Name	:	**V Vikram**	**Class**	**:** CSE ‘C’
Reg. No.	:	**18 5001 194**	**Date**	**:** 14/08/2021
Subject	:	**UCS1712---Graphics and Multimedia Lab**		

QUESTION :

Lab Exercise 4 :

Midpoint Circle Drawing Algorithm in C++ using OpenGL

a) To plot points that make up the circle with center (xc,yc) and radius r using the Midpoint circle drawing algorithm. Give atleast 2 test cases.

Case 1: With center (0,0)

Case 2: With center (xc,yc)

b) To draw any object using line and circle drawing algorithms.

CODE :-

Midpoint.cpp :

```
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>
using namespace std;

int pntX1, pntY1, r;

void plot(int x, int y)
{
      glBegin(GL_POINTS);
      glVertex2i(x + pntX1, y + pntY1);
      glEnd();
}


void myInit(void)
{
      glClearColor(0.0, 0.0, 0.0, 0.0);
      glColor3f(255.0f / 255.0f, 255.0f / 255.0f, 255.0f / 255.0f);
      glPointSize(4.0);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(-100, 100, -100, 100);
}


void midPointCircleAlgo()
{
      int x = 0;
      int y = r;
      float decision = 5 / 4 - r;
      plot(x, y);

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

}

void myDisplay(void)
{
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1.0, 1.0, 0.0);
      glPointSize(1.0);

      midPointCircleAlgo();

      glFlush();
}

void main(int argc, char** argv)
{
      cout << "\n   Lab Exercise 4 : \n\tMidpoint Circle Drawing
Algorithm \n\tin C++ using OpenGL";
      cout << "\n\nEnter X-coordinate   : "; cin >> pntX1;
      cout << "\nEnter Y-coordinate   : "; cin >> pntY1;
      cout << "\nEnter radius         : "; cin >> r;

      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize(500, 500);
      glutInitWindowPosition(300, 10);
      glutCreateWindow("Mid Pt Circle Drawing Algo");
      glutDisplayFunc(myDisplay);
      myInit();
      glutMainLoop();
}
```

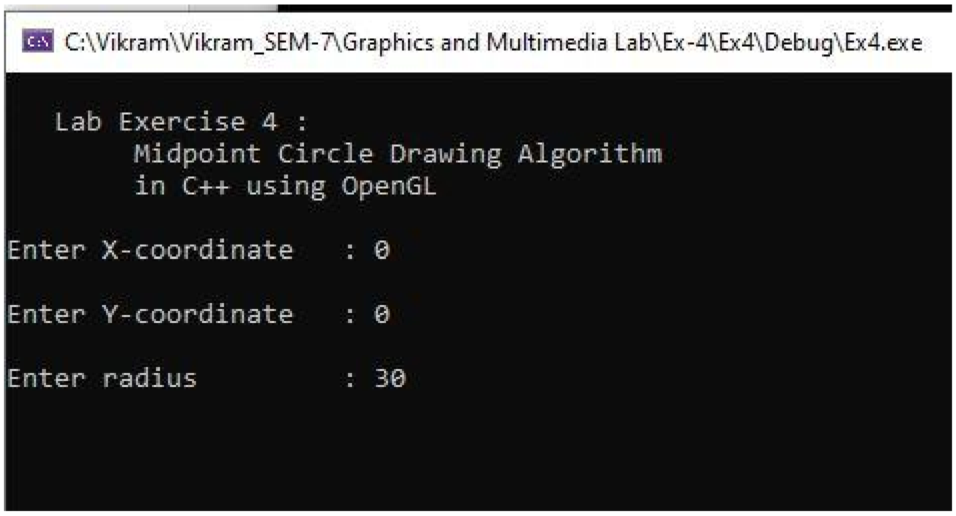
Diagram.cpp :

```
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>
using namespace std;
```

```
void draw_Traffic_Light() {

      GLfloat x_i, y_i, theta = 0;
      GLfloat x_c, y_c, r;
      int i;
      glColor3f(0, 0, 0);
      glBegin(GL_QUADS);
      //signal box
      glVertex2d(40,30);
      glVertex2d(60,30);
      glVertex2d(60,80);
      glVertex2d(40,80);
      glEnd();

      glColor3f(0.55, 0.55, 0);
      //signal pole
      glBegin(GL_QUADS);
      glVertex2d(45,0);
      glVertex2d(55,0);
      glVertex2d(55,30);
      glVertex2d(45,30);
      glEnd();

      //RED light
      x_c = 50; y_c = 70; r = 6;
      glBegin(GL_POLYGON);
      for (i = 0; i <= 10000; i++) {
            theta += 0.001;
            x_i = x_c + r * cos(theta);
            y_i = y_c + r * sin(theta);
            glColor3f(1, 0, 0);
            glVertex2d(x_i, y_i);
      }
      glEnd();

      //YELLOW light
      x_c = 50; y_c = 55; r = 6;
      glBegin(GL_POLYGON);
      for (i = 0; i <= 10000; i++) {
            theta += 0.001;
            x_i = x_c + r * cos(theta);
            y_i = y_c + r * sin(theta);
            glColor3f(1, 1, 0);
            glVertex2d(x_i, y_i);
      }
      glEnd();

      //GREEN light
      x_c = 50; y_c = 40; r = 6;
      glBegin(GL_POLYGON);
      for (i = 0; i <= 10000; i++) {
            theta += 0.001;
            x_i = x_c + r * cos(theta);
            y_i = y_c + r * sin(theta);
            glColor3f(0, 1, 0);
            glVertex2d(x_i, y_i);
      }
      glEnd();

}
void myInit(void)
{
      glClearColor(1.0, 0.5, 0.5, 0.0);
      glColor3f(255.0f / 255.0f, 255.0f / 255.0f, 255.0f / 255.0f);
      glPointSize(4.0);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0, 100, 0, 100);
}

void myDisplay(void)
{
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1.0, 1.0, 0.0);
      glPointSize(1.0);
      draw_Traffic_Light();

      glFlush();
}

void main(int argc, char** argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize(500, 500);
      glutInitWindowPosition(300, 10);
      glutCreateWindow("4-b : Draw diags with Circle and lines");
      glutDisplayFunc(myDisplay);
      myInit();
      glutMainLoop();

}
```

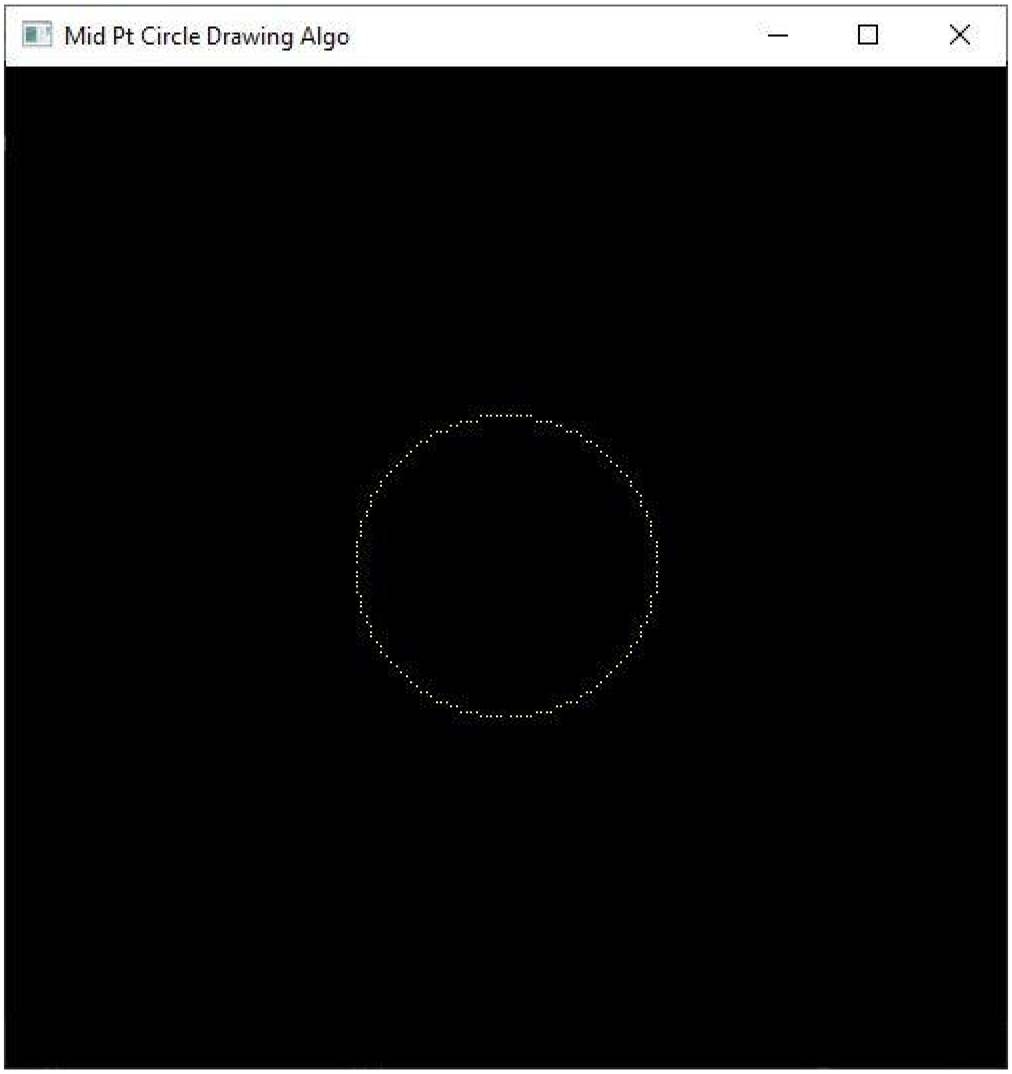
OUTPUT SNAPSHOTS :

4-A) Center : (0,0)

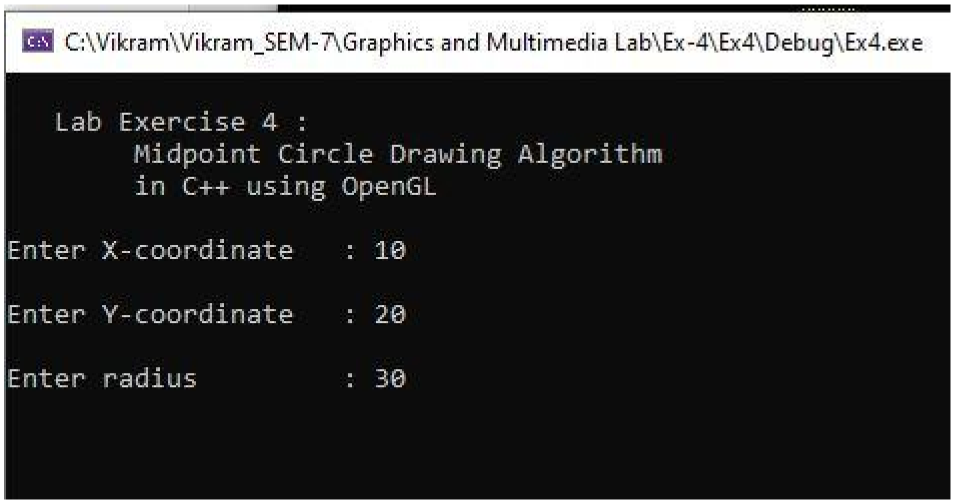


C:\Vikram\Vikram_SEM-7\Graphics and Multimedia Lab\Ex-4\Ex4\Debug\Ex4.exe

```
   Lab Exercise 4 :
        Midpoint Circle Drawing Algorithm
        in C++ using OpenGL

Enter X-coordinate    : 0

Enter Y-coordinate    : 0

Enter radius          : 30
```

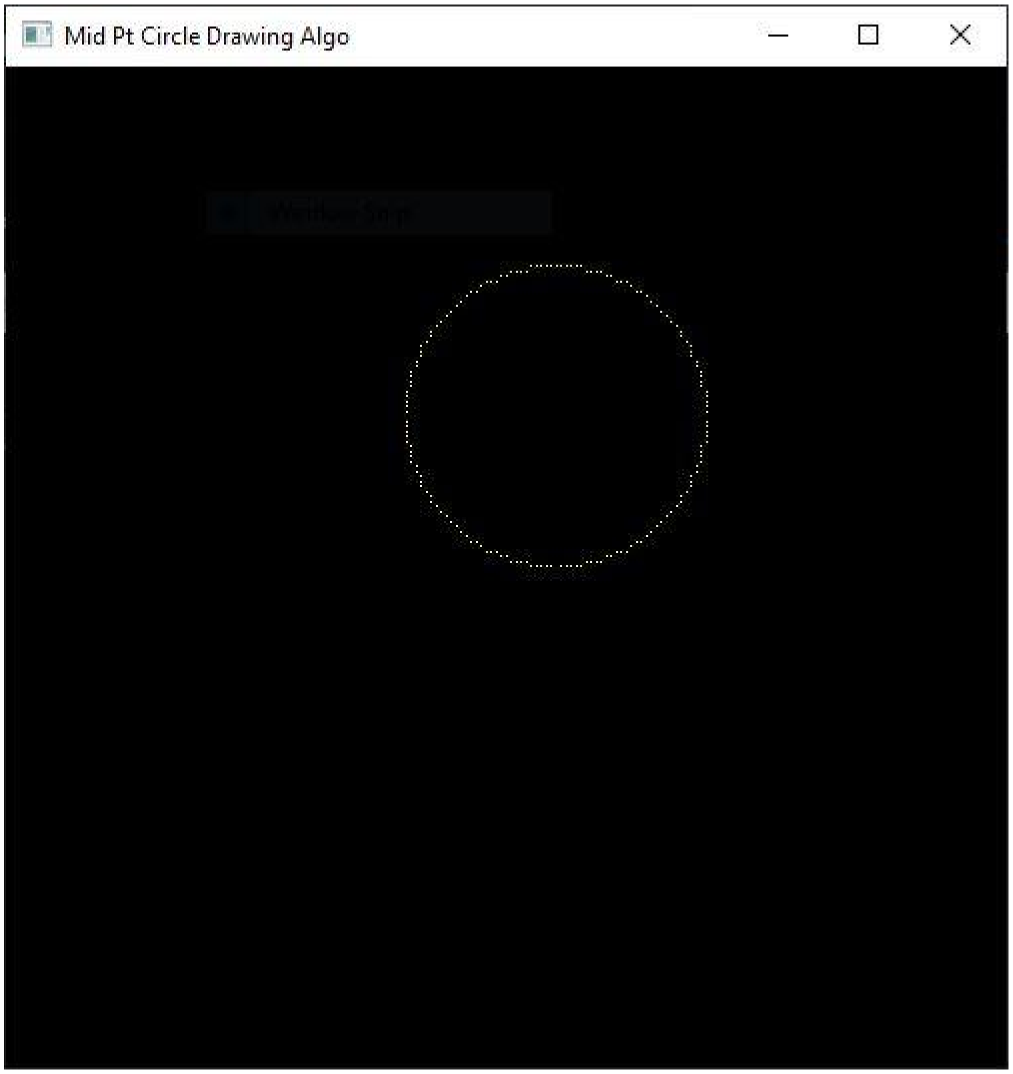


Center : (10,20)



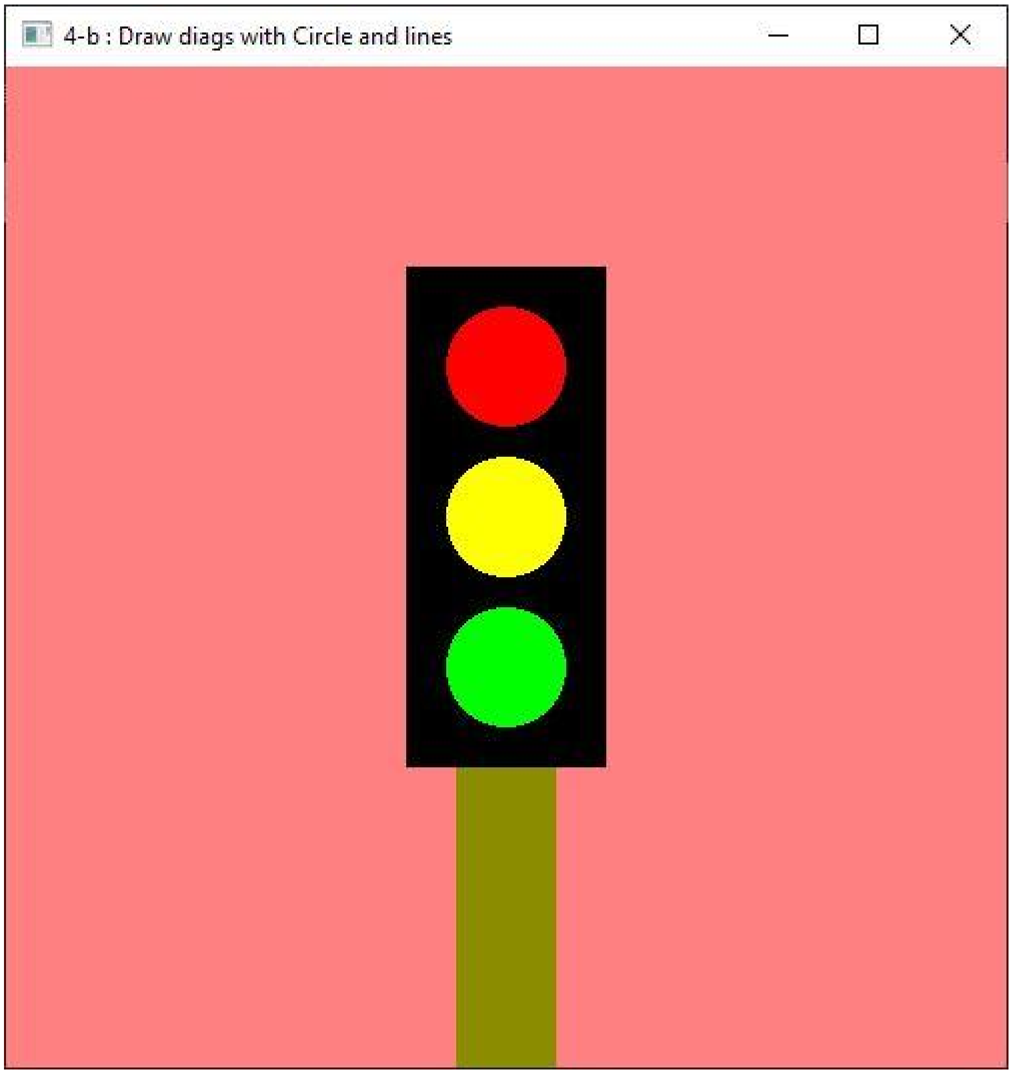
C:\Vikram\Vikram_SEM-7\Graphics and Multimedia Lab\Ex-4\Ex4\Debug\Ex4.exe

```
   Lab Exercise 4 :
        Midpoint Circle Drawing Algorithm
        in C++ using OpenGL

Enter X-coordinate   : 10

Enter Y-coordinate   : 20

Enter radius         : 30
```



4-B)

Diagram with Lines and Circle



<u>CONCLUSION :</u>

Thus the circles with centers (0,0) and (x_c,y_c) were drawn using the Midpoint Circle drawing Algorithm, and a diagram involving circles and lines were drawn.