Lab Assignment NO :9

Name:Swapnil Satish Kalshetti

Class: SE A

Roll no: CO2062

Subject: OOP&CG

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class InteractiveCohenSutherland extends Frame implements MouseListener {

    // Clipping window
    final int xmin = 100, ymin = 100, xmax = 300, ymax = 250;

    ArrayList<Point> points = new ArrayList<>();

    final int INSIDE = 0;  // 0000
    final int LEFT = 1;    // 0001
    final int RIGHT = 2;   // 0010
    final int BOTTOM = 4;  // 0100
    final int TOP = 8;     // 1000

    public InteractiveCohenSutherland() {
        super("Interactive Cohen-Sutherland Clipping");
        setSize(500, 500);
        setLocationRelativeTo(null);

        addMouseListener(this);
```

```
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            dispose();
            System.exit(0);
        }
    });

    setVisible(true);
}

private int computeCode(double x, double y) {
    int code = INSIDE;
    if (x < xmin) code |= LEFT;
    else if (x > xmax) code |= RIGHT;
    if (y < ymin) code |= BOTTOM;
    else if (y > ymax) code |= TOP;
    return code;
}

private Point[] clipLine(Point p1, Point p2) {
    double x1 = p1.x, y1 = p1.y, x2 = p2.x, y2 = p2.y;

    int code1 = computeCode(x1, y1);
    int code2 = computeCode(x2, y2);

    boolean accept = false;

    while (true) {
        if ((code1 | code2) == 0) {
            accept = true;
            break;
```

```
} else if ((code1 & code2) != 0) {
    break;
} else {
    int codeOut;
    double x = 0, y = 0;

    if (code1 != 0)
        codeOut = code1;
    else
        codeOut = code2;

    if ((codeOut & TOP) != 0) {
        x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
        y = ymax;
    } else if ((codeOut & BOTTOM) != 0) {
        x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
        y = ymin;
    } else if ((codeOut & RIGHT) != 0) {
        y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
        x = xmax;
    } else if ((codeOut & LEFT) != 0) {
        y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
        x = xmin;
    }

    if (codeOut == code1) {
        x1 = x;
        y1 = y;
        code1 = computeCode(x1, y1);
    } else {
        x2 = x;
```

```
            y2 = y;
            code2 = computeCode(x2, y2);
          }
        }
      }

      if (accept) {
        return new Point[] {
          new Point((int)Math.round(x1), (int)Math.round(y1)),
          new Point((int)Math.round(x2), (int)Math.round(y2))
        };
      } else {
        return null;
      }
    }

    @Override
    public void paint(Graphics g) {
      // draw clipping window
      g.setColor(Color.BLACK);
      g.drawRect(xmin, ymin, xmax - xmin, ymax - ymin);

      if (points.size() == 2) {
        Point p1 = points.get(0);
        Point p2 = points.get(1);

        // draw original line
        g.setColor(Color.GRAY);
        g.drawLine(p1.x, p1.y, p2.x, p2.y);

        // clip and draw result
```

```
            Point[] clipped = clipLine(p1, p2);
            if (clipped != null) {
                g.setColor(Color.RED);
                g.drawLine(clipped[0].x, clipped[0].y, clipped[1].x, clipped[1].y);
            }
        }
    }

    @Override
    public void mouseClicked(MouseEvent e) {
        if (points.size() == 2) points.clear();
        points.add(e.getPoint());
        if (points.size() == 2) repaint();
    }

    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}

    public static void main(String[] args) {
        new InteractiveCohenSutherland();
    }
}
```

Output: