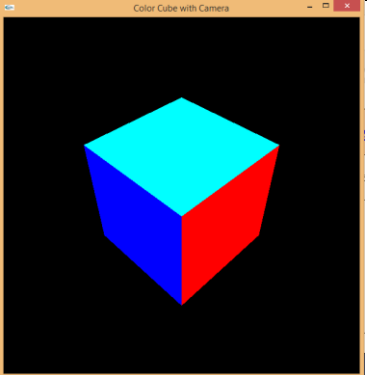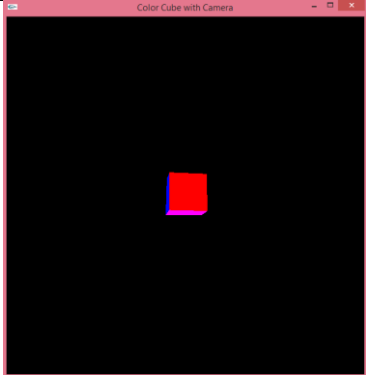
https://youtu.be/elw1WOziruk

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Moving Camera towards Left, Bottom & Away direction	Moving Camera towards Left, Top & Near direction	Moving Camera far away

Lab Program - 5

Clip a Line using Cohen-Sutherland Algorithm.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

GLfloat xMin=-0.5,xMax=0.5,yMin=-0.5,yMax=0.5;
GLfloat x1=-0.4,y1=-0.3,x2=0.2,y2=0.4;

int Left=1,Right=2,Bot=4,Top=8;
int C1,C2;
int Clip_Flag = 0, Flag = 1;;


int Get_Code(GLfloat x,GLfloat y)
{
    int Code = 0;
    if(x<xMin)
        Code = Code | Left;
    if(x>xMax)
        Code = Code | Right;
    if(y<yMin)
        Code = Code | Bot;
    if(y>yMax)
        Code = Code | Top;
    return Code;
}

void Clip()
{
    int C;
    GLfloat x,y;
    if(C1)
        C = C1;
    else
        C = C2;

    if(C & Left)
    {
        x = xMin;
        y = y1+(y2-y1)*((xMin-x1)/(x2-x1));
    }
    if(C & Right)
    {
        x = xMax;
        y = y1+(y2-y1)*((xMax-x1)/(x2-x1));
    }
    if(C & Bot)
    {
        y = yMin;
        x = x1+(x2-x1)*((yMin-y1)/(y2-y1));
    }
    if(C & Top)
    {
        y = yMax;
        x = x1+(x2-x1)*((yMax-y1)/(y2-y1));
    }
```

```
  if(C == C1)
  {
     x1 = x;
     y1 = y;
  }
  else
  {
     x2 = x;
     y2 = y;
  }
}

void Draw()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(1,1,1);
  glBegin(GL_LINE_LOOP);
     glVertex2f(xMin,yMin);
     glVertex2f(xMax,yMin);
     glVertex2f(xMax,yMax);
     glVertex2f(xMin,yMax);
  glEnd();

  glColor3f(1,0,0);
  if(Flag == 1)
  {
   glBegin(GL_LINES);
     glVertex2f(x1,y1);
     glVertex2f(x2,y2);
  glEnd();
  }

  while(1 & Clip_Flag == 1)
  {
     C1 = Get_Code(x1,y1);
     C2 = Get_Code(x2,y2);

     if((C1|C2) == 0)
        break;
     else if((C1&C2)!=0)
     {
        Flag = 0;
        break;
     }
     else
        Clip();
  }
  glFlush();
}

void Key(unsigned char ch,int x,int y)
{
  Clip_Flag = 1;
  glutPostRedisplay();
}

int main(int argC,char *argV[])
{
```
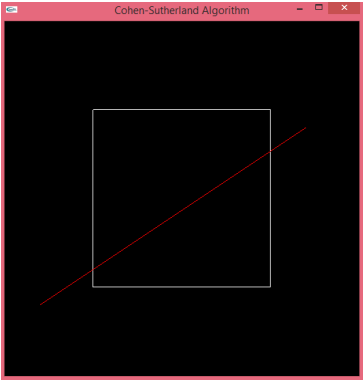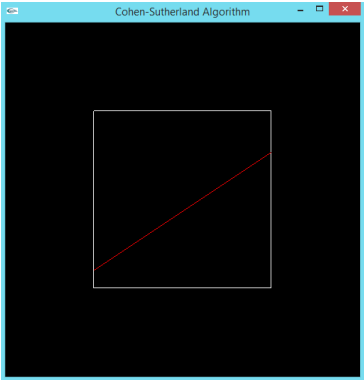
```
glutInit(&argC,argV);
glutInitWindowSize(500,500);
glutInitWindowPosition(100,100);
glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
glutCreateWindow("Cohen-Sutherland Algorithm");
glutDisplayFunc(Draw);
glutKeyboardFunc(Key);
glutMainLoop();
return 0;
}
```

https://youtu.be/ZeTk3O416QM

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Before Clipping   After Clipping

Lab Program - 6

To draw a simple shaded scene consisting of a tea pot on a table. Define suitably the position and properties of the light source along with the properties of the surfaces of the solid object used in the scene.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

GLfloat T = 0;

void Spin()
{
    T = T + 0.1;
    if(T>360)
        T = 0;
    glutPostRedisplay();
}

void Draw()
{
    GLfloat Pos[] = {0,1,0,1};
    GLfloat Col[] = {1,0,0,1};

    GLfloat M[] = {0,1,0,1};

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glLightfv(GL_LIGHT0,GL_POSITION,Pos);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,Col);


    gluLookAt(0,1,3,0,0,0,0,1,0);

    glRotatef(T,0,1,0);

    glPushMatrix();
        glScalef(1,0.05,1);
        glutSolidCube(1);
    glPopMatrix();

    glPushMatrix();
        glTranslatef(-0.5,-0.5,-0.5);
        glScalef(0.05,1,0.05);
        glutSolidCube(1);
    glPopMatrix();

    glPushMatrix();
        glTranslatef(0.5,-0.5,-0.5);
        glScalef(0.05,1,0.05);
        glutSolidCube(1);
    glPopMatrix();

    glPushMatrix();
        glTranslatef(0.5,-0.5,0.5);
        glScalef(0.05,1,0.05);
        glutSolidCube(1);
    glPopMatrix();
```

```
    glPushMatrix();
       glTranslatef(-0.5,-0.5,0.5);
       glScalef(0.05,1,0.05);
       glutSolidCube(1);
    glPopMatrix();

    glPushAttrib(GL_ALL_ATTRIB_BITS);
       glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,M);
       glPushMatrix();
          glTranslatef(0,0.25,0);
          glutSolidTeapot(0.25);
       glPopMatrix();
    glPopAttrib();

    glutSwapBuffers();
}

void MyInit()
{
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1,1,-1,1,2,10);
    glMatrixMode(GL_MODELVIEW);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
}

int main(int argC,char *argV[])
{
    glutInit(&argC,argV);
    glutInitWindowSize(600,600);
    glutInitWindowPosition(100,100);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutCreateWindow("Table & TeaPot");
    MyInit();
    glutDisplayFunc(Draw);
    glutIdleFunc(Spin);

    glutMainLoop();
    return 0;
}
```

https://youtu.be/RzCOInR4ooc

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Lab Program - 7

Design, develop and implement recursively subdivide a tetrahedron to form 3D sierpinski gasket. The number of recursive steps is to be specified by the user.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

int N;

void Triangle(GLfloat A[],GLfloat B[],GLfloat C[])
{
   glBegin(GL_TRIANGLES);
     glVertex3fv(A);
     glVertex3fv(B);
     glVertex3fv(C);
   glEnd();
}

void Tetra(GLfloat V1[],GLfloat V2[],GLfloat V3[],GLfloat V4[])
{
   glColor3f(1,1,1);
   Triangle(V1,V2,V3);
   glColor3f(1,0,0);
   Triangle(V1,V3,V4);
   glColor3f(0,1,0);
   Triangle(V2,V3,V4);
   glColor3f(0,0,1);
   Triangle(V1,V2,V4);
}

void Div(GLfloat V1[],GLfloat V2[],GLfloat V3[],GLfloat V4[],int n)
{
   GLfloat V12[3],V23[3],V31[3],V14[3],V24[3],V34[3];
   if(n>0)
   {
     V12[0] = ( V1[0] + V2[0] ) / 2;     V12[1] = ( V1[1] + V2[1] ) / 2;     V12[2] = ( V1[2] + V2[2] ) / 2;
     V23[0] = ( V2[0] + V3[0] ) / 2;     V23[1] = ( V2[1] + V3[1] ) / 2;     V23[2] = ( V2[2] + V3[2] ) / 2;
     V31[0] = ( V3[0] + V1[0] ) / 2;     V31[1] = ( V3[1] + V1[1] ) / 2;     V31[2] = ( V3[2] + V1[2] ) / 2;
     V14[0] = ( V1[0] + V4[0] ) / 2;     V14[1] = ( V1[1] + V4[1] ) / 2;     V14[2] = ( V1[2] + V4[2] ) / 2;
     V24[0] = ( V2[0] + V4[0] ) / 2;     V24[1] = ( V2[1] + V4[1] ) / 2;     V24[2] = ( V2[2] + V4[2] ) / 2;
     V34[0] = ( V3[0] + V4[0] ) / 2;     V34[1] = ( V3[1] + V4[1] ) / 2;     V34[2] = ( V3[2] + V4[2] ) / 2;

     Div(V1,V12,V31,V14,n-1);
     Div(V12,V2,V23,V24,n-1);
     Div(V31,V23,V3,V34,n-1);
     Div(V14,V24,V34,V4,n-1);
   }
   else
     Tetra(V1,V2,V3,V4);
}

void Draw()
{
   GLfloat P[4][3] =  {
               {-0.65,-0.5, 0.5},
               { 0.65,-0.5, 0.5},
               { 0  , 0.6, 0.5},
               { 0  ,-0.05,-0.5},
```

```
                };
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        Div(P[0],P[1],P[2],P[3],N);

        glutSwapBuffers();
}

int main(int argC,char *argV[])
{
        printf("Enter the Number of Division Steps : ");
        scanf("%d",&N);
        glutInit(&argC,argV);
        glutInitWindowSize(600,600);
        glutInitWindowPosition(100,100);
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
        glutCreateWindow("3D Sierpinski Gasket");
        glutDisplayFunc(Draw);

        glutMainLoop();
        return 0;
}
```
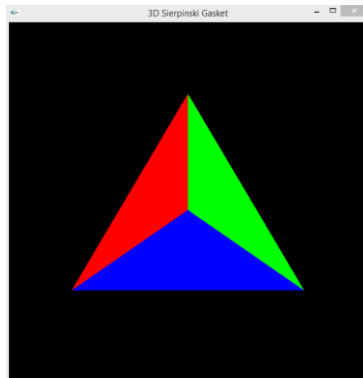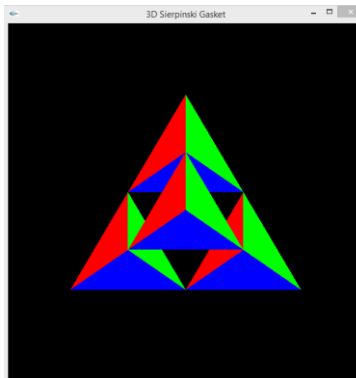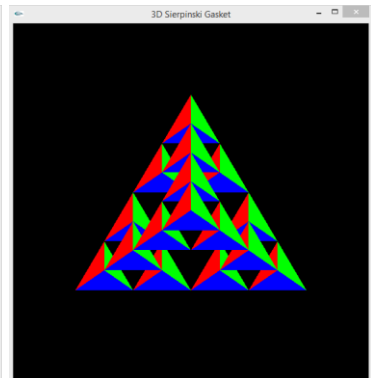
https://youtu.be/H1RtMA3XV3k

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

0 Division Step   1 Division Step   2 Division Step

Lab Program - 8

Develop a menu driven program to animate a flag using Bezier Curve algorithm.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

GLfloat ya = 50,xa = 10;
int yFlag = 1, xFlag = 1,AniFlag=1;

void Animate()
{
   if(AniFlag == 1)
   {
      if(ya>-50 && yFlag == 1)
         ya = ya - 0.2;

      if(ya<=-50 && yFlag == 1)
         yFlag = 0;

      if(ya<50 && yFlag == 0)
         ya = ya + 0.2;

      if(ya>=50 && yFlag == 0)
         yFlag = 1;


      if(xa>-10 && xFlag == 1)
         xa = xa - 0.2;

      if(xa<=-10 && xFlag == 1)
         xFlag = 0;

      if(xa<10 && xFlag == 0)
         xa = xa + 0.2;

      if(xa>=10 && xFlag == 0)
         xFlag = 1;
   }
   glutPostRedisplay();
}

void Draw()
{
   GLfloat x[4],y1[4],y2[4],y3[4],y4[4];
   GLdouble xt[200],y1t[200],y2t[200],y3t[200],y4t[200],t;
   int i;
   glClear(GL_COLOR_BUFFER_BIT);

   x[0] = 100; x[1] = 200; x[2] = 200;  x[3] = 300-xa;
   y1[0] = 450; y1[1] = 450+ya; y1[2] = 450-ya;  y1[3] = 450;
   y2[0] = 400; y2[1] = 400+ya; y2[2] = 400-ya;  y2[3] = 400;
   y3[0] = 350; y3[1] = 350+ya; y3[2] = 350-ya;  y3[3] = 350;
   y4[0] = 300; y4[1] = 300+ya; y4[2] = 300-ya;  y4[3] = 300;

   for(i=0,t=0;t<1;i++,t=t+0.01)
   {
      xt[i] = pow(1-t,3)*x[0]+3*t*pow(1-t,2)*x[1]+3*pow(t,2)*(1-t)*x[2]+pow(t,3)*x[3];
```

```
            y1t[i] = pow(1-t,3)*y1[0]+3*t*pow(1-t,2)*y1[1]+3*pow(t,2)*(1-t)*y1[2]+pow(t,3)*y1[3];
            y2t[i] = pow(1-t,3)*y2[0]+3*t*pow(1-t,2)*y2[1]+3*pow(t,2)*(1-t)*y2[2]+pow(t,3)*y2[3];
            y3t[i] = pow(1-t,3)*y3[0]+3*t*pow(1-t,2)*y3[1]+3*pow(t,2)*(1-t)*y3[2]+pow(t,3)*y3[3];
            y4t[i] = pow(1-t,3)*y4[0]+3*t*pow(1-t,2)*y4[1]+3*pow(t,2)*(1-t)*y4[2]+pow(t,3)*y4[3];
        }

    glColor3f(1,0.25,0);
    glBegin(GL_QUAD_STRIP);
        for(i=0;i<200;i++)
        {
            glVertex2d(xt[i],y1t[i]);
            glVertex2d(xt[i],y2t[i]);
        }
    glEnd();

    glColor3f(1,1,1);
    glBegin(GL_QUAD_STRIP);
        for(i=0;i<200;i++)
        {
            glVertex2d(xt[i],y2t[i]);
            glVertex2d(xt[i],y3t[i]);
        }
    glEnd();

    glColor3f(0.1,0.5,0.1);
    glBegin(GL_QUAD_STRIP);
        for(i=0;i<200;i++)
        {
            glVertex2d(xt[i],y3t[i]);
            glVertex2d(xt[i],y4t[i]);
        }
    glEnd();

    glColor3f(0.6,0.6,0.3);
    glRecti(90,460,100,50);

    glFlush();
}

void Menu(int n)
{
    if(n == 1)
        AniFlag = 1;
    else if(n == 2)
        AniFlag = 0;
    if(n == 3)
        exit(0);

    glutPostRedisplay();
}

void MyInit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,500,0,500);
    glMatrixMode(GL_MODELVIEW);

    glutCreateMenu(Menu);
    glutAddMenuEntry("Start",1);
```

```
      glutAddMenuEntry("Stop",2);
      glutAddMenuEntry("Exit",3);
      glutAttachMenu(GLUT_RIGHT_BUTTON);
}

int main(int argC,char *argV[])
{
      glutInit(&argC,argV);
      glutInitWindowSize(500,500);
      glutInitWindowPosition(100,100);
      glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
      glutCreateWindow("Animate Flag");
      MyInit();
      glutDisplayFunc(Draw);
      glutIdleFunc(Animate);
      glutMainLoop();
      return 0;
}
```

https://youtu.be/ZXuqWb133-Y

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Lab Program - 9

Develop a menu driven program to fill the polygon using scan line algorithm.

```c
#include<windows.h>
#include<GL/glu.h>
#include<GL/glut.h>

int LE[500],RE[500];
int EdgeFlag = 0,FillFlag = 0;

void Intersection(GLint x1,GLint y1,GLint x2,GLint y2)
{
    float x,M;
    int t,y;
    if(y1>y2)
    {
        t = x1;
        x1 = x2;
        x2 = t;

        t = y1;
        y1 = y2;
        y2 = t;
    }

    if((y2-y1)==0)
        M = (x2-x1);
    else
        M = (x2-x1)/(y2-y1);

    x = x1;
    for(y=y1;y<=y2;y++)
    {
        if(x<LE[y])
            LE[y]=x;
        if(x>RE[y])
            RE[y]=x;

        x = x + M;
    }
}

void Draw()
{
    int x,y,i;
    GLint P1[2] = {125,250},P2[2] = {250,125},P3[2] = {375,250},P4[2] = {250,375};
    glClear(GL_COLOR_BUFFER_BIT);

    for(i=0;i<500;i++)
    {
        LE[i] = 500;
        RE[i] = 0;
    }

    if(EdgeFlag == 1)
    {
        glBegin(GL_LINE_LOOP);
          glVertex2iv(P1);
```

```
            glVertex2iv(P2);
            glVertex2iv(P3);
            glVertex2iv(P4);
          glEnd();
        }

    Intersection(P1[0],P1[1],P2[0],P2[1]);
    Intersection(P2[0],P2[1],P3[0],P3[1]);
    Intersection(P3[0],P3[1],P4[0],P4[1]);
    Intersection(P4[0],P4[1],P1[0],P1[1]);

    if(FillFlag == 1)
    {
      for(y=0;y<500;y++)
        {
          for(x=LE[y];x<RE[y];x++)
          {
            glBegin(GL_POINTS);
              glVertex2i(x,y);
            glEnd();
            glFlush();
          }
        }
    }


    glFlush();
}

void Menu(int id)
{
    if(id == 1)
        EdgeFlag = 1;
    else if(id == 2)
        EdgeFlag = 0;
    else if(id == 3)
        exit(0);

    FillFlag = 1;
    glutPostRedisplay();
}

void MyInit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0,500,0,500);
    glMatrixMode(GL_MODELVIEW);

    glutCreateMenu(Menu);
    glutAddMenuEntry("With Edge",1);
    glutAddMenuEntry("Without Edge",2);
    glutAddMenuEntry("Exit",3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}

int main(int argC,char *argV[])
{
    glutInit(&argC,argV);
    glutInitWindowSize(500,500);
```
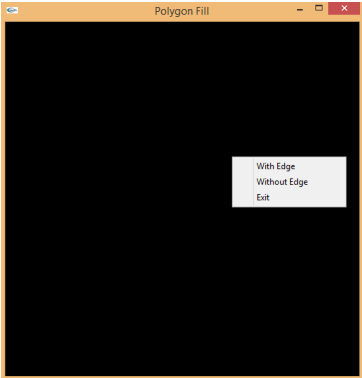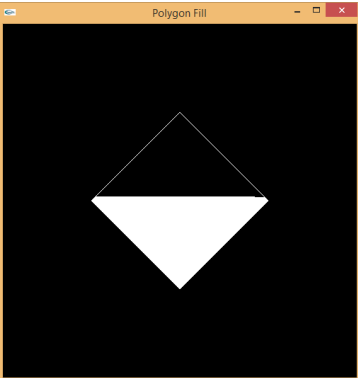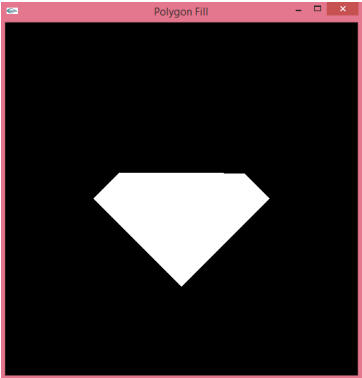
```
    glutInitWindowPosition(100,100);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutCreateWindow("Polygon Fill");
    MyInit();
    glutDisplayFunc(Draw);
    glutMainLoop();
    return 0;
}
```

https://youtu.be/j-sI3DweFdI

Design a Flow Chart or Write the Algorithm (Need to do it yourself)

Prof. Santosh E, Asst. Prof., Dept. of CS&E, MIT Mysore

Page | 25