Computer graphics practical 3

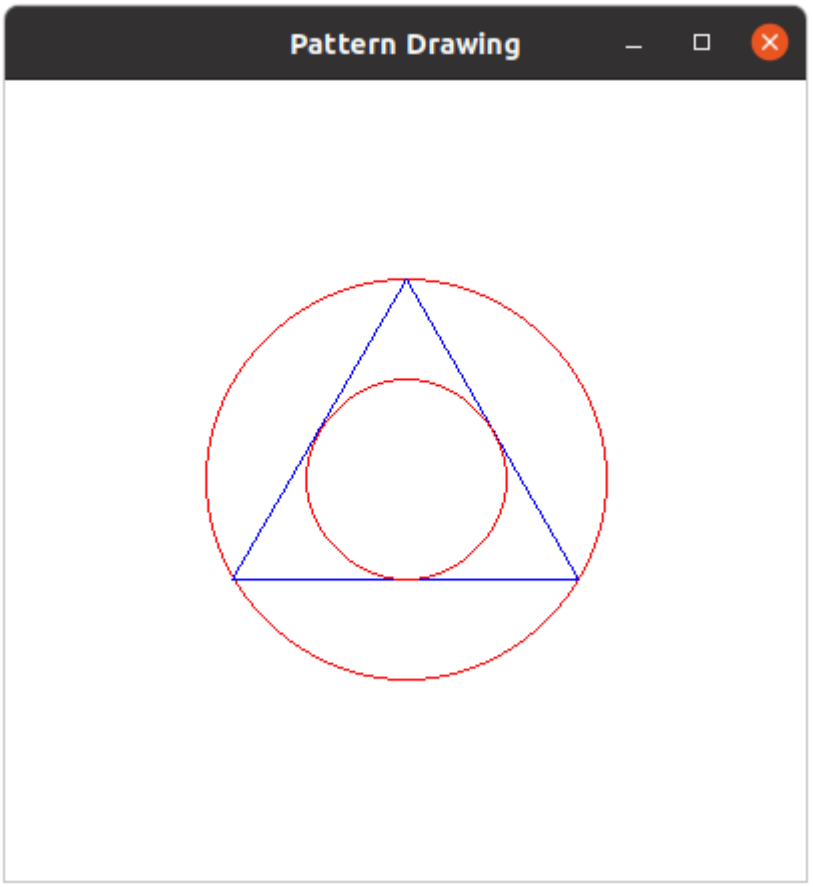
Code:-

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

class DrawPattern {
public:
        DrawPattern(int argc, char** argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(400, 400);
        glutCreateWindow("Pattern Drawing");

        glClearColor(1.0, 1.0, 1.0, 1.0);
        gluOrtho2D(0, 400, 0, 400);

        glutDisplayFunc(display);
        }

        void run() {
        glutMainLoop();
        }

private:
        static void display() {
        glClear(GL_COLOR_BUFFER_BIT);

        // Draw the pattern int xc1 = 200, yc1 = 200, r1 = 50; // Smaller
        circle inside triangle int xc2 = 200, yc2 = 200, r2 = 100; // Larger
        circle outside triangle

        drawBresenhamCircle(xc2, yc2, r2); // Larger circle outside triangle
        drawTriangle(xc2, yc2, r2); // Triangle touching the larger circle
        drawBresenhamCircle(xc1, yc1, r1); // Smaller circle inside triangle

        glFlush();
        }
```

```
static void drawDDALine(int x1, int y1, int x2, int y2) {
glColor3f(0.0, 0.0, 1.0); // Blue color for lines

float dx = x2 - x1; float dy = y2 - y1;
float steps = max(abs(dx), abs(dy));

float xIncrement = dx / steps;
float yIncrement = dy / steps;

float x = x1;
float y = y1;

glBegin(GL_POINTS); for (int
i = 0; i <= steps; i++) {
glVertex2i(round(x),
round(y)); x += xIncrement; y
+= yIncrement;
}
glEnd();
}

static void drawBresenhamCircle(int xc, int yc, int r) {
glColor3f(1.0, 0.0, 0.0); // Red color for circles

int x = 0; int y
= r; int d = 3 -
2 * r;

drawCirclePoints(xc, yc, x, y);

while (x <= y) {
x++;

if (d <= 0)
        d = d + 4 * x + 6;
else {
        y--;
        d = d + 4 * (x - y) + 10;
}

drawCirclePoints(xc, yc, x, y);
}
```

```
        }

        static void drawCirclePoints(int xc, int yc, int x, int y) {
        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); glEnd();
        }

        static void drawTriangle(int xc, int yc, int r) {
        glColor3f(0.0, 0.0, 1.0); // Blue color for lines

        // Calculate vertices of an equilateral triangle float angle
        = 60.0 * M_PI / 180.0; // 60 degrees in radians int x1 =
        xc; int y1 = yc + r; int x2 = xc - r * sin(angle); int y2 = yc -
        r * cos(angle); int x3 = xc + r * sin(angle); int y3 = yc - r *
        cos(angle);

        drawDDALine(x1, y1, x2, y2);
        drawDDALine(x2, y2, x3, y3);
        drawDDALine(x3, y3, x1, y1);
        }
};

int main(int argc, char** argv) {
        DrawPattern pattern(argc,
        argv); pattern.run(); return 0;
}
```

OUTPUT:-


Pattern Drawing