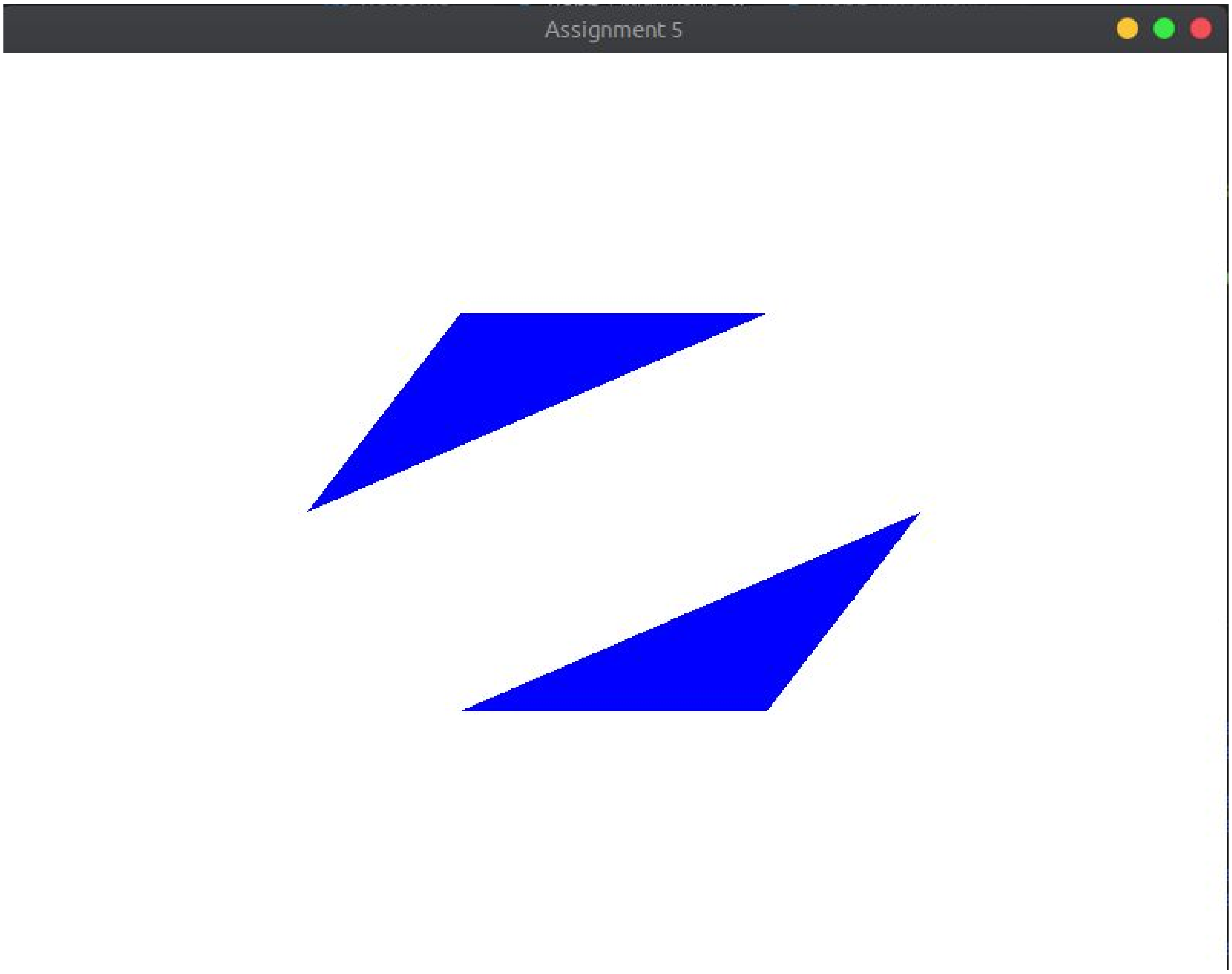
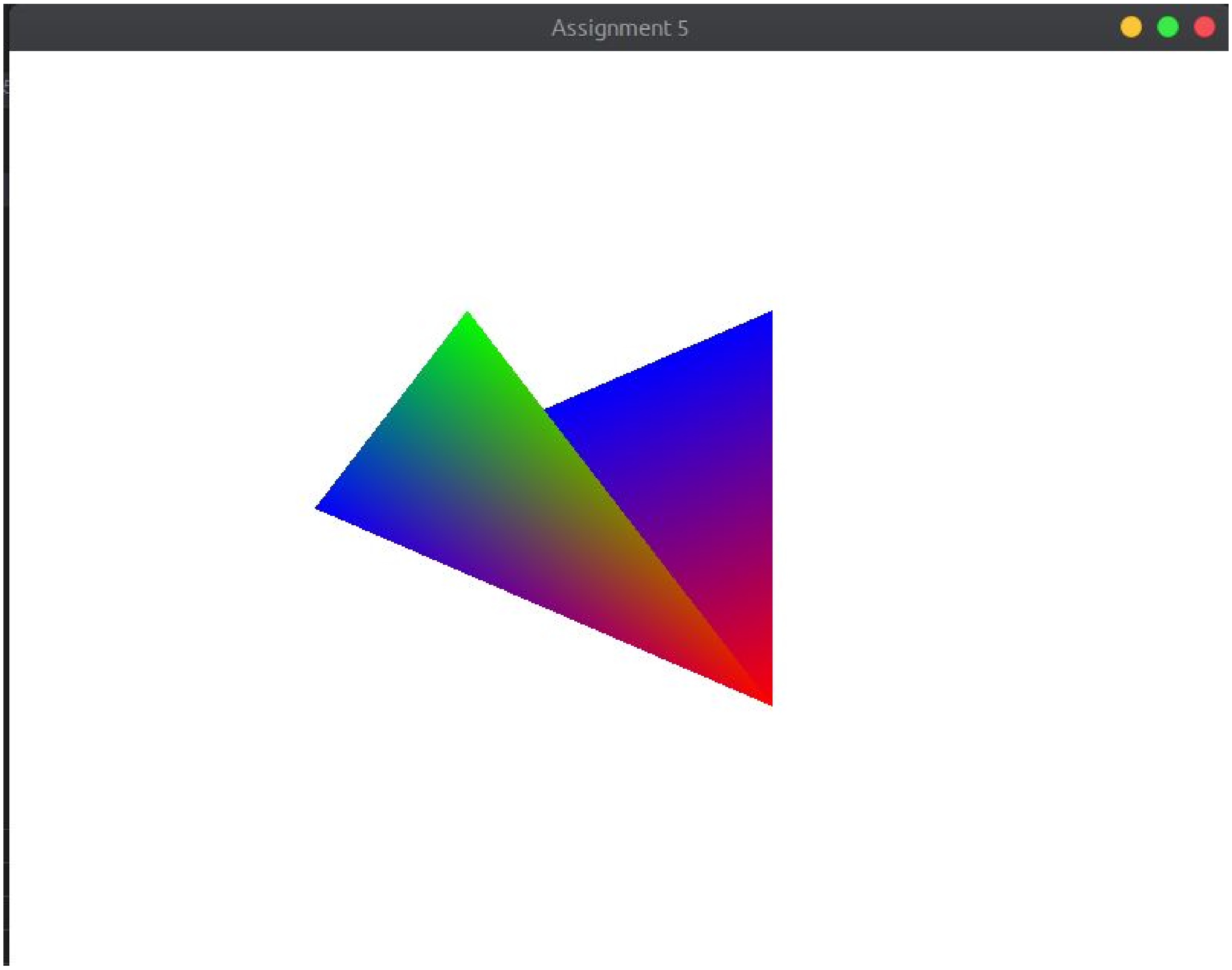
Computer Graphics Practicals
Assignment 5

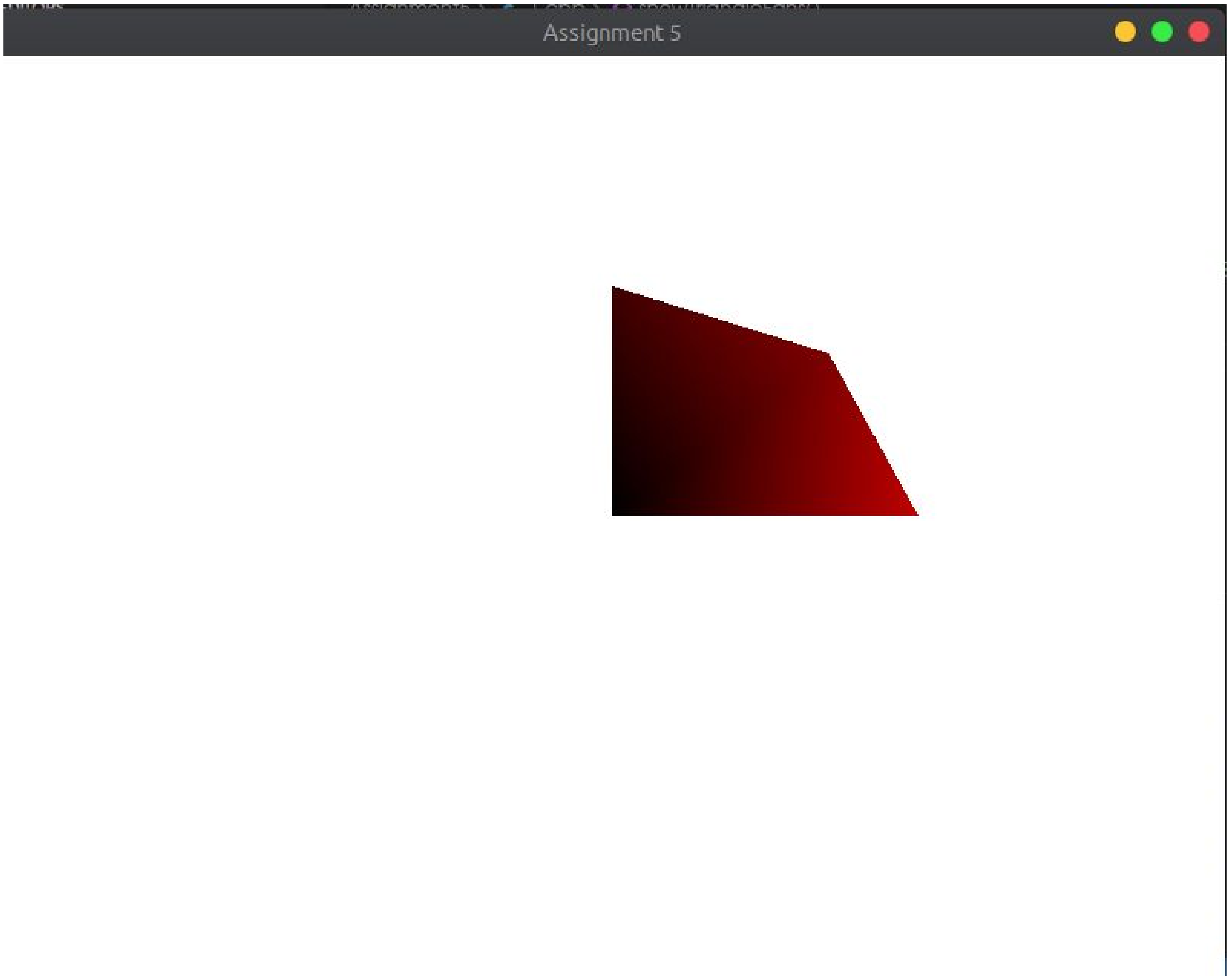
U18CO081
Krunal Rank

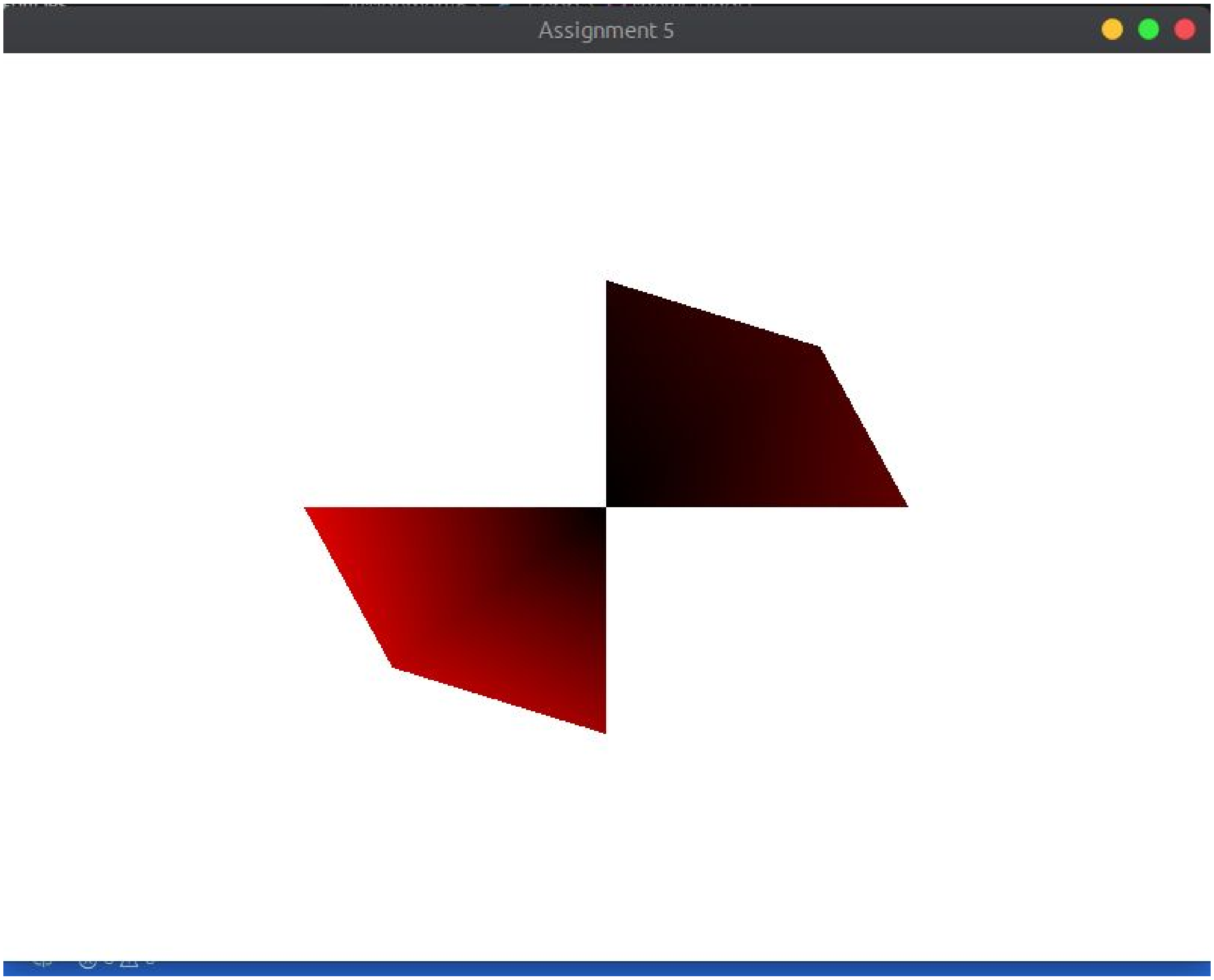
Write a program to draw the following shapes:
1. Triangles (triples of vertices interpreted as triangles)
2. Triangle Strip (linked strip of triangles)
3. Triangle Fan (linked fan of triangles)
4. Quads (quadruples of vertices interpreted as four sided polygons)
5. Quad Strip (linked strip of quadrilaterals)
6. Polygon (boundary of a simple, convex polygon)

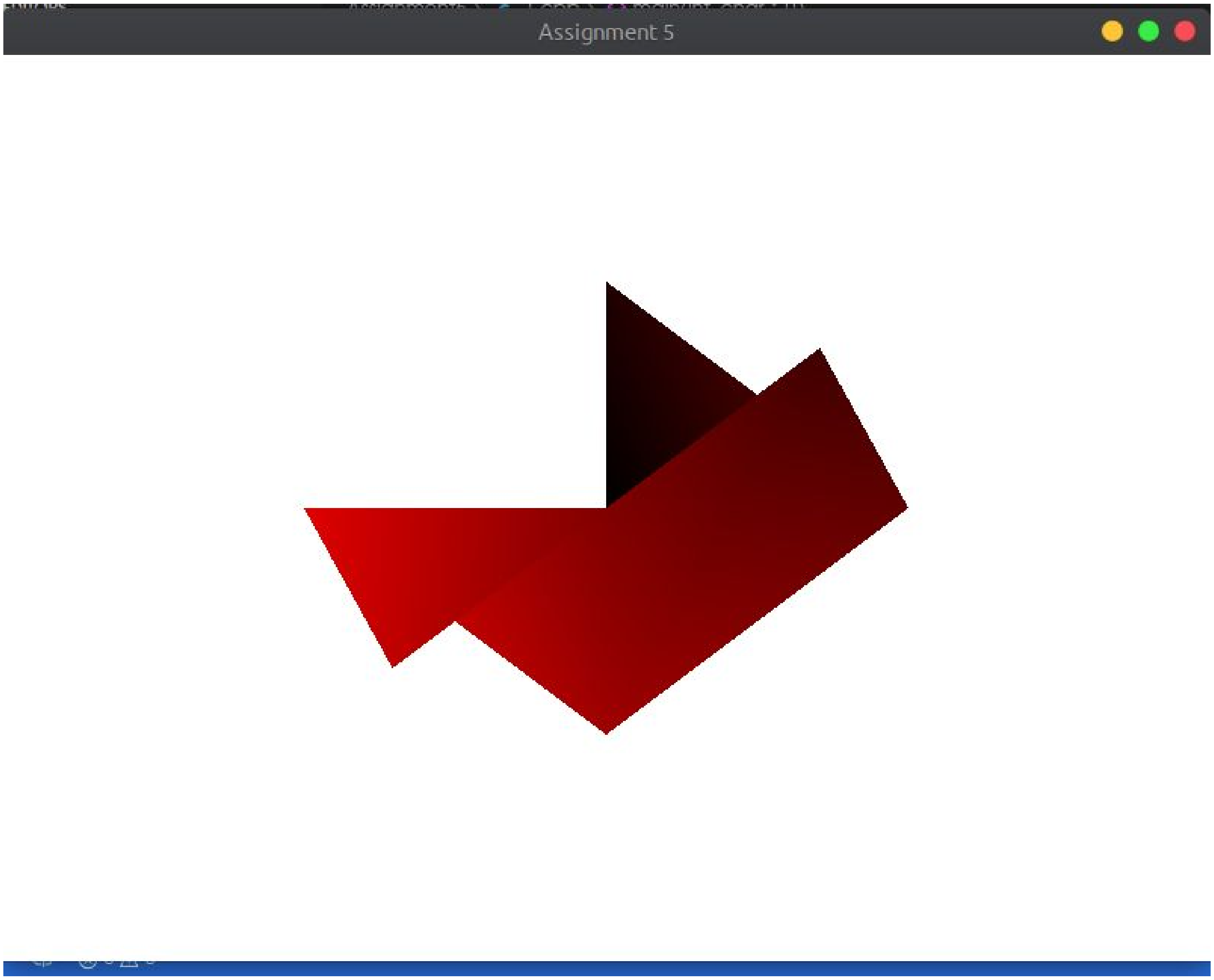
```
#include <GL/glut.h>
#include <stdio.h>
#include <bits/stdc++.h>

using namespace std;

// To Compile:-
// g++ -pthread 1.cpp -lglfw3 -lGLEW -lGLU -lGL -lXrandr -lXxf86vm -lXi -lXinerama
-lX11 -lrt -ldl -lglut

int size = 9;
float points[9][2] =
{{0.25f,0.433f},{-0.25f,0.433f},{-0.5f,0.0f},{-0.25f,-0.433f},{0.25f,-0.433f},{0.5f,0
.0f},{-0.25f,0.433f},{-0.5f,0.0f},{-0.25f,-0.433f},};

void showTriangles()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0, 0, 1);

   glBegin(GL_TRIANGLES);
   for(int i = 0;i<size;i++){
       glVertex2f(points[i][0],points[i][1]);
   }
   glEnd();
   glutSwapBuffers();
}

void showTriangleStripes()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0, 0, 1);
   glBegin(GL_TRIANGLE_STRIP_ADJACENCY);
   for(int i = 0;i<size;i++){
       if(i>=3) glColor3f(1,0,0);
       if(i>=6) glColor3f(0,1,0);
       glVertex2f(points[i][0],points[i][1]);
   }
   glEnd();
   glutSwapBuffers();
}

void showTriangleFans(){
   glClear(GL_COLOR_BUFFER_BIT);
   vector<vector<float>> points = {{0,0},{0,0.5},{0.5/1.414,0.5/1.414},{0.5,0}};
   glBegin(GL_TRIANGLE_FAN);
   for(int i = 0;i<points.size();i++){
       glColor3f(0.25*i,0,0);
       glVertex2f(points[i][0],points[i][1]);
   }
   glEnd();
   glutSwapBuffers();

}

void showQuad(){

   glClear(GL_COLOR_BUFFER_BIT);
   vector<vector<float>> points =
{{0,0},{0,0.5},{0.5/1.414,0.5/1.414},{0.5,0},{0,0},{0,-0.5},{-0.5/1.414,-0.5/1.414},{
-0.5,0}};
   glBegin(GL_QUADS);
   for(int i = 0;i<points.size();i++){
       glColor3f(0.125*i,0,0);
       glVertex2f(points[i][0],points[i][1]);
   }
   glEnd();
   glutSwapBuffers();
}

void showQuadStripe(){
   glClear(GL_COLOR_BUFFER_BIT);
   vector<vector<float>> points =
{{0,0},{0,0.5},{0.5/1.414,0.5/1.414},{0.5,0},{0,0},{0,-0.5},{-0.5/1.414,-0.5/1.414},{
-0.5,0}};
   glBegin(GL_QUAD_STRIP);
   for(int i = 0;i<points.size();i++){
       glColor3f(0.125*i,0,0);
       glVertex2f(points[i][0],points[i][1]);
   }
   glEnd();
   glutSwapBuffers();

}

void showPolygon(){
   glClear(GL_COLOR_BUFFER_BIT);
   vector<vector<float>> points =
{{0,0},{0,0.5},{0.5/1.414,0.5/1.414},{0.5,0},{0,-0.5},{-0.5/1.414,-0.5/1.414},{-0.5,0
}};
   glBegin(GL_POLYGON);
   for(int i = 0;i<points.size();i++){
       glColor3f(0.125*i,0,0);
       glVertex2f(points[i][0],points[i][1]);
   }
   glEnd();
   glutSwapBuffers();

}


int main(int argc, char *argv[])
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);

   glutInitWindowSize(800, 600);
   glutCreateWindow("Assignment 5");

   glClearColor(1, 1, 1, 0);
   glShadeModel(GLU_FLAT);
   glPointSize(6.0);

   glutDisplayFunc(showPolygon);
   glutMainLoop();
}
```

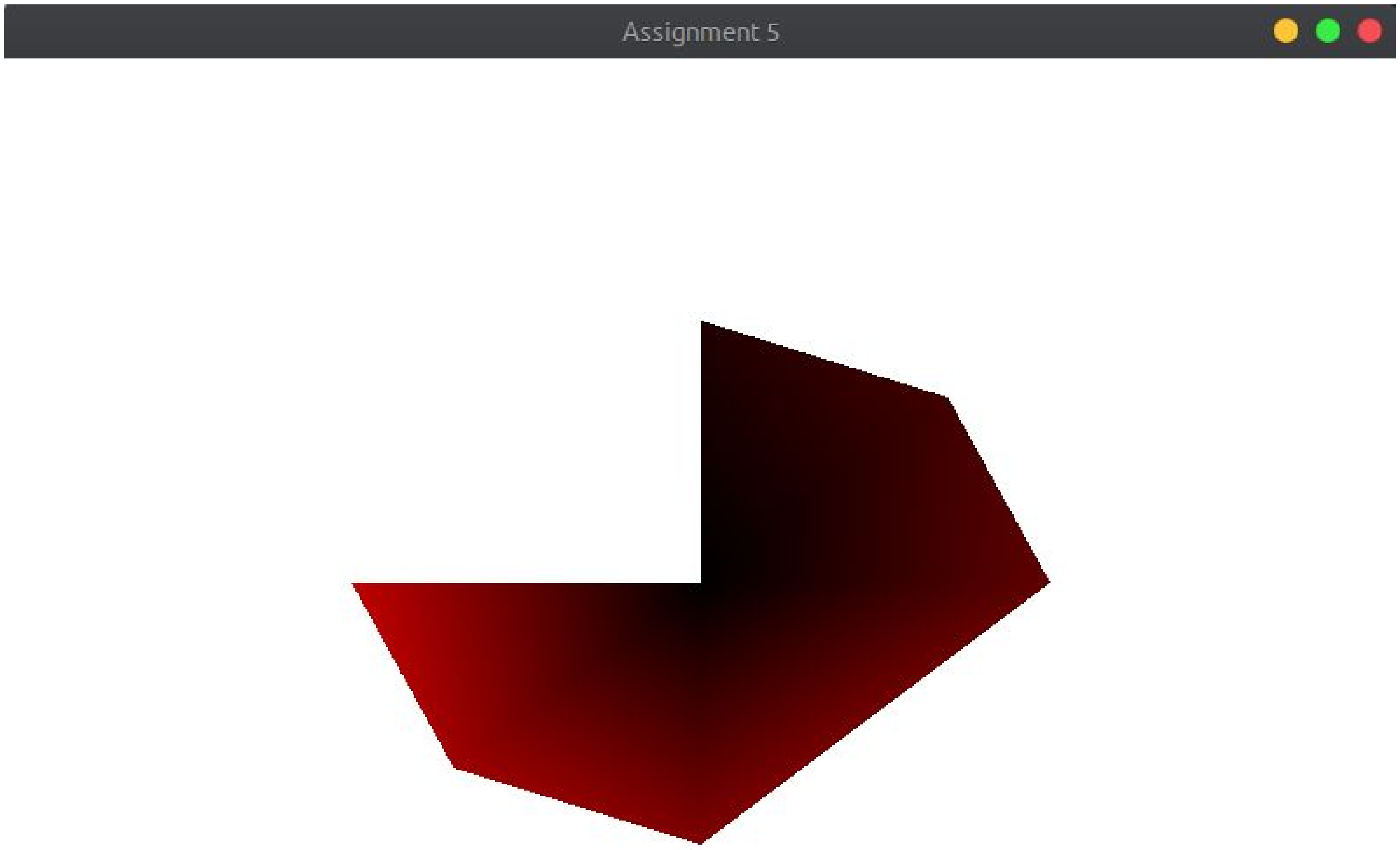

Assignment 5


Assignment 5


Assignment 5


Assignment 5

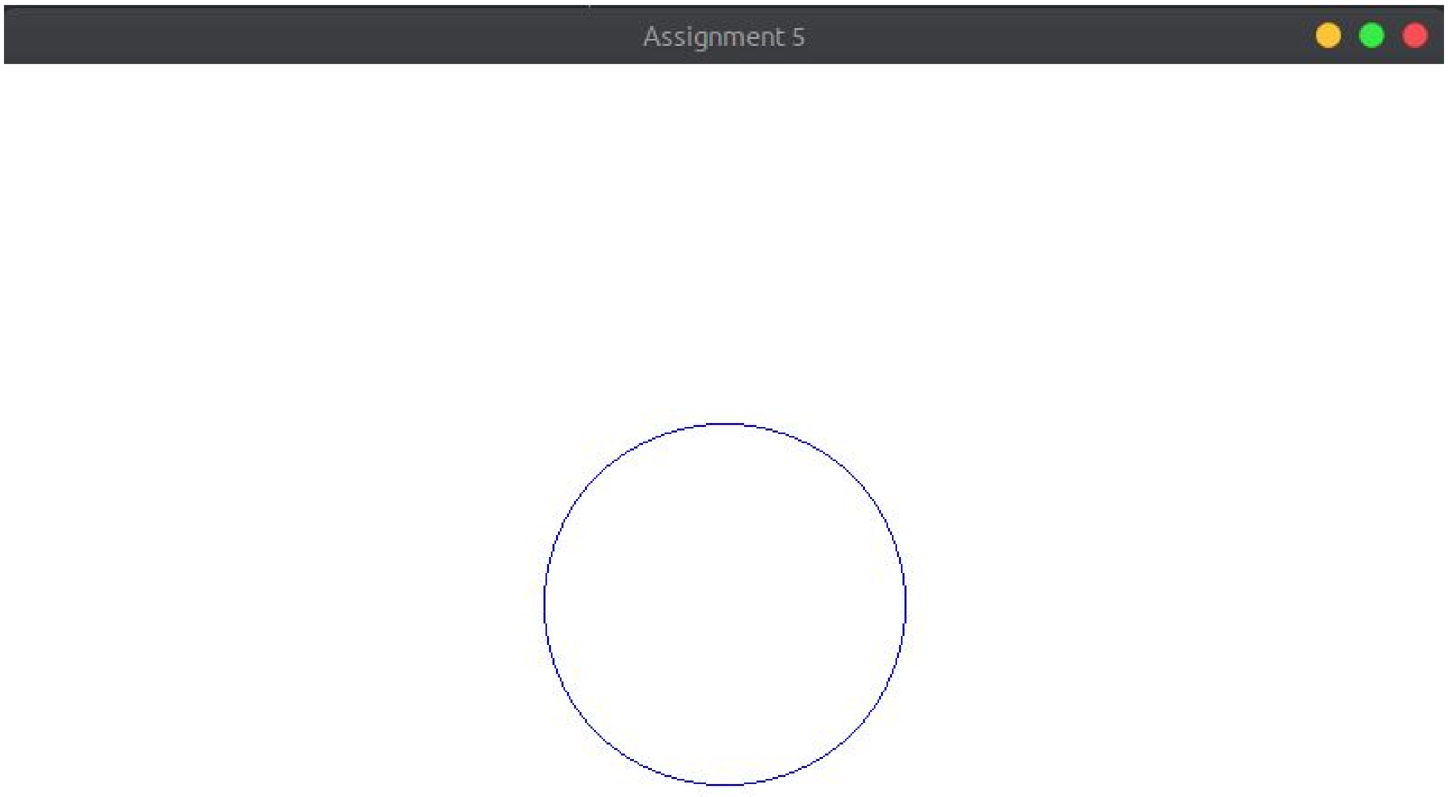

Assignment 5

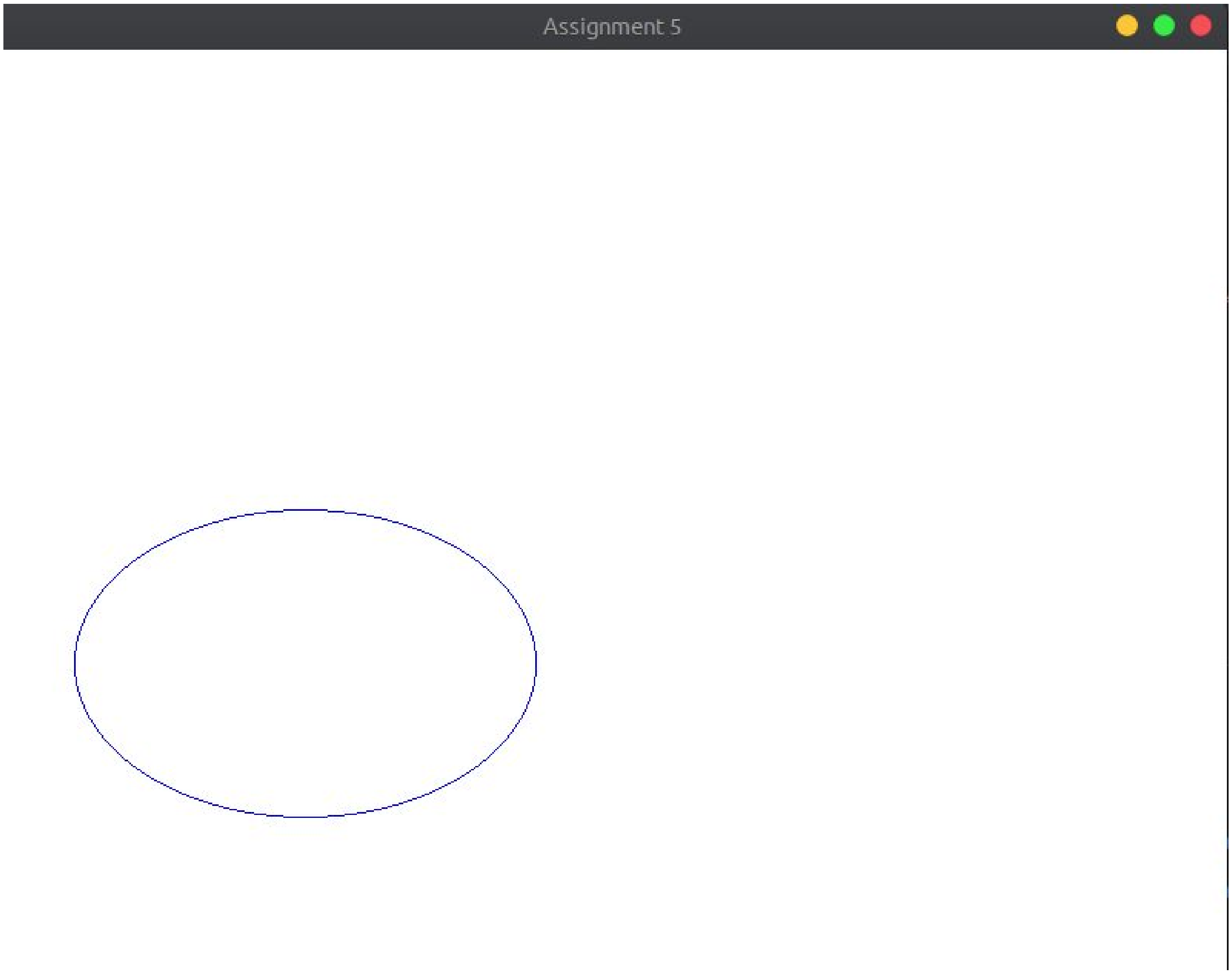

Assignment 5

2. Write a menu driven program for following algorithms:
a) Mid Point Circle generating algorithm
b) Mid Point Ellipse generating algorithm

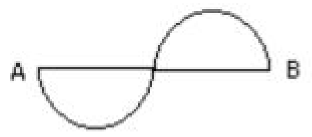
```
#include <bits/stdc++.h>
#include <GL/glut.h>

using namespace std;

// To Compile:-
// g++ -pthread 2.cpp -lglfw3 -lGLEW -lGLU -lGL -lXrandr -lXxf86vm -lXi -lXinerama
-lX11 -lrt -ldl -lglut

void plot(int x, int y)
{
   glBegin(GL_POINTS);
   glVertex2i(x,y);
   glEnd();
}
void midpoint_circle_util(int r,int cx,int cy){

   int y = 0, x = r;
   int p = 1 - r;

   while(x>=y){
       plot(cx+x,cy+y);
       plot(cx-x,cy+y);
       plot(cx+x,cy-y);
       plot(cx-x,cy-y);
       plot(cx+y,cy+x);
       plot(cx-y,cy+x);
       plot(cx+y,cy-x);
       plot(cx-y,cy-x);
       if(p<=0){
           p += 2*(y+1)+1;
       }else{
           p += 2*(y+1)+1-2*(x+1);
           x--;
       }
       y++;
   }
}
void midpoint_circle(){
   glClear(GL_COLOR_BUFFER_BIT);
   glPointSize(1.0);
   glColor3f(0,0,1);
   midpoint_circle_util(100,400,300);
   glutSwapBuffers();
}

void midpoint_ellipse_util(int rx,int ry,int cx,int cy){
   float dx, dy, p1, p2, x, y;
   x = 0;
   y = ry;
    p1 = (ry * ry) - (rx * rx * ry) +
                    (0.25 * rx * rx);
   dx = 2 * ry * ry * x;
   dy = 2 * rx * rx * y;
    while (dx < dy)
   {
       plot(cx+x,cy+y);
       plot(cx-x,cy+y);
       plot(cx+x,cy-y);
       plot(cx-x,cy-y);
       if (p1 < 0)
       {
           x++;
           dx = dx + (2 * ry * ry);
           p1 = p1 + dx + (ry * ry);
       }
       else
       {
           x++;
           y--;
           dx = dx + (2 * ry * ry);
           dy = dy - (2 * rx * rx);
           p1 = p1 + dx - dy + (ry * ry);
       }
   }
    p2 = ((ry * ry) * ((x + 0.5) * (x + 0.5))) +
         ((rx * rx) * ((y - 1) * (y - 1))) -
          (rx * rx * ry * ry);
    while (y >= 0)
   {
       plot(cx+x,cy+y);
       plot(cx-x,cy+y);
       plot(cx+x,cy-y);
       plot(cx-x,cy-y);
       if (p2 > 0)
       {
           y--;
           dy = dy - (2 * rx * rx);
           p2 = p2 + (rx * rx) - dy;
       }
       else
       {
           y--;
           x++;
           dx = dx + (2 * ry * ry);
           dy = dy - (2 * rx * rx);
           p2 = p2 + dx - dy + (rx * rx);
       }
   }
}
void midpoint_ellipse(){
   glClear(GL_COLOR_BUFFER_BIT);
   glPointSize(1.0);
   glColor3f(0,0,1);
   midpoint_ellipse_util(150,100,200,200);
   glutSwapBuffers();
}

int main(int argc, char *argv[]){
   cout<<"Midpoint Algorithms\n1. Circle\n2. Ellpise\n>>";
   int choice;
   cin >> choice;

   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(800, 600);
   glutCreateWindow("Assignment 5");
   glShadeModel(GLU_FLAT);
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 800.0, 0.0, 600.0);

   if(choice==1){
       glutDisplayFunc(midpoint_circle);
   }else if(choice==2){
       glutDisplayFunc(midpoint_ellipse);
   }else{
       cout<<"Invalid Choice!"<<endl;
       return 0;
   }
   glutMainLoop();
```

```
}
```


Assignment 5


Assignment 5

3.Write a program to generate the following figure :-

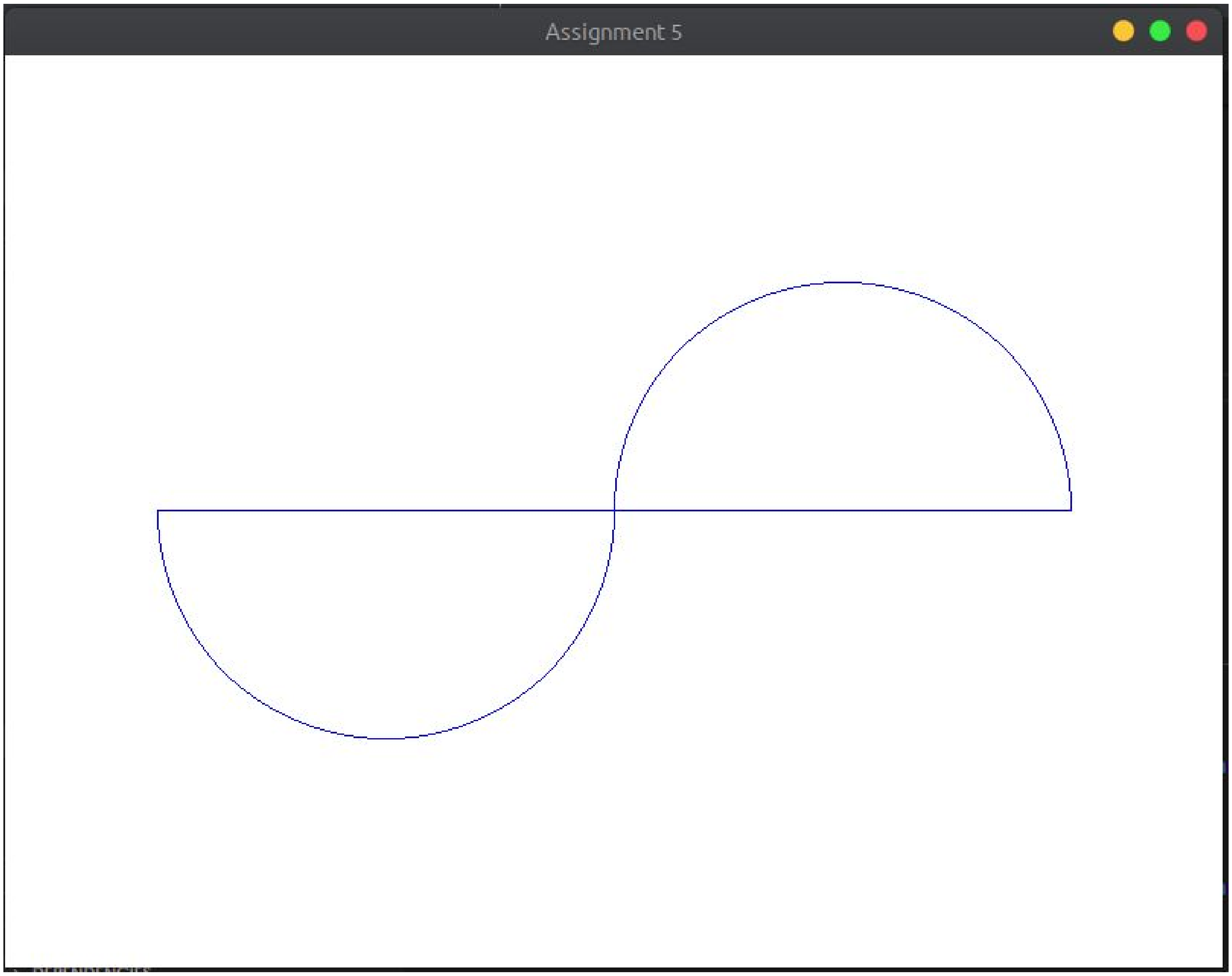


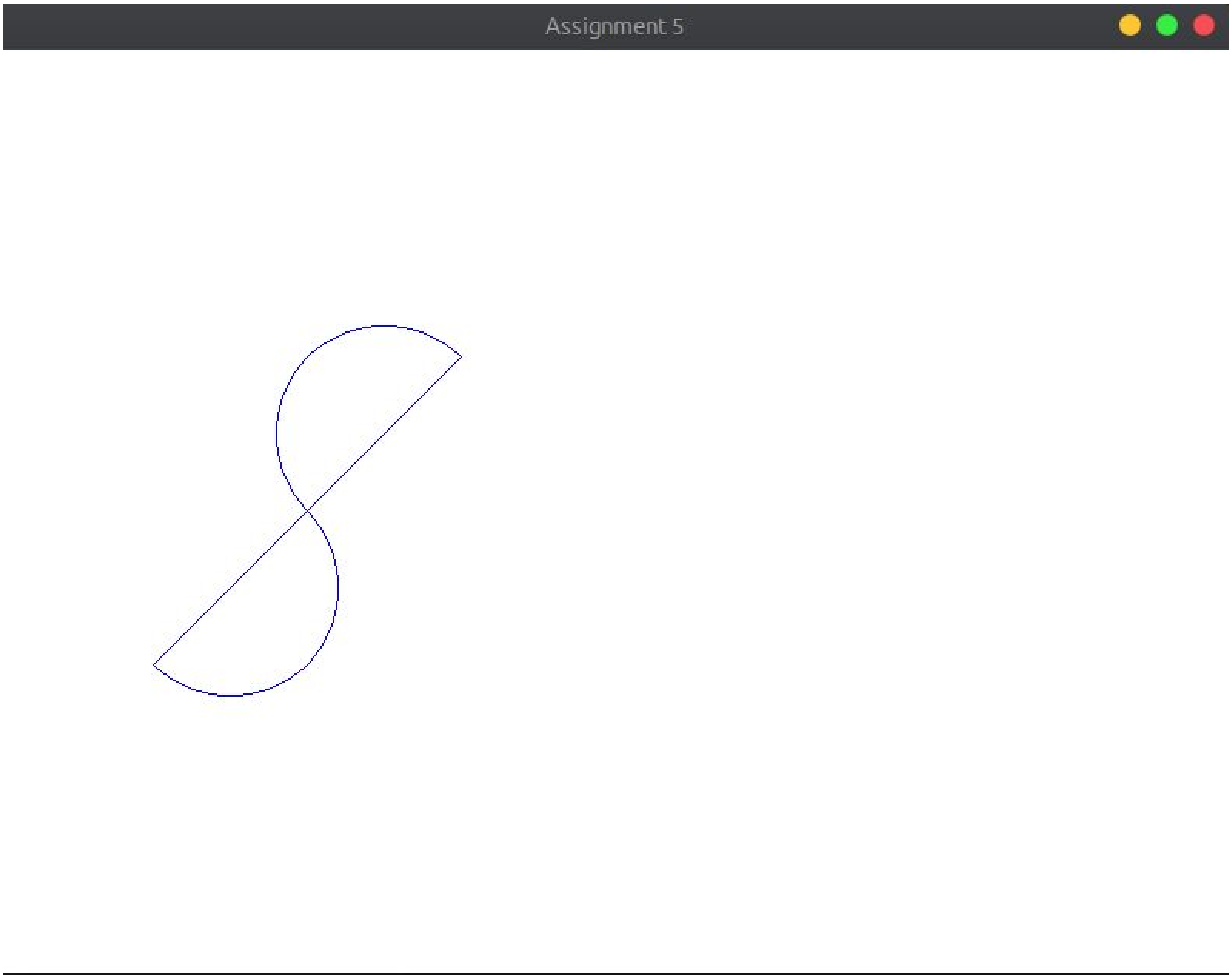
Point A and B is input.
(Use the concept of Mid Point Circle generating algorithm and DDA Line Drawing Algorithm)

```
#include <bits/stdc++.h>
#include <GL/glut.h>

using namespace std;

// To Compile:-
// g++ -pthread 3.cpp -lglfw3 -lGLEW -lGLU -lGL -lXrandr -lXxf86vm -lXi -lXinerama
-lX11 -lrt -ldl -lglut

int a,b,y;
void plot(int x, int y)
{
   glBegin(GL_POINTS);
   glVertex2i(x,y);
   glEnd();
}


void draw_line_bres(int x1, int y1, int x2, int y2,int t = 0)
{
   if(abs(y2-y1)>abs(x2-x1)){
       draw_line_bres(y1,x1,y2,x2,1);
       return;
   }
   if(x1>x2){
       swap(x1,x2);
       swap(y1,y2);
   }
   int inc = y1>y2?-1:1;
   int m_new = 2 * (y2 - y1);
   int slope_error_new = m_new + inc==1?-(x2 - x1):(x2-x1);
   for (int x = x1, y = y1; x <= x2; x++)
   {
       if(t) plot(y,x);
       else plot(x, y);

       slope_error_new += m_new;
```

```
        if (inc==1 && slope_error_new >= 0)
        {
            y+=inc;
            slope_error_new  -= 2 * (x2 - x1);
        }else if(inc==-1 && slope_error_new<=0){
            y+=inc;
            slope_error_new += 2*(x2-x1);
        }

    }
}

void draw_half_circle(int r,int cx,int cy, int up){

    int y = 0, x = r;
    int p = 1 - r;

    while(x>=y){
        if(up){
            plot(cx+x,cy+y);
            plot(cx-x,cy+y);
            plot(cx+y,cy+x);
            plot(cx-y,cy+x);
        }else{
            plot(cx+x,cy-y);
            plot(cx-x,cy-y);
            plot(cx+y,cy-x);
            plot(cx-y,cy-x);
        }
        if(p<=0){
            p += 2*(y+1)+1;
        }else{
            p += 2*(y+1)+1-2*(x+1);
            x--;
        }
        y++;
    }
}

void draw_custom_util(){
    int mid = (a + b)/2;

    int cx1 = (a+mid)/2;
    int cx2 = (mid+b)/2;
    int cy1 = y, cy2 = y;
```

```
   int r = (mid-cx1);

   draw_line_bres(a,y,b,y);
   draw_half_circle(r,cx1,cy1,0);
   draw_half_circle(r,cx2,cy2,1);

}
void draw_custom(){
   glClear(GL_COLOR_BUFFER_BIT);
   glPointSize(1.0);
   glColor3f(0,0,1);
   draw_custom_util();
   glutSwapBuffers();
}


int main(int argc, char *argv[]){
   cout<<"Custom Diagram Drawing Program"<<endl;
   cout<<"A: ";
   cin >> a;
   cout<<"B: ";
   cin >> b;
   cout<<"Y: ";
   cin >> y;

   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(800, 600);
   glutCreateWindow("Assignment 5");
   glShadeModel(GLU_FLAT);
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0.0, 800.0, 0.0, 600.0);

   glutDisplayFunc(draw_custom);
   glutMainLoop();
}
```


Assignment 5


Assignment 5