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

void TriRenderFunc();
void RhoRenderFunc();
void init_mat_tri();
void init_mat_rho();
void DDA(int,int,int,int);
void matmul();
void matmul_rho();
void translate_Tri();
void translate_Rho();
void rotate_Tri();
void rotate_Rho();
void shear_Tri();
void shear_Rho();
void scale_Tri();
void scale_Rho();

int orig[10][3];
float trans[3][3], final[10][3];

int main(int argc, char**argv)
{
   int choice;

   glutInit(&argc, argv);
```

```
   glutInitDisplayMode(GLUT_SINGLE);
   glutInitWindowSize(500, 500);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("Assignment 6");

   printf("\nChoose from the following: ");
   printf("\n1. Equilateral Triangle");
   printf("\n2. Rhombus");
   printf("\n3. Exit");
   printf("\nEnter your choice: ");
   scanf("%d", &choice);
   switch(choice)
   {
      case 1:
           glutDisplayFunc(TriRenderFunc);
         glutMainLoop();
         break;
      case 2:
           glutDisplayFunc(RhoRenderFunc);
         glutMainLoop();
         break;
      case 3:
              return 0;
           }
           return 0;
}

void TriRenderFunc()
{
```

```
int choice;
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
gluOrtho2D(-1000, 1000, -1000, 1000);

glColor3f(0.5, 0.5, 0.5);
DDA(0, -1000, 0, 1000);
DDA( -1000, 0, 1000, 0);

//original Figure
init_mat_tri();
glColor3f(0.0, 1.0, 0.0);
DDA(orig[0][0], orig[0][1], orig[1][0], orig[1][1]);
DDA(orig[1][0], orig[1][1], orig[2][0], orig[2][1]);
DDA(orig[0][0], orig[0][1], orig[2][0], orig[2][1]);



printf("\nChoose from the following: ");
printf("\n1. Translation");
printf("\n2. Rotation");
printf("\n3. Shear ");
printf("\n4. Scaling ");
printf("\nEnter your choice: ");
scanf("%d", &choice);
switch(choice)
{
   case 1: translate_Tri();
      break;
   case 2: rotate_Tri();
```

```
            break;
        case 3: shear_Tri();
            break;
        case 4: scale_Tri();
            break;
    }

}

void RhoRenderFunc()
{
    int choice;
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    gluOrtho2D(-1000, 1000, -1000, 1000);

    glColor3f(0.5, 0.5, 0.5);
    DDA(0, -1000, 0, 1000);
    DDA( -1000, 0, 1000, 0);

    //original Figure
    init_mat_rho();
    glColor3f(0.0, 1.0, 0.0);
    DDA(orig[0][0], orig[0][1], orig[1][0], orig[1][1]);
    DDA(orig[1][0], orig[1][1], orig[2][0], orig[2][1]);
    DDA(orig[3][0], orig[3][1], orig[2][0], orig[2][1]);
    DDA(orig[0][0], orig[0][1], orig[3][0], orig[3][1]);
```

```
    printf("\nChoose from the following: ");
    printf("\n1. Translation");
    printf("\n2. Rotation");
    printf("\n3. Shear ");
    printf("\n4. Scaling ");
    printf("\nEnter your choice: ");
    scanf("%d", &choice);
    switch(choice)
    {
        case 1: translate_Rho();
            break;
        case 2: rotate_Rho();
            break;
        case 3: shear_Rho();
            break;
        case 4: scale_Rho();
            break;
    }
}

void init_mat_tri()
{
    int i, size, x, y, j,length;

    for(i=0;i<10;i++)
    {
        orig[i][2] = 1;
        final[i][2] = 1;
    }
```

```
    orig[0][0] = 500;
    orig[0][1] = 500;
    orig[1][0] = 0;
    orig[1][1] = -500;
    orig[2][0] = 1000;
    orig[2][1] = -500;
}

void init_mat_rho()
{
    int i;

    for(i=0;i<10;i++)
    {
        orig[i][2] = 1;
        final[i][2] = 1;
    }

    orig[0][0] = 100;
    orig[0][1] = 0;
    orig[1][0] = 500;
    orig[1][1] = 0;
    orig[2][0] = 700;
    orig[2][1] = 700;
    orig[3][0] = 300;
    orig[3][1] = 700;
}
```

```
void DDA(int x1, int y1, int x2, int y2)
{
   int dx, dy, i;
   float xinc, yinc, x, y, steps;

   x = x1;
   y = y1;
   dx = x2 - x1;
   dy = y2 - y1;
   if(dx > dy)
      steps = abs(dx);
   else
      steps = abs(dy);
   xinc = dx/steps;
   yinc = dy/steps;

   glBegin(GL_POINTS);
   glVertex2i(x, y);
   glEnd();

   for(i =0; i < (steps - 1); i++)
   {
      x += xinc;
      y += yinc;
      glBegin(GL_POINTS);
      glVertex2i(x, y);
      glEnd();
   }
```

```
   glFlush();

}

void matmul()
{
   int i, j, k;
   //init_mat();

   for(i=0;i<4;i++)
      for(j=0;j<3;j++)
         final[i][j] = 0;

   for (i=0; i< 4; i++)
   {
      for(j=0; j< 3; j++)
      {
         final[i][j] = 0;
         for (k = 0; k< 3; k++)
         {
            final[i][j] += (orig[i][k] * trans[k][j]);
         }
      }
   }

}

void matmul_rho()
{
```

```
   int i, j, k;
   //init_mat();

   for(i=0;i<5;i++)
      for(j=0;j<3;j++)
         final[i][j] = 0;

   for (i=0; i< 5; i++)
   {
      for(j=0; j< 3; j++)
      {
         final[i][j] = 0;
         for (k = 0; k< 3; k++)
         {
            final[i][j] += (orig[i][k] * trans[k][j]);
         }
      }
   }

}

void translate_Tri()
{
   int i, j, tx, ty;

   printf("\nEnter translation factor in x direction: ");
   scanf("%d", &tx);
   printf("\nEnter translation factor in y direction: ");
   scanf("%d", &ty);
```

```
    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            trans[i][j] = 0;

    for(i=0;i<3;i++)
        trans[i][i] = 1;



    trans[2][0] = tx;
    trans[2][1] = ty;

    matmul();

    glColor3f(1.0, 1.0, 0.0);
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[2][0], final[2][1]);

}

void translate_Rho()
{
    int i, j, tx, ty;

    printf("\nEnter translation factor in x direction: ");
    scanf("%d", &tx);
    printf("\nEnter translation factor in y direction: ");
```

```
    scanf("%d", &ty);

    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            trans[i][j] = 0;

    for(i=0;i<3;i++)
        trans[i][i] = 1;



    trans[2][0] = tx;
    trans[2][1] = ty;

    matmul_rho();

    glColor3f(1.0, 1.0, 0.0);
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[3][0], final[3][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[3][0], final[3][1]);

}

void rotate_Tri()
{
    int i, j;
    float d, rx, ry;
```

```
    printf("\nEnter angle in degrees: ");
    scanf("%f", &d);
    d = (float)(3.14*d/180);
    //rx = cos(d);
    //ry = sin(d);

    rx = 1.414;
    ry = 1.414;
    for(i=0;i<3;i++)
       for(j=0;j<3;j++)
           trans[i][j] = 0;

    trans[0][0] = cos(d);
    trans[0][1] = sin(d);
    trans[1][0] = (-cos(d));
    trans[1][1] = sin(d);
    trans[2][2] = 1;

    matmul();

    glColor3f(1.0, 1.0, 0.0);
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[2][0], final[2][1]);

}

void rotate_Rho()
{
```

```
int i, j;
float d, rx, ry;

printf("\nEnter angle in degrees: ");
scanf("%f", &d);
d = (float)(3.14*d/180);
//rx = cos(d);
//ry = sin(d);

rx = 1.414;
ry = 1.414;
for(i=0;i<3;i++)
   for(j=0;j<3;j++)
      trans[i][j] = 0;

trans[0][0] = cos(d);
trans[0][1] = sin(d);
trans[1][0] = (-cos(d));
trans[1][1] = sin(d);
trans[2][2] = 1;

matmul_rho();

glColor3f(1.0, 1.0, 0.0);
DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
DDA(final[3][0], final[3][1], final[2][0], final[2][1]);
DDA(final[0][0], final[0][1], final[3][0], final[3][1]);
```

```
}
void shear_Tri()
{
   int i, j, tx, ty;

   printf("\nEnter shearing factor in x direction: ");
   scanf("%d", &tx);
   printf("\nEnter shearing factor in y direction: ");
   scanf("%d", &ty);

   for(i=0;i<3;i++)
      for(j=0;j<3;j++)
         trans[i][j] = 0;

   for(i=0; i<3; i++)
      trans[i][i] = 1;

   trans[1][0] = tx;
   trans[0][1] = ty;

   //matmul();
   final[0][0] = 600;
   final[0][1] = 500;
   final[1][0] = 0;
   final[1][1] = -500;
   final[2][0] = 1000;
   final[2][1] = -500;

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

```
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[2][0], final[2][1]);

}

void shear_Rho()
{
    int i, j, tx, ty;

    printf("\nEnter shearing factor in x direction: ");
    scanf("%d", &tx);
    printf("\nEnter shearing factor in y direction: ");
    scanf("%d", &ty);

    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            trans[i][j] = 0;

    for(i=0; i<3; i++)
        trans[i][i] = 1;

    trans[1][0] = tx;
    trans[0][1] = ty;

    //matmul_rho();
    for(i=0;i<5;i++)
        for(j=0;j<2;j++)
            final[i][j] = 0;
```

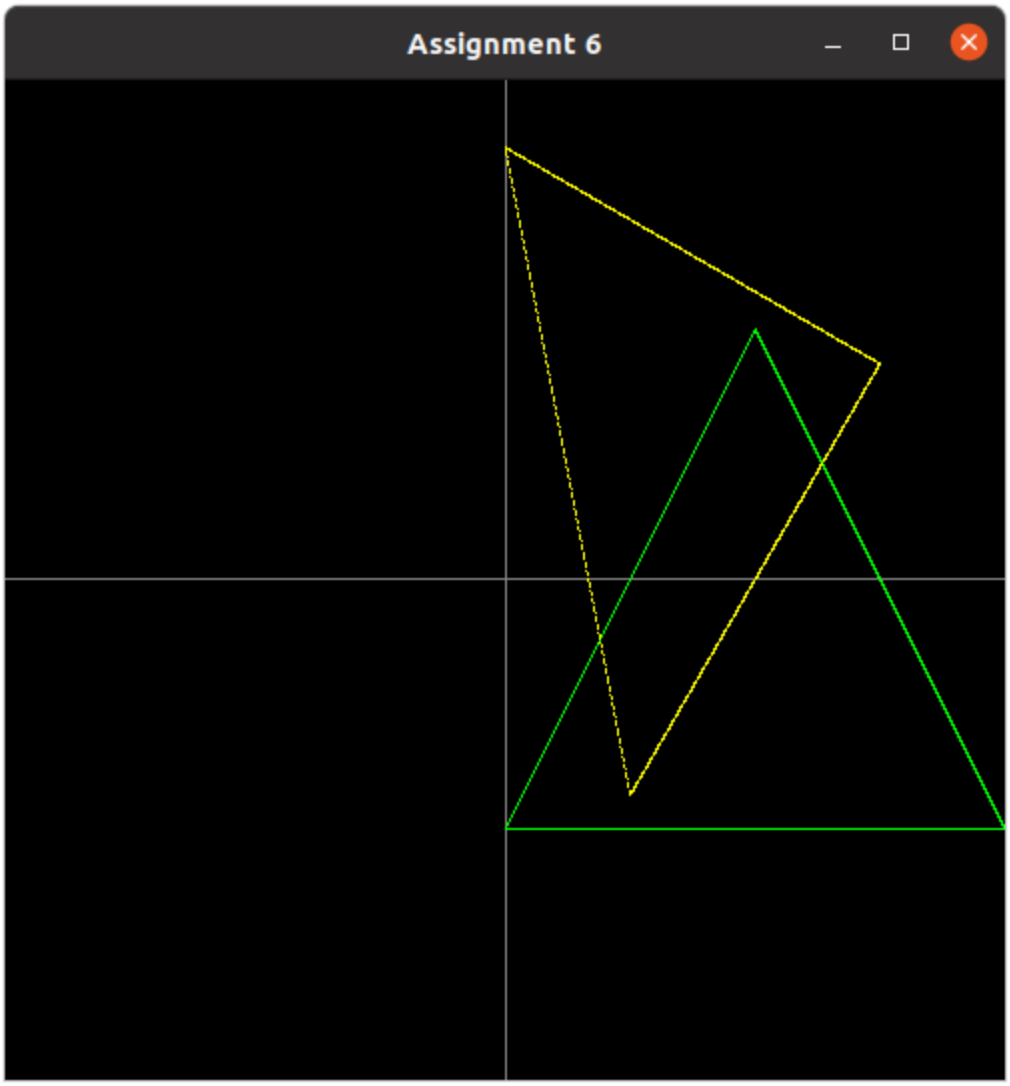
```
for(i=0;i<5;i++)
   final[i][2] = 1;

//in x direction
final[0][0] = 100;
final[0][1] = 0;
final[1][0] = 500;
final[1][1] = 0;
final[2][0] = 800;
final[2][1] = 700;
final[3][0] = 400;
final[3][1] = 700;


glColor3f(1.0, 0.5, 1.0);
DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
DDA(final[3][0], final[3][1], final[2][0], final[2][1]);
DDA(final[0][0], final[0][1], final[3][0], final[3][1]);

//in y direction
final[0][0] = 100;
final[0][1] = 0;
final[1][0] = 500;
final[1][1] = 100;
final[2][0] = 700;
final[2][1] = 800;
final[3][0] = 300;
```

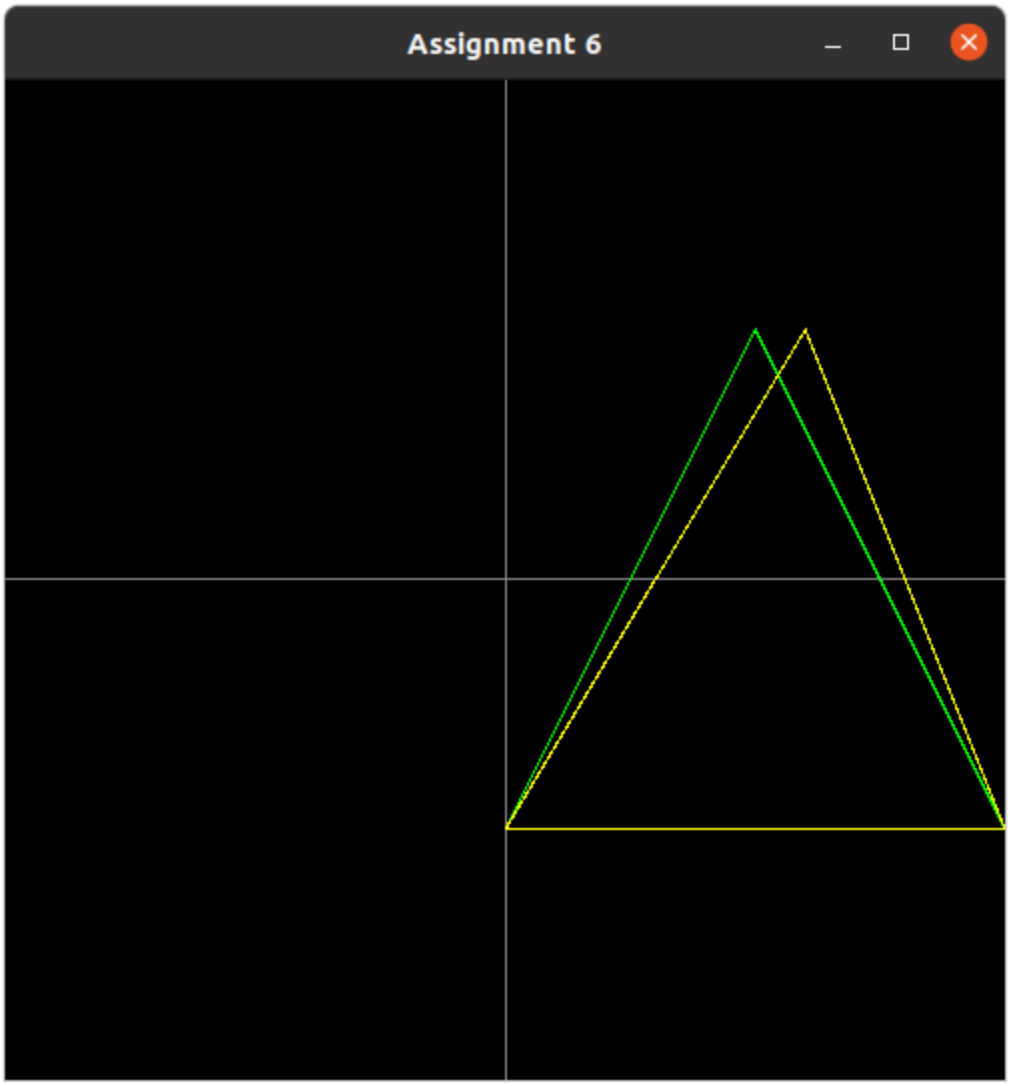
```
    final[3][1] = 700;



    glColor3f(1.0, 1.0, 0.0);
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[3][0], final[3][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[3][0], final[3][1]);

}
void scale_Tri()
{
    int i, j;
    float tx, ty;

    printf("\nEnter scaling factor in x direction: ");
    scanf("%f", &tx);
    printf("\nEnter scaling factor in y direction: ");
    scanf("%f", &ty);

    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            trans[i][j] = 0;

    trans[0][0] = tx;
    trans[1][1] = ty;
    trans[2][2] = 1;

    matmul();
```

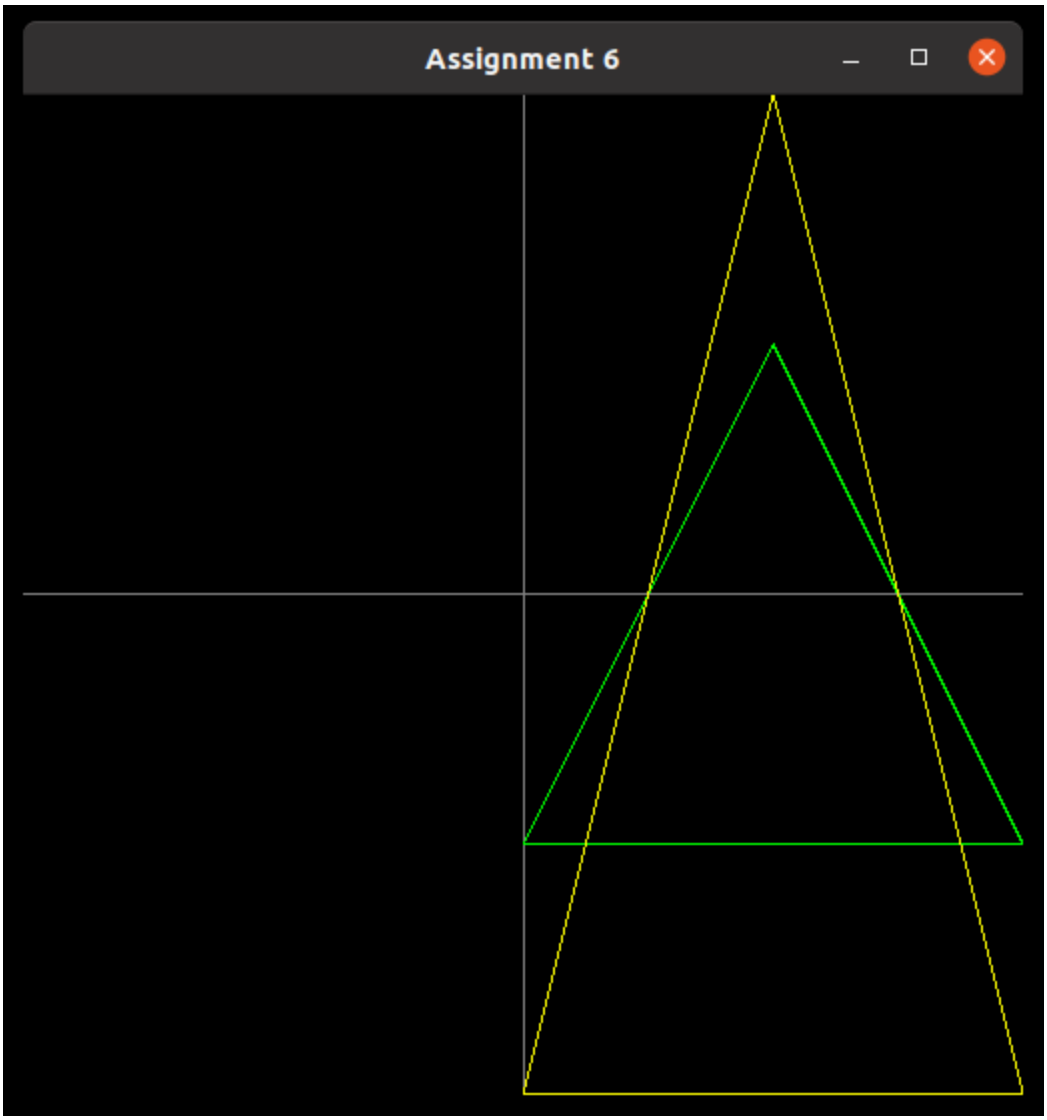
```
    glColor3f(1.0, 1.0, 0.0);
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[2][0], final[2][1]);

}

void scale_Rho()
{
    int i, j;
    float tx, ty;

    printf("\nEnter scaling factor in x direction: ");
    scanf("%f", &tx);
    printf("\nEnter scaling factor in y direction: ");
    scanf("%f", &ty);

    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            trans[i][j] = 0;

    trans[0][0] = tx;
    trans[1][1] = ty;
    trans[2][2] = 1;

    matmul_rho();
```

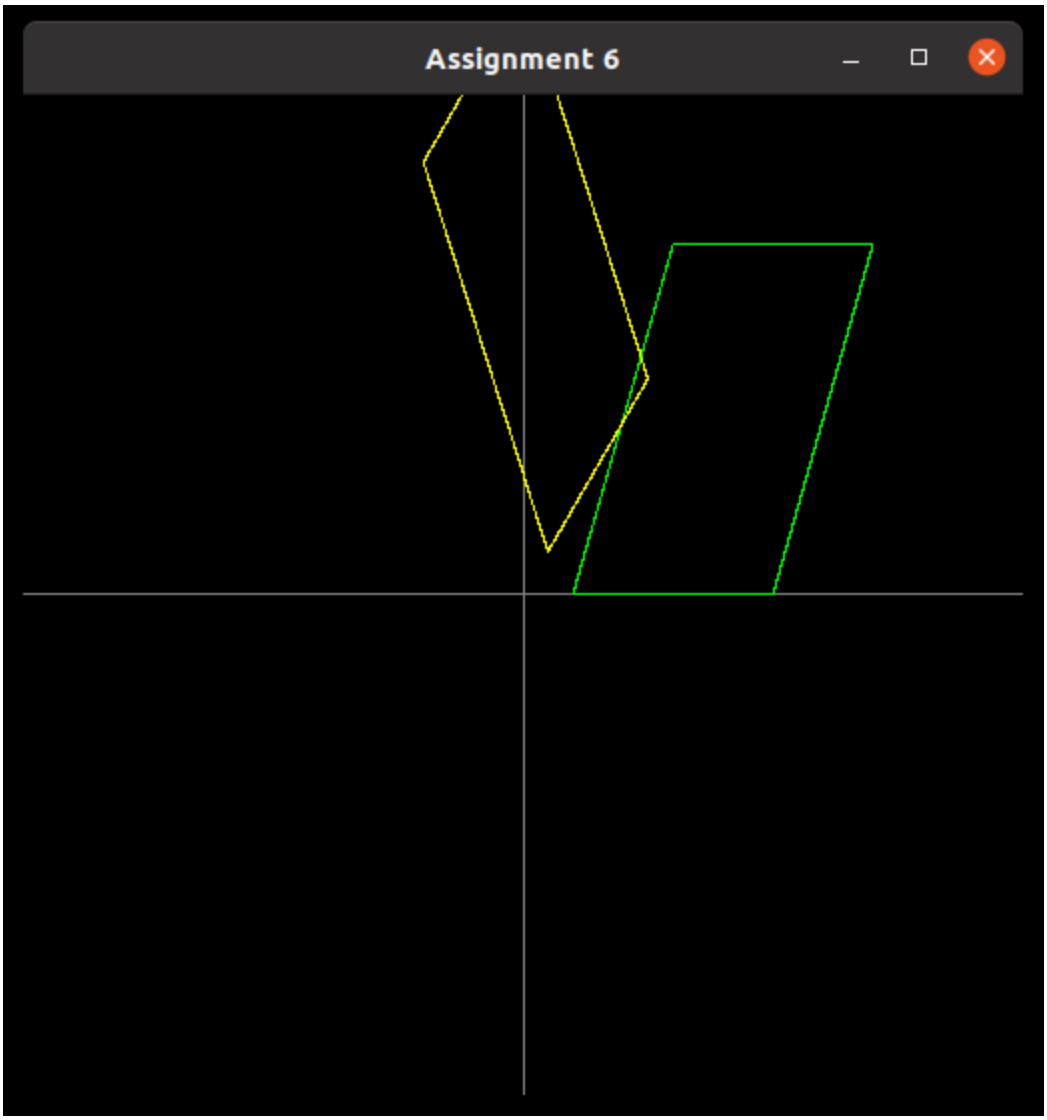
```
    glColor3f(1.0, 1.0, 0.0);
    DDA(final[0][0], final[0][1], final[1][0], final[1][1]);
    DDA(final[1][0], final[1][1], final[2][0], final[2][1]);
    DDA(final[3][0], final[3][1], final[2][0], final[2][1]);
    DDA(final[0][0], final[0][1], final[3][0], final[3][1]);

}
```

OUTPUT

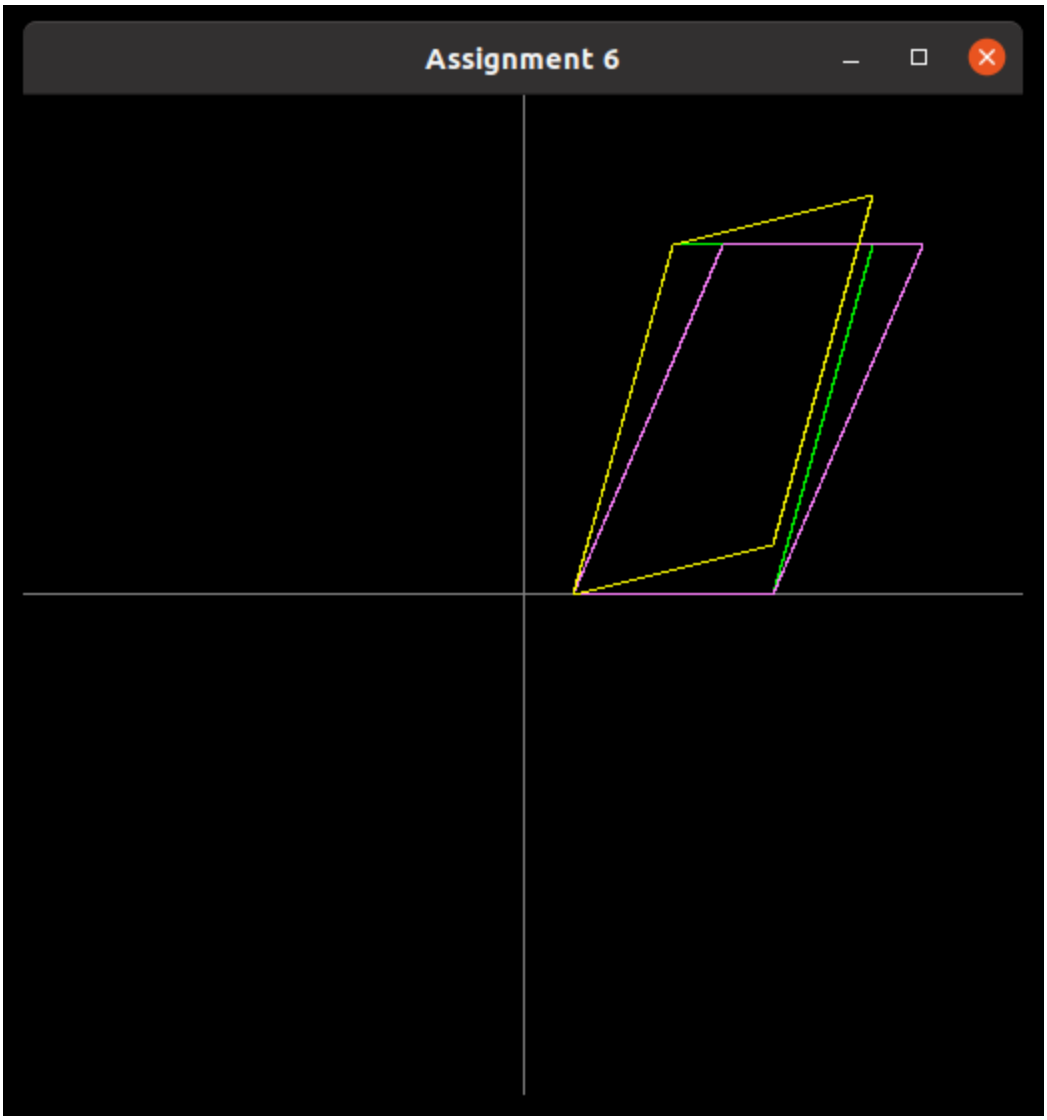

Assignment 6

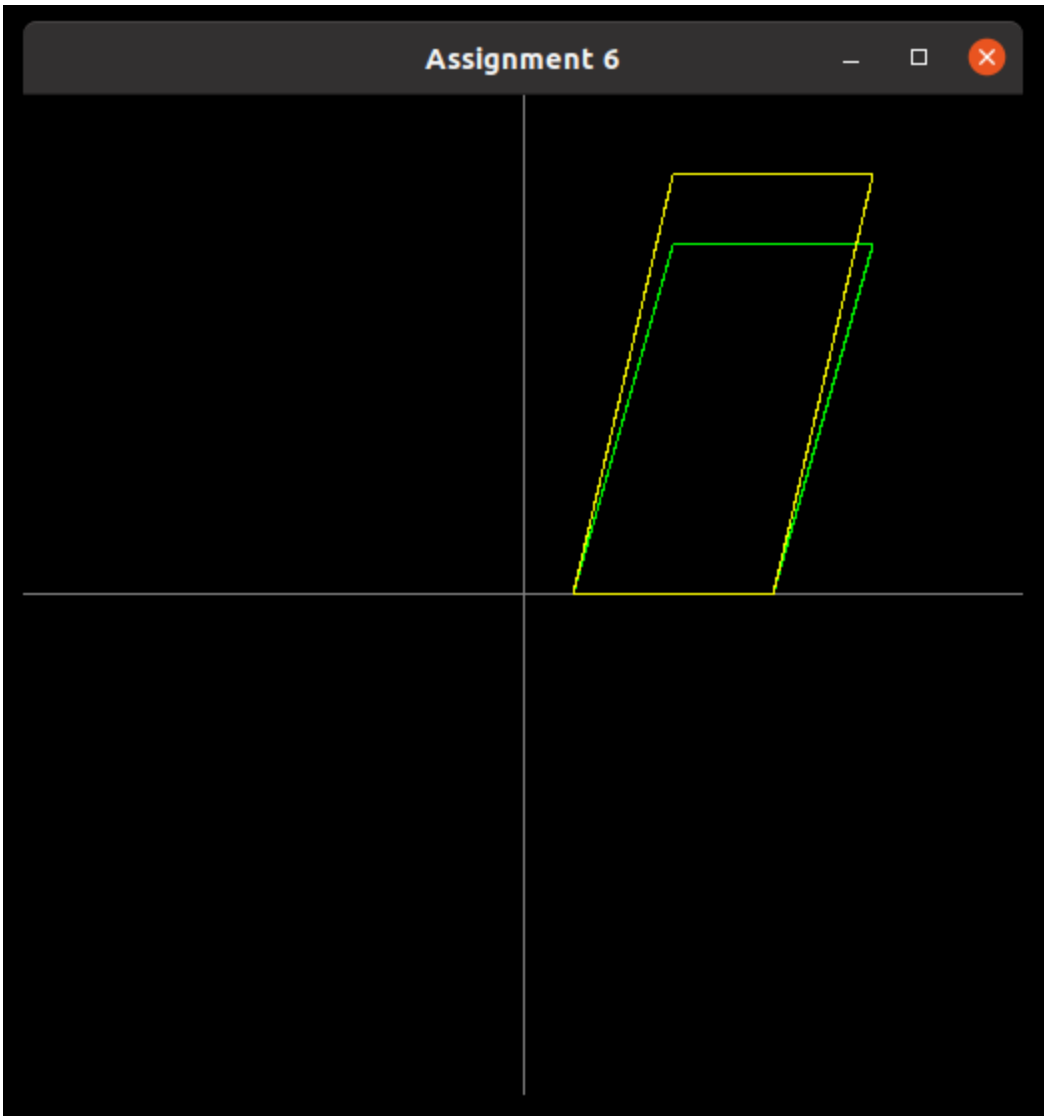

Assignment 6


Assignment 6


Assignment 6


Assignment 6


Assignment 6


Assignment 6