QUESTÃO 1 (3 valores) (1.1 = 1.0; 1.2 = 2.0)

Recorra ao algoritmo *scan-line* para calcular as coordenadas dos *pixels* de preenchimento da área bidimensional definida pelo polígono constituído pelos vértices $\boldsymbol{A}(4,1),\ \boldsymbol{B}(7,4),\ \boldsymbol{C}(7,7),\text{e}\ \boldsymbol{D}(0,5)$

1.1. Apresente o estado da tabela de arestas (ET - *Edge Table*) e tabela de arestas activas (AET - *Active Edge Table*) no início do algoritmo.

1.2. Calcule as coordenadas dos pixels de preenchimento até à 3ª iteração, apresentando cada iteração do algoritmo separadamente, indicando o estado da ET e AET, e apresente no final, de forma gráfica, o preenchimento.

(Resposta: 25 linhas)

QUESTÃO 2 (3 valores)

Ao polígono da questão 1 aplique o algoritmo Surtherland-Hodgman com a janela de recorte $[(x_{min}, y_{min}), (x_{max}, y_{max})] = [(2,2),(6,4)]$

Mostre como os vértices são processados ao longo das várias arestas de recorte (*clippers*) da fronteira da janela e apresente o conjunto final de vértices de saída (*output*). Apresente todos os cálculos, por aresta de recorte, separadamente, e de forma gráfica e o resultado final do recorte.

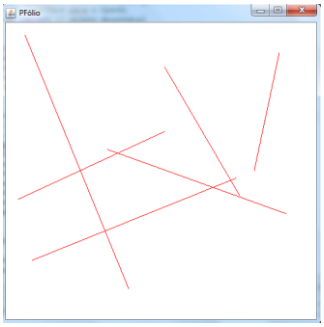
(Resposta: 15 linhas)

QUESTÃO 3 (6 valores)

Codifique em JOGL um programa que desenhe vários segmentos de reta, com inclinações diferentes. Na apresentação do programa deverá minimamente:

- Especificar a funções *init*(), *display*() e *reshape*(), definindo todas as linhas de código necessárias para definição de janela-visor, projecção espacial, configuração da cor, entre outras, que julgue serem necessárias para conseguir que o programa execute o solicitado;
- Identificar as bibliotecas (*import*) necessárias para o código compilar;
- Especificar a função *main*(), definindo os comandos necessários para a instanciação do objecto e criação da visualização;
- Comentar o código, explicando o que ele faz em cada linha;

Na implementação do desenho das retas, será valorizada a utilização de algum algoritmo de desenho de retas (porém, não é obrigatória a sua utilização). A imagem abaixo ilustra um possível resultado visual:



FIM

Resolução questão 3)

```
import com.sun.opengl.util.Animator;
import java.awt.Frame;
import java.awt.event.*;
import javax.media.opengl.*;
import javax.media.opengl.glu.GLU;

public class PFolio implements GLEventListener {
 static GL gl;
 static GLCanvas canvas;

 public void init(GLAutoDrawable drawable) {
   gl = drawable.getGL();
 }

 public void reshape(GLAutoDrawable drawable, int x, int y, int width,int height)
{
    GL gl = drawable.getGL();
    GLU glu = new GLU();
    if (height <= 0) height = 1;
    final float h = (float) width / (float) height;
    gl.glViewport(0, 0, width, height);
    gl.glMatrixMode(GL.GL_PROJECTION);
    gl.glLoadIdentity();
    glu.gluPerspective(45.0f, h, 0.0, 100);
    gl.glMatrixMode(GL.GL_MODELVIEW);
    gl.glLoadIdentity();
 }

 public void display(GLAutoDrawable drawable)
 {
  GL gl = drawable.getGL();
  gl.glLoadIdentity();
  gl.glTranslatef(0,0,-100); // desloca em z

  for (int i = 0; i < 6; i++)
  {
   int x1 = (int)(Math.random() * 80) - 40;
   int y1 = (int)(Math.random() * 80) - 40;
   int x2 = (int)(Math.random() * 80) - 40;
   int y2 = (int)(Math.random() * 80) - 40;
   gl.glColor3f(1,0,0);
   gl.glBegin(GL.GL_LINES);
    gl.glVertex2f(x1,y1);
    gl.glVertex2f(x2,y2);
   gl.glEnd();
  }
  gl.glFlush();
 }

 public static void main(String[] args)
 {
   Frame frame = new Frame("PFólio");
   canvas = new GLCanvas();
   canvas.addGLEventListener(new PFolio());
   frame.add(canvas);
   frame.setSize(500, 500);
   final Animator animator = new Animator(canvas);
   frame.addWindowListener(new WindowAdapter()
```

```
    {
      @Override
      public void windowClosing(WindowEvent e)
      {
        new Thread(new Runnable()
        {
          public void run()
          {
            animator.stop();
            System.exit(0);
          }
        }).start();
      }
    });
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
  }
}
```