Section, Name and ID#: ______________________________

See MH

0 4/5

AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
CS 120 Introduction to Object-Oriented Programming
MIDTERM EXAM

Date / Time:	Friday, March 17 2017 at 17:30
Duration:	2 hours
Attention:	**ANY TYPE OF COMMUNICATION IS STRICTLY PROHIBITED**
	Write down your section, name and ID# at the top of all used pages

Participation:

Problem 1: Consider below a C++ function ***float kahan(float num1, float num2, float& compensation)*** that implements the ***Kahan Summation Algorithm*** for high-precision compensated summation of two float arguments ***float num1*** and ***float num2***:

```
float kahan(float num1, float num2, float &compensation)
{
        float result;
        num2 -= compensation;
        result = num1 + num2;
        compensation = (result - num1) - num2;
        return result;
}
```

Using this function, write a C++ function ***float pi(int n)*** that computes the value π by the following formula:

$$\pi = 16\sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)5^{2k+1}} - 4\sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)239^{2k+1}} = \left(\frac{16}{1*5}-\frac{4}{1*239}\right)-\left(\frac{16}{3*5^3}-\frac{4}{3*239^3}\right)+\left(\frac{16}{5*5^5}-\frac{4}{5*239^5}\right)-\dots$$

The initial value of ***float compensation*** is ***0.0***.

pi (intn)

int sum, int m, intd

for (int K = 0, K<n, K+1)

m = 16 · (pow (-1K)/(2K+1) · pow(5, 2K+1) - 4(pow(-1)K/

/2K+1) · pow (239, 2K+1)

d = (16pow(-1)K+1)(2(K+1)+1) pow(5, 2(K+1)+1) - 4(pow(-1)(K+1)/

/(2(K+1)+1) pow (239, 2(K+1)+1)

Sum = sum + Kahan (m, d, 0, 0)

Use the backside, if needed

OOP.MT.170317.L079

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

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

Here $\bar{x}$ is the mean of the x coordinates, $\bar{y}$ 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.

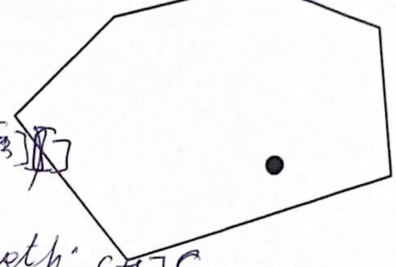
```
public static double [] expReg (double [] data) {
int x1, y1, xy, m, a, x2
for (int x=0, x< data length, x++)
{ xy = xy + x double Math log (data [x]);
x1 = x1+x
y1 = double Math log (data} [x] + y ;
x2 = x2 + pow {x, 2 };
m = (xy/data length) - (x1/data length)(y1/data length)
    ----------------------------------------------------
    (x2/data length) - pow (x1/data length, 2)
a = (y1/data length - m (x1/data length)
double [] barev = new double [] ] ;
    double [0] = m;
    double [1] = a;

        return barev;
```

Use the backside, if needed

OOP.MT. 170317.L043

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.



```
public static boolean isInsid (clos[3][]
vertex , double x, double y) {
for (int i = 0;   vertex[0] length; i++) {
if (boolean lnabt ( vertex[0] [1], vertex [1] [i], vertex[0][i+1]
          vertex [1] [i+1]) {
     return true ;
       {
     else { return false; }
```


OOP.MT.170317.2019