ASSIGNMENT-4

1)
```
#include <GLUT/glut.h>
#include <math.h>

void init()
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-100, 100, -100, 100, -1, 1);
}

void drawCircle(int xc, int yc, int r)
{
   int x = 0, y = r;
   int d = 1 - r;

   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);
   while (y > x)
   {
      if (d < 0)
      {
         d = d + 2 * x + 3;
         x = x + 1;
      }
      else
      {
         d = d + 2 * (x - y) + 5;
         x = x + 1;
         y = y - 1;
      }
      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 display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    drawCircle(0, 0, 50);
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Midpoint Circle Algorithm");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}


2)#include <iostream>
#include <GLUT/glut.h>

using namespace std;

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glPointSize(2.0);

    int xCenter = 320, yCenter = 240;    int a = 200, b = 100; // major and minor axis
lengths
    int x = 0, y = b;
    int aSquared = a * a;
    int bSquared = b * b;
    int twoASquared = 2 * aSquared;
    int twoBSquared = 2 * bSquared;
    int d = bSquared - aSquared*b + aSquared/4;
    glBegin(GL_POINTS);
    while(aSquared*(y - 0.5) > bSquared*(x + 1))
    {
        glVertex2i(xCenter + x, yCenter + y);        glVertex2i(xCenter - x, yCenter + y);
        glVertex2i(xCenter - x, yCenter - y);
        glVertex2i(xCenter + x, yCenter - y);
        if (d < 0)
        {
            x++; // choose E
            d += twoBSquared*x + bSquared;
        }
        else
        {
            x++; y--; // choose SE
            d += twoBSquared*x - twoASquared*y + aSquared + bSquared;
        }
    }

    d = bSquared*(x + 0.5)*(x + 0.5) + aSquared*(y - 1)*(y - 1) - aSquared*bSquared; initial
decision parameter
    while (y >= 0) // loop through second half of the ellipse
    {
        glVertex2i(xCenter + x, yCenter + y);       glVertex2i(xCenter - x, yCenter + y);
        glVertex2i(xCenter - x, yCenter - y);
        glVertex2i(xCenter + x, yCenter - y);
        if (d > 0)
        {
            y--; // choose N
            d += twoASquared*y + aSquared;
        }
        else
        {
            x++; y--; // choose NE
            d += twoBSquared*x - twoASquared*y + aSquared + bSquared;
        }
    }
    glEnd();

    glFlush();
}

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow("Midpoint Ellipse Drawing");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```