Section, Name and ID#: _____

Problem 2: Write a Java method ***public static double[] lin(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 slope of the linear regression and the second element being the intercept. The linear regression approximates the data points by the linear formula

$$y = k\,x + b,$$

where the slope k and the intercept b are computed as

$$k = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\overline{x^2} - \bar{x}^2}, b = \bar{y} - k\,\bar{x}$$

Here $\bar{x}$ is the mean of the x coordinates, $\bar{y}$ is the mean of the 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 y coordinates. Use the element indices of the array ***double[] data*** as x coordinates and the element values as y coordinates. You may assume and use the method double ***mean(double[] a)***.

```
public static double[] lin(double[] data)
    double[] result = new double[2];
    double[] xy = new double[data.length];
    double[] x = new double[data.length];
    double[] x2 = new double[data.length];
    for (int i=0; i < data.length; i++){
        xy[i] = i * data[i];
        x2[i] = i * i;
        x[i] = i;
    }
    for (int i = 0; i < result.length; i+2)
        result[i] = (xy* - x* * y*) / (x2* - x*.pow(2));
        result[i+1] = y* - result[i] * x*;
    return result;
```

filling xy, x2, and x arrays →

→ I assume that * mean(double[] a) has computed the mean of four functions such: xy, x, x2 and y, and denoted the results with $\overline{xy}$, $\bar{x}$, $\bar{y}$ and $\overline{x^2}$ respectively

Use the backside, if needed

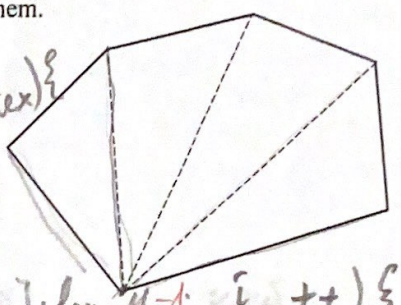
OOP.MT.1F03/J.H009

Section, Name and ID#: _____

Problem 3: Write a Java function ***public static double area(double[][] vertex)*** 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. It returns polygon's area as follows:

1. Divides the polygon into triangles by connecting the ***first*** vertex with the ***nth*** and ***(n+1)st*** vertices;
2. Adds the areas of the constructed triangles using the formula $area = \sqrt{p(p-a)(p-b)(p-c)}$, where ***a***, ***b*** and ***c*** are the sides and ***p = (a + b + c) / 2***.

You may assume and use a method ***double dist(double x1, double y1, double x2, double y2)*** that takes as its arguments coordinates of two points and returns the distance between them.



```
public static double area(double[][] vertex){
    double result = 0;
    double a, b, c, p;
    for(int i = 1; i < vertex[i].length-1; i++){
        a = double dist(double vertex[0][0], double vertex[0,1]
                        double vertex[0,i] double vertex[1,i]
        b = double dist(double vertex[0][0]; double vertex[0][
                        double vertex[0][i+1], double vertex[0][i]
        c = double dist(double vertex[0][i], double vertex[i][
                        double vertex[0][i+1], double vertex
                                                [i][i+1]
        double p = (a+b+c)/2;
        result += Math.sqrt(p(p-a)(p-b)(p-c));}
```

calculating the ~~length~~ distance of each ~~triangle~~ points, then parameter, then the area by $\sqrt{p(p-a)\ldots}$

Use the backside, if needed

OOP.MT.170317.H009