Section, Name and ID#:

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

CS 120 Introduction to Object-Oriented Programming

MIDTERM EXAM

8

0 left

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:

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

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

```
float pi(int n){ float pi=0.
for (int k=0; k<=n; k++){
kahan(pow(-1,k)/((2*k+1)*pow(5,(2*(k+1)))),(pow(-1,k+1)/
((2*(k+1)+1)*pow(5,(2*(k+1)+1))),compensation)
- 4. kahan(pow. half ?
}.
```

2

Use the backside, if needed

Problem 1 of 4

OOP.MT.170317.LOTB

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

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

^ means power(first, second)

3^2 = 9.

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){
    double xres, ylogs, x2s, xpros;
    xres=0; ylogs=0; for(int i=0; i<data.length; i++){ xpros*=i;
    x2s=0; xpros=0;   xres+=i; ylogs+= Math.log(data[i]); x2s+=i^2; }
    double xmean= xres/2; double ymeanlog=ylogs/2.;
    double x2mean=x2s/2; double xproylogmean=(xpros+ylogs)/2;
    double m=(xproylogmean - xmean*ymeanlog)/
    ← x2mean - (xmean^2));
    double a = ymeanlog - m* xmean;
    double[] result; result = new double[2];
    result[0]= m; result[1]= a; return result;
```

This if statement must be in the for loop above. if(data[i] <0){ result[0]=0; result[1]=0;
return result; }.

3

Use the backside, if needed

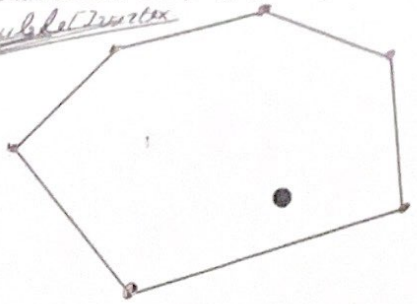
Problem 2 of 4

OOPMT.12031Y.L036.

Section, 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.



```
public static boolean isInside(double[][] vertex, double x, double y)
{
    int g = 0;
    for(int i=0, i<vertex.length; i++){
    for(int j=0; j<vertex.length; j++){
    for(int q=0, q<vertex.length; q++){
    for(int e=0, e<vertex.length; e++){
    if(toLeft(i, j, q, e, x, y)){ if(g==1){ return true; }
                                  else g=2; }
    if(!toLeft(i, j, q, e, x, y){ if(g==2) return true;
                                  else g=1. }
}}}} return false }
```

too many cases

all cases needed

3

Use the backside, if needed

OOP. MT. 17031d. L086