**LAB-6**

// OPEN GL app for drawing random 50 points

#include <GL/glut.h>

#include <GL/gl.h>

#include <cstdlib>

#include <ctime>

int total = 50;

int WIN\_WIDTH = 1024;

int WIN\_HEIGHT = 768;

void display(){

srand(time(0));

glClearColor(0, 0, 0, 0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1, 1, 1);

glPointSize(5.0f);

glBegin(GL\_POINTS);

for (int i = 0; i < total; i++) {

int pointX = rand() % WIN\_WIDTH;

int pointY = rand() % WIN\_HEIGHT;

glVertex2f(pointX, pointY);

}

glEnd();

glFlush();

}

int main(int argc, char \*\*argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutCreateWindow("LAB 6 Niraj Maharjan");

glLoadIdentity();

gluOrtho2D(0, WIN\_WIDTH, 0, WIN\_HEIGHT);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT**

**LAB-7**

//OPEN GL app to for implementation of direct method to draw line

#include <GL/glut.h>

#include <iostream>

using namespace std;

int x1, y1, x2, y2;

void drawLine() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0, 0, 0);

float dx = x2 - x1;

float dy = y2 - y1;

float m = dy / dx;

float b = y1 - m \* x1;

glBegin(GL\_POINTS);

if (dx != 0) {

for (int x = x1; x <= x2; x++) {

int y = m \* x + b;

glVertex2i(x, y);

}

}

else {

for (int y = y1; y <= y2; y++) {

glVertex2i(x1, y);

}

}

glEnd();

glFlush();

}

void myInit() {

glutInitWindowSize(1000,600);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("LAB 7 Niraj Maharjan");

glClearColor(1, 1, 1, 1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, 1000,0,600);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv) {

cout << "Enter the coordinates:" << endl;

cout << "x1 = "; cin >> x1;

cout << "y1 = "; cin >> y1;

cout << "x2 = "; cin >> x2;

cout << "y2 = "; cin >> y2;

glutInit(&argc, argv);

myInit();

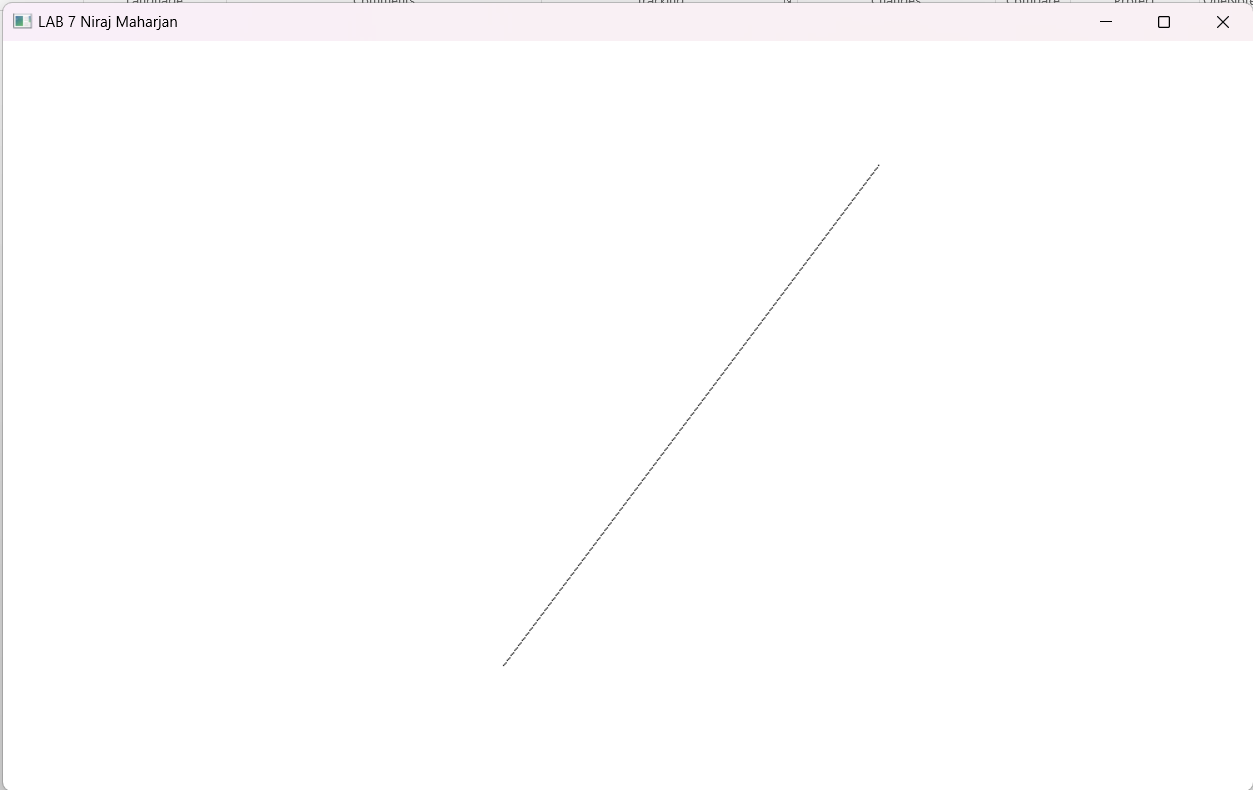
glutDisplayFunc(drawLine);

glutMainLoop();

return 0;

}

**OUTPUT:**



# LAB-8

//OPEN GL app to for implementation of DDA algorithm to draw line

#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

float xi, yi, xf, yf;

int WIN\_WIDTH = 800;

int WIN\_HEIGHT = 600;

void drawLineDDA() {

float dx = xf - xi;

float dy = yf - yi;

float step = (abs(dx) > abs(dy)) ? abs(dx) : abs(dy);

float x\_inc = dx/step, y\_inc = dy/step;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0, 0, 0);

glPointSize(2);

glBegin(GL\_POINTS);

glVertex2f(round(xi),round(yi));

for (int i = 0; i <= step; i++){

xi += x\_inc;

yi += y\_inc;

glVertex2f(round(xi), round(yi));

}

glEnd();

glFlush();

}

void myInit() {

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutCreateWindow("LAB 8 Niraj Maharjan");

glClearColor(1, 1, 1, 0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, WIN\_WIDTH, 0, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv) {

cout << "Enter the coordinates:" << endl;

cout << "x1 = "; cin >> xi;

cout << "y1 = "; cin >> yi;

cout << "x2 = "; cin >> xf;

cout << "y2 = "; cin >> yf;

glutInit(&argc, argv);

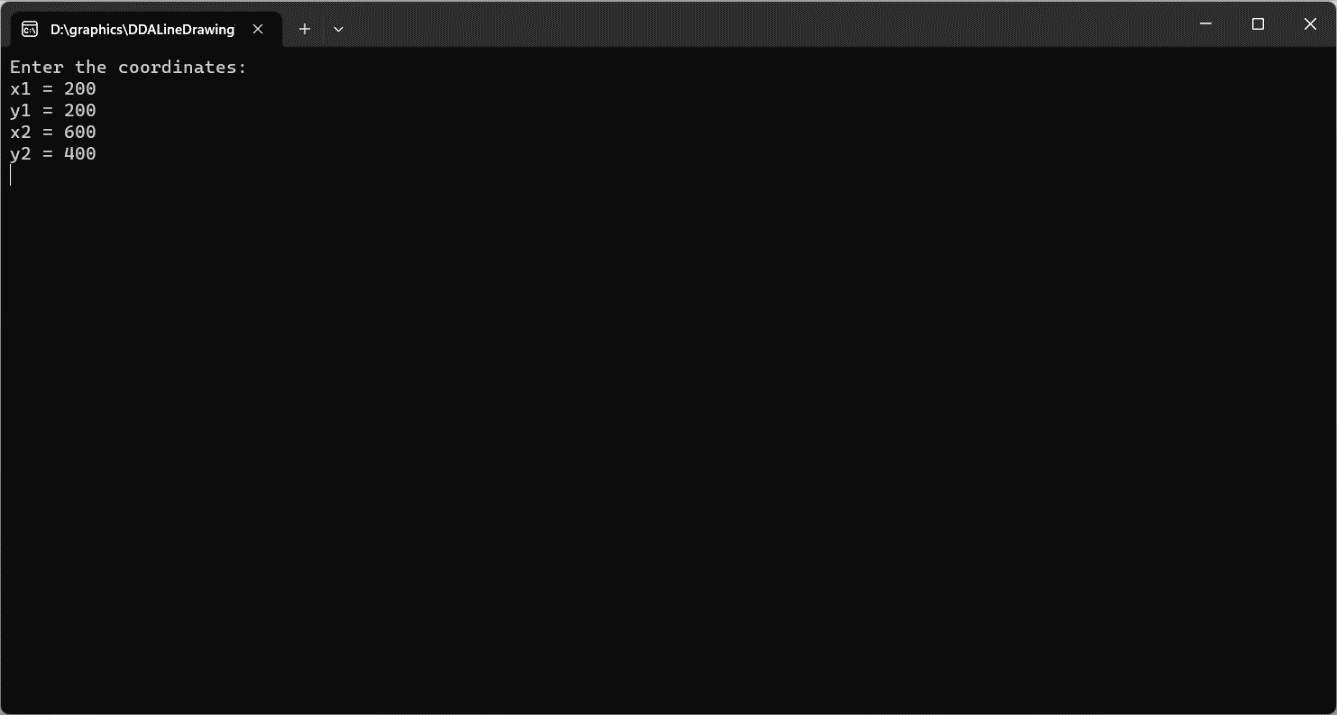
myInit();

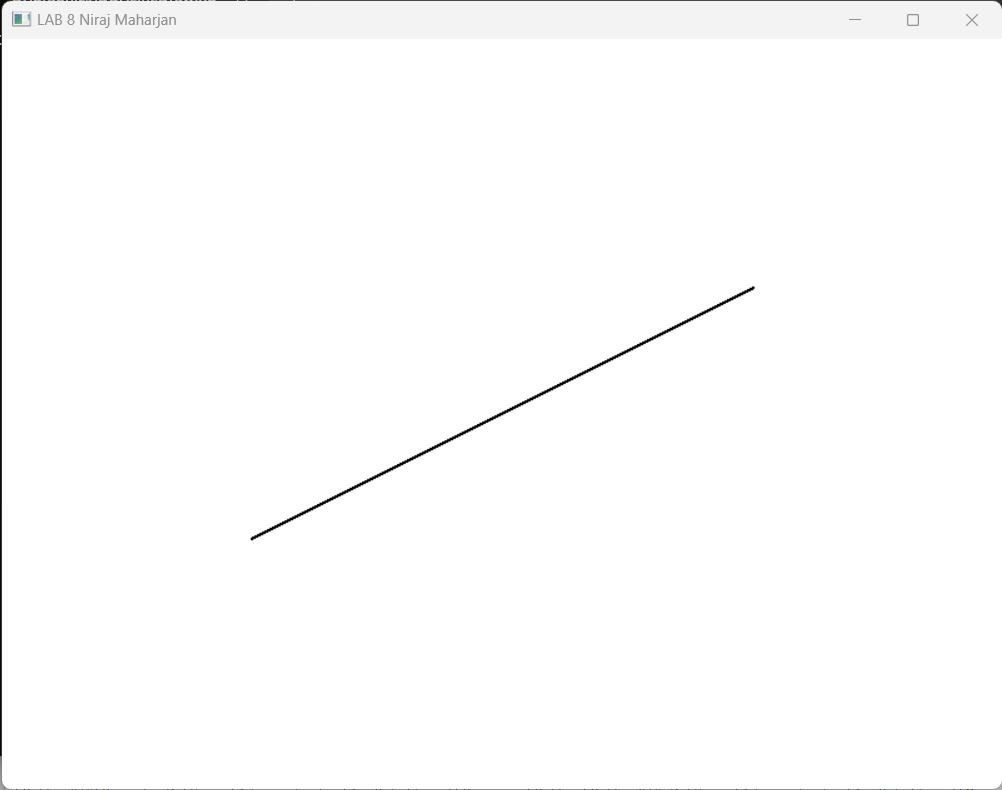
glutDisplayFunc(drawLineDDA);

glutMainLoop();

return 0;

}

**OUTPUT**

****

**LAB-10**

//OPEN GL app to for implementation of mid-point algorithm to draw circle

#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

int WIN\_WIDTH = 800;

int WIN\_HEIGHT = 600;

int xi,yi,r;

void drawPoints(int x, int y, int xi, int yi) {

glVertex2i(xi + x, yi + y);

glVertex2i(xi - x, yi + y);

glVertex2i(xi + x, yi - y);

glVertex2i(xi - x, yi - y);

glVertex2i(xi + y, yi + x);

glVertex2i(xi - y, yi + x);

glVertex2i(xi + y, yi - x);

glVertex2i(xi - y, yi - x);

}

// Midpoint Circle Algorithm

void drawCircle() {

int x = 0;

int y = r;

int d = 1 - r;

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POINTS);

glColor3d(0, 0, 0);

while (x <= y) {

glPointSize(20.0f);

drawPoints(x, y, xi, yi);

if (d < 0) {

d += 2 \* x + 1;

} else {

d += 2 \* (x - y) + 1;

y--;

}

x++;

}

glEnd();

glFlush();

}

void myInit() {

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("LAB 10 Niraj Maharjan");

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0, 0, 0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-WIN\_WIDTH, WIN\_WIDTH, -WIN\_HEIGHT, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv) {

cout << "Enter the coordinates:" << endl;

cout << "xi = "; cin >> xi;

cout << "yi = "; cin >> yi;

cout << "Radius = "; cin >> r;

glutInit(&argc, argv);

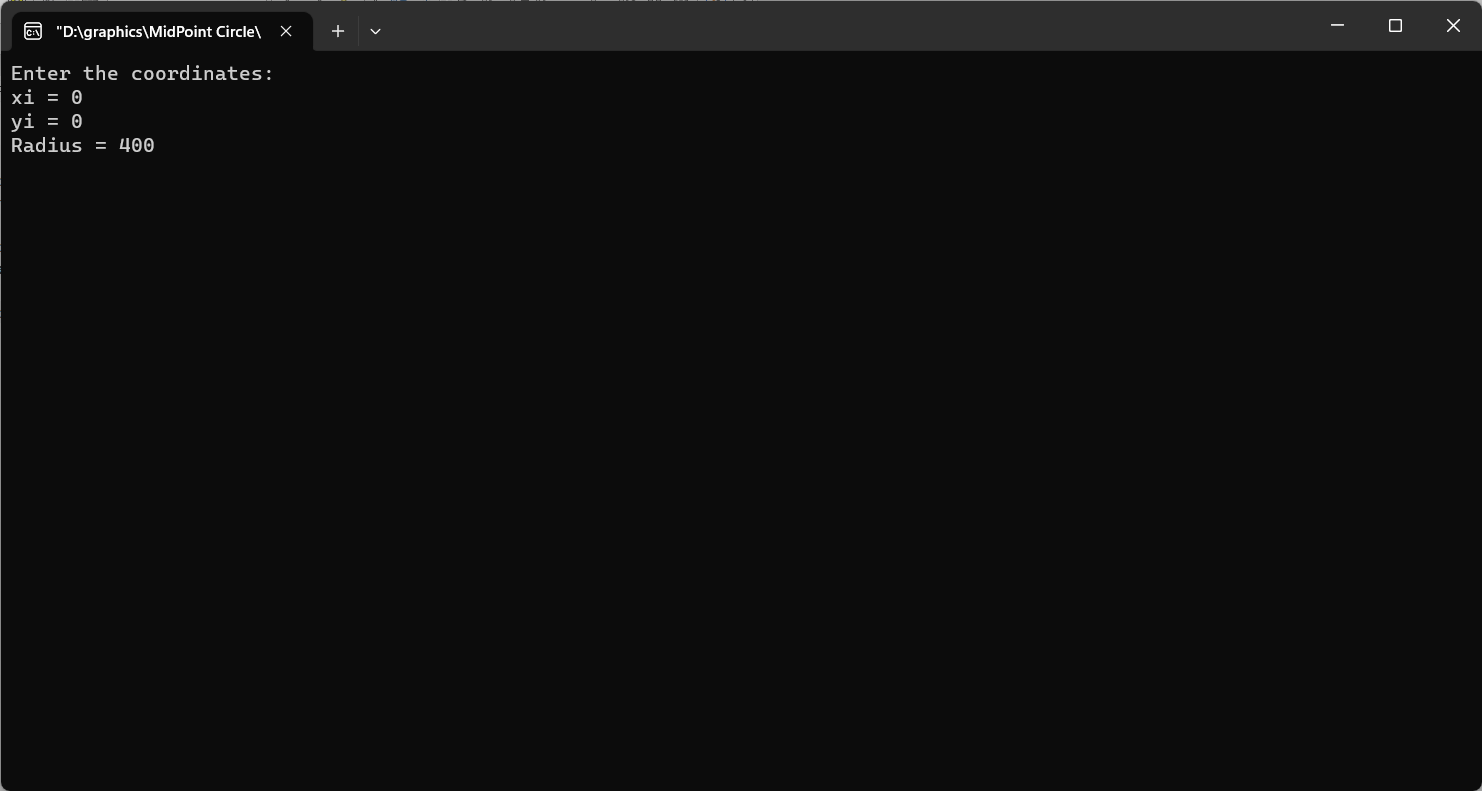
myInit();

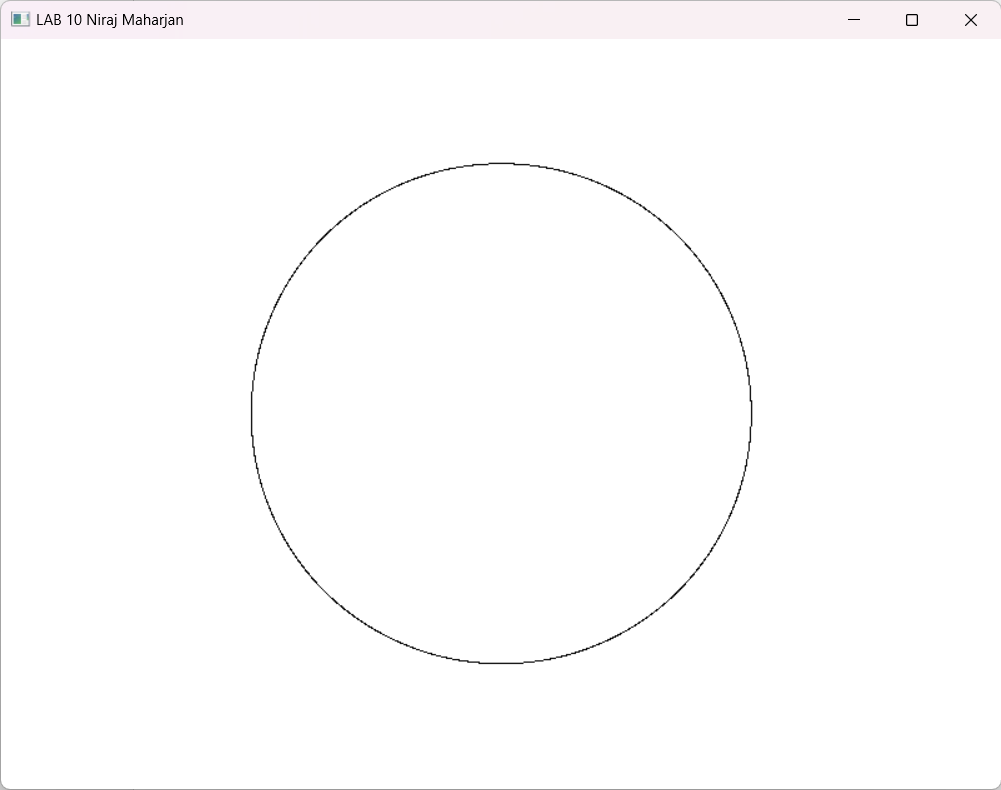
glutDisplayFunc(drawCircle);

glutMainLoop();

return 0;

}

**OUTPUT**

****

**LAB-11**

//OPEN GL app to for implementation of mid-point algorithm to draw ellipse

#include <GL/glut.h>

#include <iostream>

#include <cmath>

using namespace std;

int xi, yi, a, b;

int WIN\_WIDTH = 800;

int WIN\_HEIGHT = 600;

void drawPoints(int x, int y) {

glBegin(GL\_POINTS);

glColor3d(0, 0, 0);

glVertex2i(xi + x, yi + y);

glVertex2i(xi - x, yi + y);

glVertex2i(xi + x, yi - y);

glVertex2i(xi - x, yi - y);

glEnd();

}

// Midpoint Ellipse Algorithm

void drawEllipse() {

int x, y;

float d1, d2;

x = 0;

y = b;

d1 = b \* b - a \* a \* b + 0.25 \* a \* a;

drawPoints(x, y);

// Region 1

while (a \* a \* (y - 0.5) > b \* b \* (x + 1)) {

if (d1 < 0) {

d1 += b \* b \* (2 \* x + 3);

} else {

d1 += b \* b \* (2 \* x + 3) + a \* a \* (-2 \* y + 2);

y--;

}

x++;

drawPoints(x, y);

}

// Region 2

d2 = b \* b \* (x + 0.5) \* (x + 0.5) + a \* a \* (y - 1) \* (y - 1) - a \* a \* b \* b;

while (y > 0) {

if (d2 < 0) {

d2 += b \* b \* (2 \* x + 2) + a \* a \* (-2 \* y + 3);

x++;

} else {

d2 += a \* a \* (-2 \* y + 3);

}

y--;

drawPoints(x, y);

}

glFlush();

}

void myInit() {

glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Drawing an ellipse using the Midpoint Ellipse Algorithm");

glClearColor(1, 1, 1, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0, 0, 0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-WIN\_WIDTH, WIN\_WIDTH, -WIN\_HEIGHT, WIN\_HEIGHT);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv) {

cout << "Enter the coordinates for the center of the ellipse:" << endl;

cout << "xi = "; cin >> xi;

cout << "yi = "; cin >> yi;

cout << "Enter the values for the semi-major and semi-minor axes (a and b):" << endl;

cout << "a = "; cin >> a;

cout << "b = "; cin >> b;

glutInit(&argc, argv);

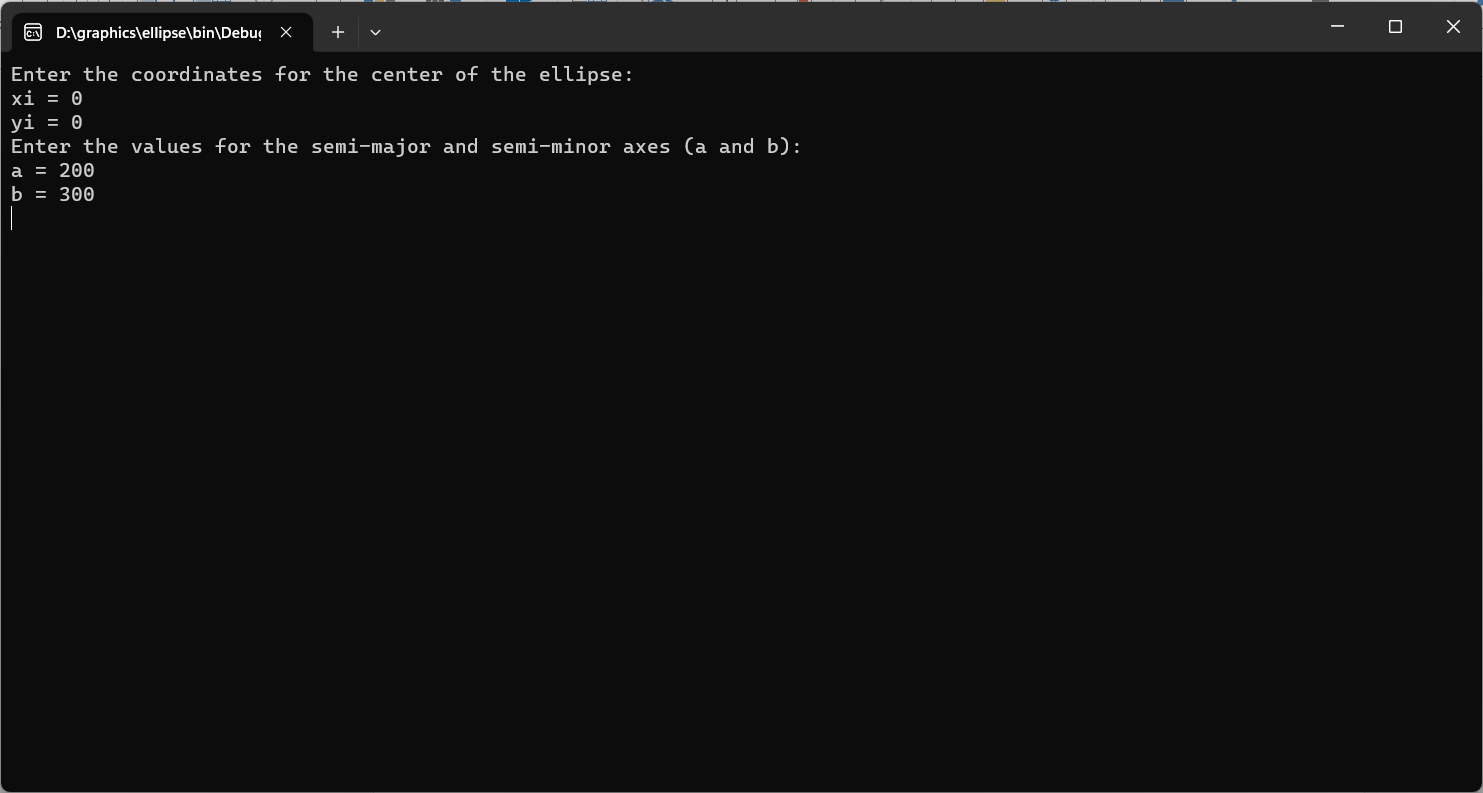
myInit();

glutDisplayFunc(drawEllipse);

glutMainLoop();

return 0;

}

**OUTPUT**

