Practical-4

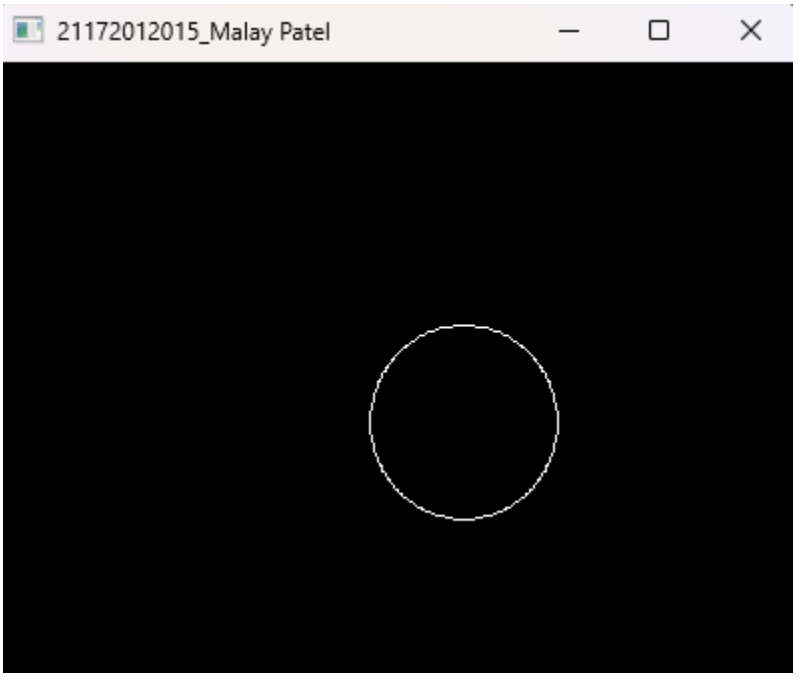
1.Write a C/C++ Program to draw a circle using a midpoint circlealgorithm.

Code :

```
namespace gp4_1
{
	void setPixel(int x, int y) {
		glBegin(GL_POINTS);
		glVertex2d(x, y);
		glEnd();
	}
	void display();
	void circleMidpoint(int, int, int);
	void circleplotpoints(int, int, int, int);
	void reshape(int, int);
	void init()
	{
		glClearColor(0.0, 0.0, 0.0, 0.0);
	}
	void main(int argc, char** argv) {
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
		glutInitWindowPosition(640, 480);
		glutInitWindowSize(400, 415);
		glutCreateWindow("21172012015_Malay Patel");
		glutDisplayFunc(display);
		glutReshapeFunc(reshape);
		init();
		glutMainLoop();
	}
	void circleMidpoint(int xCenter, int yCenter, int radius) {
		int x = 0;
		int y = radius;
		int p = 1 - radius;
		void circleplotpoints(int, int, int, int);
		circleplotpoints(xCenter, yCenter, x, y);
		while (x < y) {
			x++;
			if (p < 0)
				p += 2 * x + 1;
			else {
				y--;
				p += 2 * (x - y) + 1;
			}
			circleplotpoints(xCenter, yCenter, x, y);
		}
	}
```

```
	void circleplotpoints(int xCenter, int yCenter, int x, int y) {
		setPixel(xCenter + x, yCenter + y);
		setPixel(xCenter - x, yCenter + y);
		setPixel(xCenter + x, yCenter - y);
		setPixel(xCenter - x, yCenter - y);
		setPixel(xCenter + y, yCenter + x);
		setPixel(xCenter - y, yCenter + x);
		setPixel(xCenter + y, yCenter - x);
		setPixel(xCenter - y, yCenter - x);
	}
	void display()
	{
		glClear(GL_COLOR_BUFFER_BIT);
		circleMidpoint(50, 40, 70);
		glLoadIdentity();
		glFlush();
	}
	void reshape(int w, int h) {
		glViewport(0, 0, (GLsizei)w, (GLsizei)h);
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(-300, 300, -300, 300);
		glMatrixMode(GL_MODELVIEW);
	}
}
```

Output:



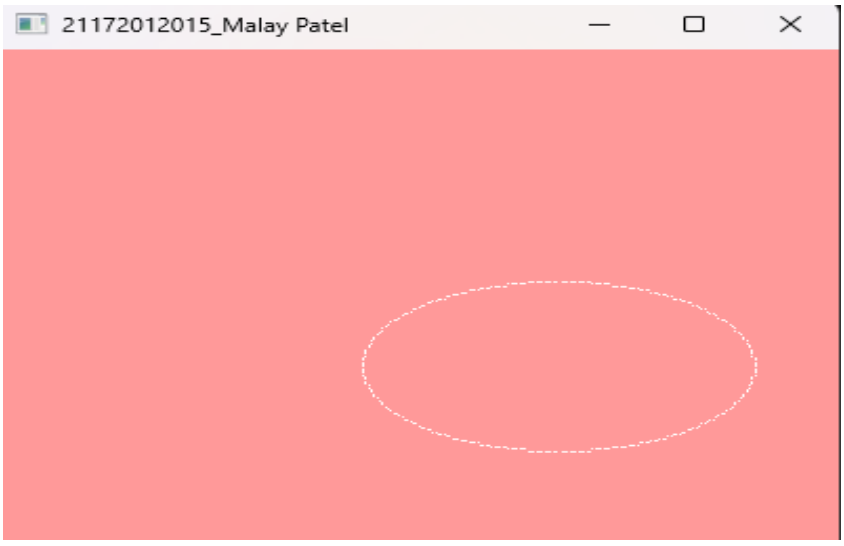
2. Write a C/C++ Program to draw an ellipse using midpoint ellipse algorithm.

Code:

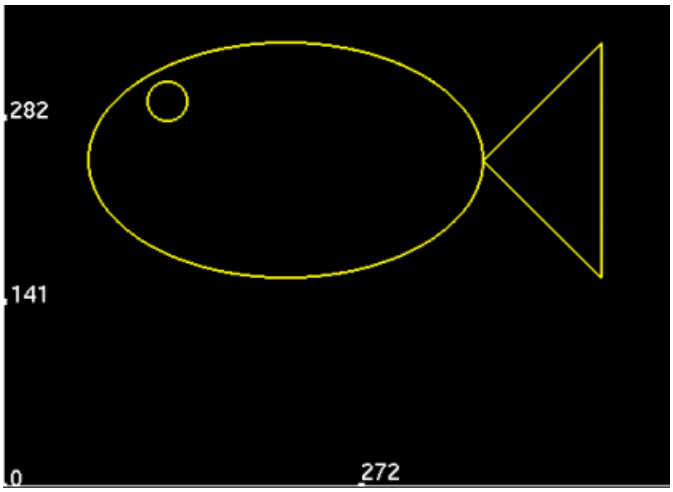
```
namespace gp4_2
{
	void setPixel(int x, int y) {
		glBegin(GL_POINTS);
		glVertex2d(x, y);
		glEnd();
	}
	void display();
#define ROUND(a)(a < 0 ? (int)a - 0.5 : (int)a + 0.5);
	void ellipseMidpoint(int, int, int, int);
	void ellipsePlotPoints(int, int, int, int);
	void reshape(int, int);
	void init() {
		glClearColor(1.0, 0.6, 0.6, 1.0);
	}
	void main(int argc, char** argv) {
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_RGB);
		glutInitWindowPosition(500, 500);
		glutInitWindowSize(400, 400);
		glutCreateWindow("21172012015_Malay Patel");
		glutDisplayFunc(display);
		glutReshapeFunc(reshape);
		init();
		glutMainLoop();
	}
	void ellipseMidpoint(int xCenter, int yCenter, int Rx, int Ry) {
		int Rx2 = Rx * Rx, Ry2 = Ry * Ry, twoRx2 = 2 * Rx2, twoRy2 = 2 *
			Ry2;
		int p, x = 0, y = Ry, px = 0, py = twoRx2 * y;
		p = ROUND((float)((float)Ry2 - (float)(Rx2 * Ry) + (float)(0.25 *
			(float)Rx2)));
		while (px < py)
		{
			x++;
			px += twoRy2;
			if (p < 0)
				p += Ry2 + px;
			else {
				y--;
				py -= twoRx2;
				p += Ry2 + px - py;
			}
			ellipsePlotPoints(xCenter, yCenter, x, y);
		}
		p = ROUND(Ry2 * (x + 0.5) * (x + 0.5) + Rx2 * (y - 1) * (y - 1) -
			Rx2
			* Ry2);
```

```
while (y > 0)
		{
			y--;
			py -= twoRx2;
			if (p > 0)
				p += Rx2 - py;
			else {
				x++;
				px += twoRy2;
				p += Rx2 - py + px;
			}
			ellipsePlotPoints(xCenter, yCenter, x, y);
		}
	}
	void ellipsePlotPoints(int xCenter, int yCenter, int x, int y) {
		glBegin(GL_POINTS);
		glVertex2i(xCenter + x, yCenter + y);
		glVertex2i(xCenter - x, yCenter + y);
		glVertex2i(xCenter + x, yCenter - y);
		glVertex2i(xCenter - x, yCenter - y);
		glEnd();
	}
	void display() {
		glClear(GL_COLOR_BUFFER_BIT);
		ellipseMidpoint(50, 0, 70, 40);
		glFlush();
	}
	void reshape(int w, int h) {
		glViewport(0, 0, (GLsizei)w, (GLsizei)h);
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(-150, 150, -150, 150);
		glMatrixMode(GL_MODELVIEW);
	}
}
```

Output:



3. Write a C/C++ Program to draw fish using a circle and ellipse



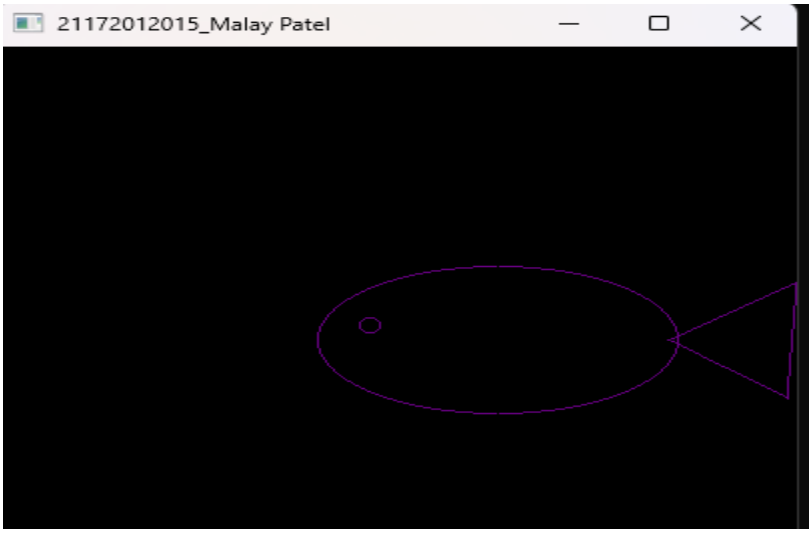
Code:

```
namespace gp4_3 {
	void setPixel(int x, int y) {
		glBegin(GL_POINTS);
		glVertex2d(x, y);
		glEnd();
	}
	void display();
#define ROUND(a)(a < 0 ? (int)a - 0.5 : (int)a + 0.5);
	void ellipseMidpoint(int, int, int, int);
	void ellipsePlotPoints(int, int, int, int);
	void circleMidpoint(int, int, int);
	void circlePlotPoints(int, int, int, int);
	void reshape(int, int);
	void init() {
		glClearColor(0.0, 0.0, 0.0, 0.0);
	}
	void main(int argc, char** argv) {
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_RGB);
		glutInitWindowPosition(500, 500);
		glutInitWindowSize(400, 400);
		glutCreateWindow("21172012015_Malay Patel");
		glutDisplayFunc(display);
		glutReshapeFunc(reshape);
		init();
		glutMainLoop();
	}
	void ellipseMidpoint(int xCenter, int yCenter, int Rx, int Ry) {
		int Rx2 = Rx * Rx, Ry2 = Ry * Ry, twoRx2 = 2 * Rx2, twoRy2 = 2 *
			Ry2;
		int p, x = 0, y = Ry, px = 0, py = twoRx2 * y;
		p = ROUND((float)((float)Ry2 - (float)(Rx2 * Ry) + (float)(0.25 *
			(float)Rx2)));
		while (px < py)
```

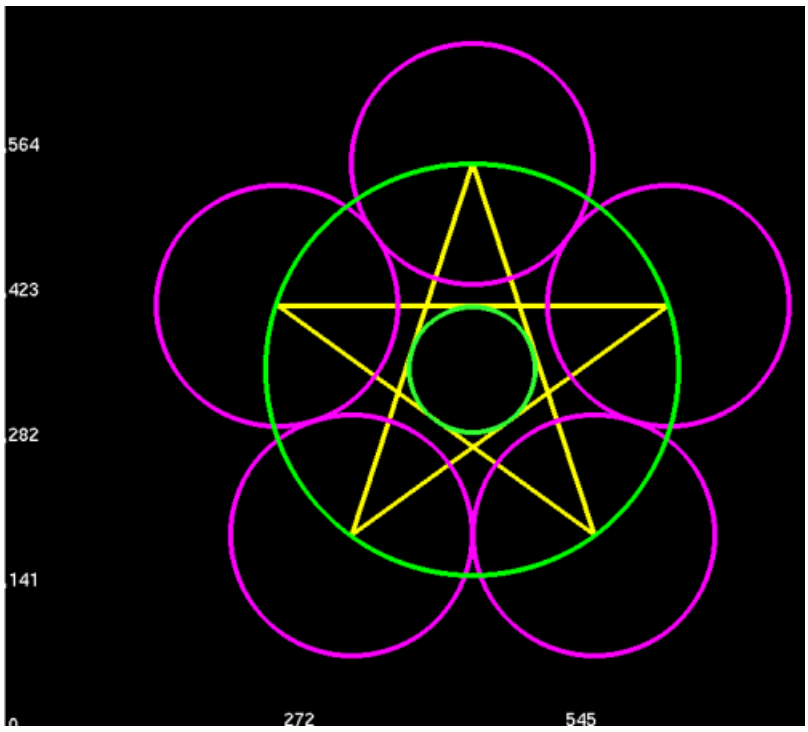
```
		{
			x++;
			px += twoRy2;
			if (p < 0)
				p += Ry2 + px;
			else {
				y--;
				py -= twoRx2;
				p += Ry2 + px - py;
			}
			ellipsePlotPoints(xCenter, yCenter, x, y);
		}
		p = ROUND(Ry2 * (x + 0.5) * (x + 0.5) + Rx2 * (y - 1) * (y - 1) -
			Rx2
			* Ry2);
		while (y > 0)
		{
			y--;
			py -= twoRx2;
			if (p > 0)
				p += Rx2 - py;
			else {
				x++;
				px += twoRy2;
				p += Rx2 - py + px;
			}
			ellipsePlotPoints(xCenter, yCenter, x, y);
		}
	}
	void ellipsePlotPoints(int xCenter, int yCenter, int x, int y) {
		glBegin(GL_POINTS);
		glVertex2i(xCenter + x, yCenter + y);
		glVertex2i(xCenter - x, yCenter + y);
		glVertex2i(xCenter + x, yCenter - y);
		glVertex2i(xCenter - x, yCenter - y);
		glEnd();
	}
	void circleMidpoint(int xCenter, int yCenter, int radius) {
		int x = 0;
		int y = radius;
		int p = 1 - radius;
		void circlePlotPoints(int, int, int, int);
		circlePlotPoints(xCenter, yCenter, x, y);
		while (x < y) {
			x++;
			if (p < 0)
				p += 2 * x + 1;
			else {
				y--;
				p += 2 * (x - y) + 1;
			}
			circlePlotPoints(xCenter, yCenter, x, y);
		}
	}
	void circlePlotPoints(int xCenter, int yCenter, int x, int y) {
		setPixel(xCenter - x, yCenter - y);
		setPixel(xCenter + x, yCenter - y);
		setPixel(xCenter - x, yCenter + y);
```

```
            setPixel(xCenter + x, yCenter + y);
            setPixel(xCenter - y, yCenter - x);
            setPixel(xCenter + y, yCenter - x);
            setPixel(xCenter - y, yCenter + x);
            setPixel(xCenter + y, yCenter + x);
      }
      void dda(float x1, float y1, float x2, float y2) {
            float dx = x2 - x1, dy = y2 - y1;
            float xInc = 0, yInc = 0, x = x1, y = y1, steps = 0;
            steps = (fabs(dx) > fabs(dy)) ? fabs(dx) : fabs(dy);
            xInc = dx / (float)steps; yInc = dy / (float)steps;
            setPixel(x, y);
            for (int k = 0; k < steps; k++)
            {
            x += xInc; y += yInc;
                  setPixel(x, y);
            }
            setPixel(x, y);
      }
      void display() {
            glClear(GL_COLOR_BUFFER_BIT);
            glColor3f(0.4, 0.0, 0.5);
            ellipseMidpoint(50, 0, 90, 50);
            circleMidpoint(-14, 10, 5);
            dda(135, 0, 200, 40);
            dda(200, 40, 195, -40);
            dda(195, -40, 135, 0);
            glFlush();
      }
      void reshape(int w, int h) {
            glViewport(0, 0, (GLsizei)w, (GLsizei)h);
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            gluOrtho2D(-200, 200, -200, 200);
            glMatrixMode(GL_MODELVIEW);
      }
}
```

Output:



4.Write a C/C++ Program to draw according to the figure below.



Code:

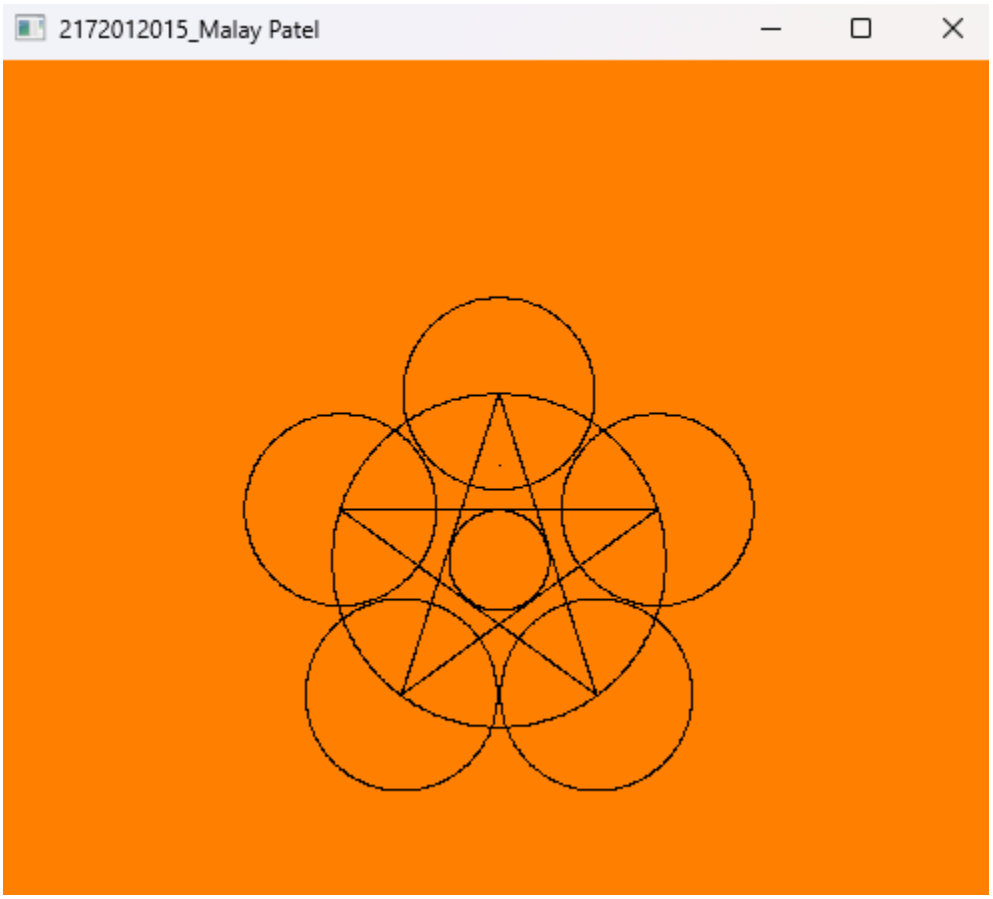
```
namespace gp4_4 {
	void display();
	void reshape(int, int);
	void init()
	{
		glClearColor(1.0, 0.5, 0.0, 0.0);
	}
	void DDAline(float x1, float x2, float y1, float y2);
	void MidpointCiricle(int Xc, int Yc, int r);
	void CirclePlotPoint(int Xc, int Yc, int x, int y);
	void MidpointEllipse(int Xc, int Yc, int Rx, int Ry);
	void main(int argc, char** argv) {
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_RGB);
		glutInitWindowPosition(200, 100);
		glutInitWindowSize(500, 500);
		glutCreateWindow("2172012015_Malay Patel");
		init();
		glutDisplayFunc(display);
		glutReshapeFunc(reshape);
		glutMainLoop();
	}
	void setPixel(float x, float y)
	{
		glBegin(GL_POINTS);
		glVertex2d(x, y);
```

```
	glEnd();
}
void DDAline(float x1, float y1, float x2, float y2) {
	float dx = x2 - x1, dy = y2 - y1;
	float xInc = 0, yInc = 0, x = x1, y = y1, steps = 0;
	steps = (fabs(dx) > fabs(dy)) ? fabs(dx) : fabs(dy);
	xInc = dx / float(steps);
	yInc = dy / float(steps);
	setPixel(x, y);
	for (int k = 0; k < steps; k++) {
		x += xInc;
		y += yInc;
		setPixel(x, y);
	}
	setPixel(x, y);
}
void MidpointCiricle(int Xc, int Yc, int r) {
	int x = 0, y = r, p = 1 - r;
	void CirclePlotPoint(int, int, int, int);
	CirclePlotPoint(Xc, Yc, x, y);
	setPixel(x, y);
	while (x < y) {
		x++;
		if (p < 0) {
			p += 2 * x + 1;
		}
		else {
			y--;
			p += 2 * (x - y) + 1;
		}
		CirclePlotPoint(Xc, Yc, x, y);
	}
}
void CirclePlotPoint(int Xc, int Yc, int x, int y) {
	setPixel(Xc + x, Yc + y);
	setPixel(Xc - x, Yc + y);
	setPixel(Xc + x, Yc - y);
	setPixel(Xc - x, Yc - y);
	setPixel(Xc + y, Yc + x);
	setPixel(Xc - y, Yc + x);
	setPixel(Xc + y, Yc - x);
	setPixel(Xc - y, Yc - x);
}
void MidpointEllipse(int Xc, int Yc, int Rx, int Ry) {
	float x = 0;
	float y = Ry;//(0,ry)
	//slope
	float dx = 2 * (Ry * Ry) * x;
	float dy = 2 * (Rx * Rx) * y;
	//----------------------Region-1-------------------------//
	float p1 = Ry * Ry - (Rx * Rx) * Ry + (Rx * Rx) * (0.25);
	while (dx < dy) {
		//plot (x,y)
		setPixel(Xc + x, Yc + y);
		setPixel(Xc - x, Yc + y);
		setPixel(Xc + x, Yc - y);
		setPixel(Xc - x, Yc - y);
		if (p1 < 0) {
```

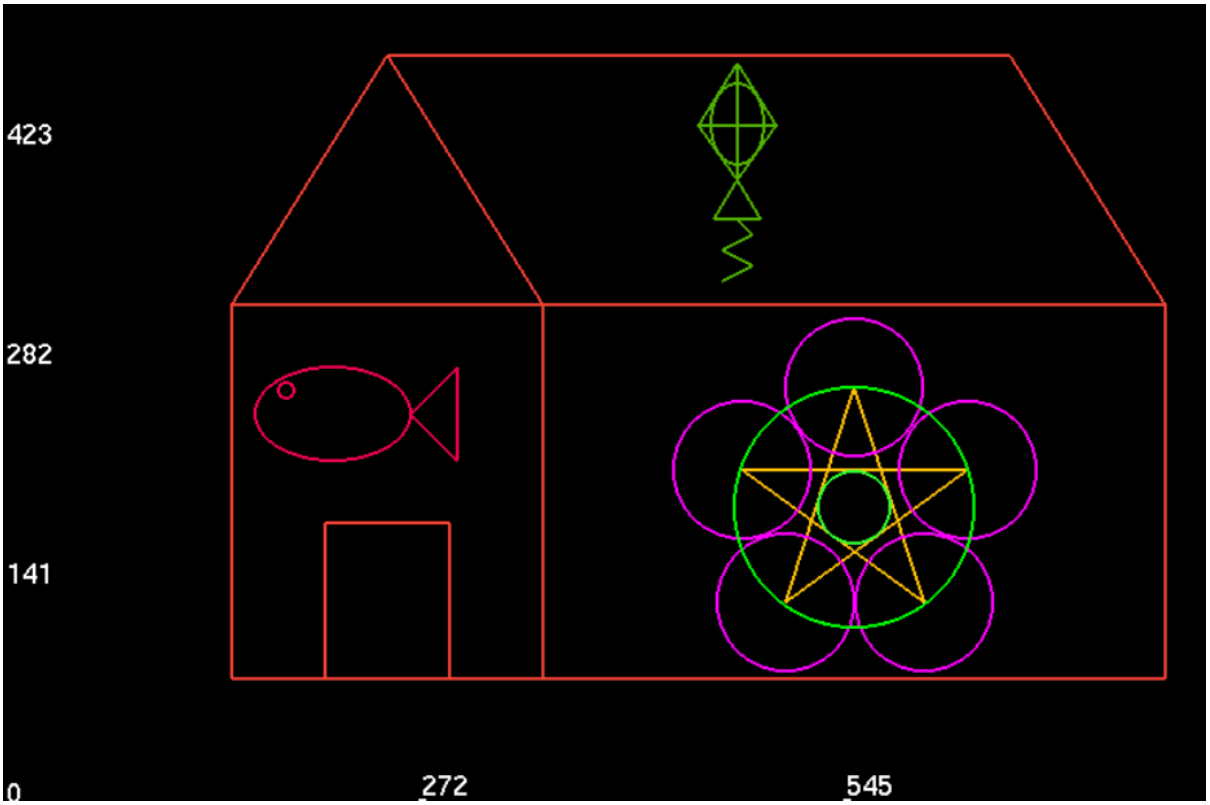
```
                        x = x + 1;
                        dx = 2 * (Ry * Ry) * x;
                        p1 = p1 + dx + (Ry * Ry);
                }
                else {
                        x = x + 1;
                        y = y - 1;
                        dx = 2 * (Ry * Ry) * x;
                        dy = 2 * (Rx * Rx) * y;
                        p1 = p1 + dx - dy + (Ry * Ry);
                }
        }
        //ploting for 2nd region of 1st quardant and the slope will be > -1
        //----------------------Region-2-----------------------//
        float p2 = (Ry * Ry) * (x + 0.5) * (x + 0.5) + (Rx * Rx) * (y - 1) *
                (y
                        - 1) - (Rx * Rx) * (Ry * Ry);
        while (y > 0) {
                //plot (x,y)
                setPixel(Xc + x, Yc + y);
                setPixel(Xc - x, Yc + y);
                setPixel(Xc + x, Yc - y);
                setPixel(Xc - x, Yc - y); //glEnd();
                if (p2 > 0) {
                        x = x;
                        y = y - 1;
                        dy = 2 * (Rx * Rx) * y;
                        //dy = 2 * rx * rx *y;
                        p2 = p2 - dy + (Rx * Rx);
                }
                else {
                        x = x + 1;
                        y = y - 1;
                        dy = dy - 2 * (Rx * Rx);
                        dx = dx + 2 * (Ry * Ry);
                        p2 = p2 + dx - dy + (Rx * Rx);
                }
        }
}
void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        glColor3f(0.0f, 0.0f, 0.0f);
        //glPointSize(5.0f);
        MidpointCiricle(0, 0, 200);
        DDAline(0, 200, -117, -161);
        DDAline(-117, -161, 190, 61);
        DDAline(190, 61, -190, 61);
        DDAline(-190, 61, 117, -161);
        DDAline(117, -161, 0, 200);
        MidpointCiricle(0, 0, 60);
        MidpointCiricle(0, 200, 115);
        MidpointCiricle(190, 61, 115);
        MidpointCiricle(117, -161, 115);
        MidpointCiricle(-117, -161, 115);
        MidpointCiricle(-190, 61, 115);
        glFlush();
}
```

```
	void reshape(int w, int h) {
		glViewport(0, 0, w, h);
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(-600, 600, -600, 600);
		glMatrixMode(GL_MODELVIEW);
	}
}
```

Output:



5.Write a C/C++ Program to draw according to the figure below.



Code:

```
namespace gp4_5 {
	void display();
	void reshape(int, int);
	void init()
	{
		glClearColor(1.0, 0.5, 0.0, 0.0);
	}
	void DDAline(float x1, float x2, float y1, float y2);
	void MidpointCiricle(int Xc, int Yc, int r);
	void CirclePlotPoint(int Xc, int Yc, int x, int y);
	void MidpointEllipse(int Xc, int Yc, int Rx, int Ry);
	void main(int argc, char** argv) {
		glutInit(&argc, argv);
		glutInitDisplayMode(GLUT_RGB);
		glutInitWindowPosition(200, 100);
		glutInitWindowSize(500, 500);
		glutCreateWindow("21172012015_Malay Patel");
		init();
		glutDisplayFunc(display);
		glutReshapeFunc(reshape);
		glutMainLoop();
	}
```

```
void setPixel(float x, float y) {
	glBegin(GL_POINTS);
	glVertex2d(x, y);
	glEnd();
}
void DDAline(float x1, float y1, float x2, float y2) {
	float dx = x2 - x1, dy = y2 - y1;
	float xInc = 0, yInc = 0, x = x1, y = y1, steps = 0;
	steps = (fabs(dx) > fabs(dy)) ? fabs(dx) : fabs(dy);
	xInc = dx / float(steps);
	yInc = dy / float(steps);
	setPixel(x, y);
	for (int k = 0; k < steps; k++) {
		x += xInc;
		y += yInc;
		setPixel(x, y);
	}
	setPixel(x, y);
}
void MidpointCiricle(int Xc, int Yc, int r) {
	int x = 0, y = r, p = 1 - r;
	void CirclePlotPoint(int, int, int, int);
	CirclePlotPoint(Xc, Yc, x, y);
	setPixel(x, y);
	while (x < y) {
		x++;
		if (p < 0) {
			p += 2 * x + 1;
		}
		else {
			y--;
			p += 2 * (x - y) + 1;
		}
		CirclePlotPoint(Xc, Yc, x, y);
	}
}
void CirclePlotPoint(int Xc, int Yc, int x, int y) {
	setPixel(Xc + x, Yc + y);
	setPixel(Xc - x, Yc + y);
	setPixel(Xc + x, Yc - y);
	setPixel(Xc - x, Yc - y);
	setPixel(Xc + y, Yc + x);
	setPixel(Xc - y, Yc + x);
	setPixel(Xc + y, Yc - x);
	setPixel(Xc - y, Yc - x);
}
void MidpointEllipse(int Xc, int Yc, int Rx, int Ry) {
	float x = 0;
	float y = Ry;//(0,ry)
	//slope
	float dx = 2 * (Ry * Ry) * x;
	float dy = 2 * (Rx * Rx) * y;
	//----------------------Region-1-----------------------//
	float p1 = Ry * Ry - (Rx * Rx) * Ry + (Rx * Rx) * (0.25);
	while (dx < dy) {
		//plot (x,y)
		setPixel(Xc + x, Yc + y);
		setPixel(Xc - x, Yc + y);
```

```
                setPixel(Xc + x, Yc - y);
                setPixel(Xc - x, Yc - y);
                if (p1 < 0) {
                        x = x + 1;
                        dx = 2 * (Ry * Ry) * x;
                        p1 = p1 + dx + (Ry * Ry);
                }
                else {
                        x = x + 1;
                        y = y - 1;
                        dx = 2 * (Ry * Ry) * x;
                        dy = 2 * (Rx * Rx) * y;
                        p1 = p1 + dx - dy + (Ry * Ry);
                }
        }
        //ploting for 2nd region of 1st quardant and the slope will be > -1
        //----------------------Region-2-------------------------//
        float p2 = (Ry * Ry) * (x + 0.5) * (x + 0.5) + (Rx * Rx) * (y - 1) *
                (y
                        - 1) - (Rx * Rx) * (Ry * Ry);
        while (y > 0) {
                //plot (x,y)
                setPixel(Xc + x, Yc + y);
                setPixel(Xc - x, Yc + y);
                setPixel(Xc + x, Yc - y);
                setPixel(Xc - x, Yc - y); //glEnd();
                if (p2 > 0) {
                        x = x;
                        y = y - 1;
                        dy = 2 * (Rx * Rx) * y;
                        //dy = 2 * rx * rx *y;
                        p2 = p2 - dy + (Rx * Rx);
                }
                else {
                        x = x + 1;
                        y = y - 1;
                        dy = dy - 2 * (Rx * Rx);
                        dx = dx + 2 * (Ry * Ry);
                        p2 = p2 + dx - dy + (Rx * Rx);
                }
        }
}
void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        glColor3f(0.0f, 0.0f, 0.0f);
        //HOUSE
        DDAline(-300, 200, 300, 200);
        DDAline(300, 200, 300, -200);
        DDAline(300, -200, -300, -200);
        DDAline(-300, -200, -300, 200);
        DDAline(-300, -200, -500, -200);
        DDAline(-500, -200, -500, 200);
        DDAline(-500, 200, -300, 200);
        DDAline(-500, 200, -400, 350);
        DDAline(-400, 350, -300, 200);
        DDAline(-400, 350, 200, 350);
        DDAline(200, 350, 300, 200);
```

```
		DDAline(-450, -200, -450, 0);
		DDAline(-450, 0, -350, 0);
		DDAline(-350, 0, -350, -200);
		//FISH
		MidpointEllipse(-400, 100, 60, 30);
		MidpointCiricle(-420, 100, 10);
		DDAline(-340, 100, -310, 120);
		DDAline(-340, 100, -310, 80);
		DDAline(-310, 120, -310, 80);
		//DESIGN
		//MAIN CIRCLE
		MidpointCiricle(0, 0, 100);
		//STAR
		DDAline(0, 100, -58, -80);
		DDAline(-58, -80, 95, 30);
		DDAline(95, 30, -95, 30);
		DDAline(-95, 30, 58, -80);
		DDAline(58, -80, 0, 100);
		//MID CIRCLE
		MidpointCiricle(0, 0, 30);
		//CIRCLE AT EVERY STAR POINT
		MidpointCiricle(0, 100, 57);
		MidpointCiricle(95, 30, 57);
		MidpointCiricle(58, -80, 57);
		MidpointCiricle(-58, -80, 57);
		MidpointCiricle(-95, 30, 57);
		glFlush();
	}
	void reshape(int w, int h) {
		glViewport(0, 0, w, h);
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(-600, 600, -600, 600);
		glMatrixMode(GL_MODELVIEW);
	}
}
```

Output: