Section, Name and ID:

Problem 2: Write a Java method ***public static double[] expReg(double[] data)*** that takes as its argument an array of data points ***double[] data***, and returns a two-element array – the first element being the exponent of an exponential regression and the second element being the amplitude. The exponential regression approximates the data points by a formula

$$y = a\, e^{mx},$$

where the exponent m and the amplitude a are computed as

$$m = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2},\ a = \overline{y} - m\overline{x}$$

Here $\overline{x}$ is the mean of the x coordinates, $\overline{y}$ is the mean of the natural logarithm of y coordinates, $\overline{x^2}$ is the mean of the squares of the x coordinates, and $\overline{xy}$ is the mean of the products of the x and natural logarithm of y coordinates. Use the element indices of the array ***double[] data*** as x coordinates and the element values as y coordinates. For natural logarithm, use the method ***double Math.log()***.

Both result elements are zeros, if at least one data element is non-positive.

```
public static double[] expReg(double[] data)
{   double a[] ~~[illegible]~~ = new int[2] ~~[n]~~;
    int i, cur; ~~double~~ y=0, x=0, X sqr, X mean sqr; ~~[illegible]~~ Xy=0
    int
    for (i=0; i<data.length; i++)
    {             math.log
y=Math.log(y)  y += (data[i]), X += i; X sqr += i·i; Xy += math.log(i)·data[i])
                    ↖ ln needed
    }
    y /= data.length;
    X /= data.length;
    X sqr /= data.length;
    X mean sqr = X·X;
    Xy/= data.length;
    a[0]=(Xy - X·y)/(X sqr - X mean sqr);
```

Use the backside, if needed

```
    a[1] = y - a[0]·X;
    return a;
}
```

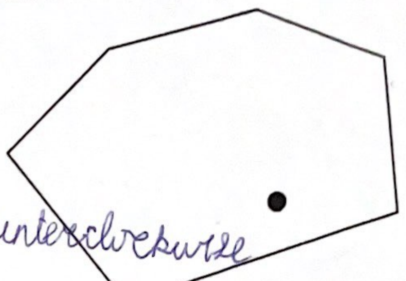
3

OOP.MT-IJ03IJ.H004

ection, Name and ID#

Problem 3: Write a Java function ***public static boolean isInside(double[][] vertex, double x, double y)*** that takes as its argument a ***2-by-n*** array of a convex polygon's vertex coordinates ***double[][] vertex*** – the *x* coordinates in the first row and *y* coordinates in the second row, and ***double x*** and ***double y*** coordinates of a point. It checks, if the point is inside the polygon.

Assume and use a method ***boolean toLeft(double x1, double y1, double x2, double y2, double x0, double y0)*** that takes as its arguments coordinates of three points and returns ***true***, if the third point ***(x0, y0)*** is in the left-hand side, when moving from the first point ***(x1, y1)*** to the second one ***(x2, y2)***; and ***false***, if it is in the right-hand side.



assume the vertexes are given counterclockwise

```
public static boolean isInside(double[][] vertex,
 double x, double y)
{ ~~for~~ int i, n = ~~double~~ vertex[1].length;

    for (i=0; i<n-1; i++)
    { if (!toLeft(vertex[0][i], vertex[1][i],
  vertex[0][i+1], vertex[1][i+1], x, y)
        return false;
    }
    if (!toLeft(vertex[0][n-1], vertex[1][n-1],
      vertex[0][0], vertex[1][0],
      , x, y))
      return false;
    else
      return true;
}
```

3

OOP.MT.170313.H004