Section, Name and ID#:

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

CS 120 Introduction to Object-Oriented Programming

MIDTERM EXAM

see MH

7

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)
{   int sum P, S;
  for(int k=0; k<n; k++)
   {
      P=16(Pow(-1, k) / (2k+1) • Pow(5, 2k+1)) —
   — 4( Pow(-1, k) / (2k+1) • Pow(239, 2k+1);
      S=16 • (Pow(-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 += float (Kahan( P, S, 0,0);
   }
   return sum;
   }
```

see MH

Use the backside, if needed

3/20.5

OOP.MT.170317.L07

tion, 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)
{     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] + y1;
      X2 = Pow( X2+ Pow(X,2);
   }
   m= (Xy/data.length) - (X/data.length)·(y/data.length)
      ---------------------------------------------------- ;
      (X2/data.length) - Pow(X1,2) Pow(X1/data.length, 2)

   a=(y1/data.length) - m·(X1/data.length);
   double[] aka = int[ new double[2];
     double[0]= m;
     double[1]=a;
           return aka;
```

3/2.1.5

Use the backside, if needed

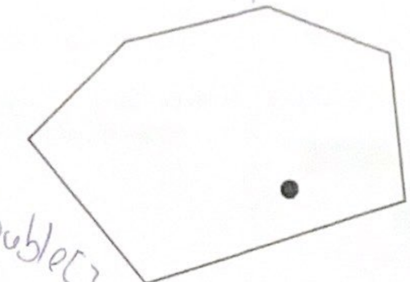
see MH

OOP.MT.170317.L074

...on, 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.



-n

public static boolean isInside(double[] vertex, double x, doubley

2

{

for (int i=0; i< ~~...~~ 2 ~~vertex~~ ~~length~~; i++)

{ for (int j=0; j < vertex[i].length-1, j++)

{

if (~~Boolean vertex[i][j]~~ toLeft (vertex[i][j], vertex[i][j+1], , x,y) == false;

return not inside;

else

~~return inside;~~

inside;

} return we need to check if our point is inside a polygon and for that we check every ~~point~~ cordinate of a polygon. we call method toLeft with two vertexes of polygon and our point if we can true for all vertexes then our point is inside our polygon.

2

DOP.MT.170314.L074

on, Name and ID#: ____________________

Problem 4: Write a Java method ***public static void magicOdd(int[][] square)*** that creates a magic square of an ***odd*** size using the following algorithm:

1. The number *1* goes in the middle of the top row;
2. All numbers are then placed one ***column to the right*** and ***one row up*** from the previous number;
3. Whenever the next number placement is above the top row, stay in the same column and place the number in the bottom row (note the place of *2* instead of the shaded location);
4. Whenever the next number placement is outside of the rightmost column, stay in the same row and place the number in the leftmost column (note the place of *3* instead of the shaded location);
5. When encountering an already filled-in square, place the next number directly below the previous number;
6. When the next number position is outside both a row and a column, place the number directly beneath the previous number (note the place of *7* instead of the shaded location).

	9	2	7
8	1	6	8
3	5	7	3
4	9	2	

~~we have a two dimensional array and we need to change places of rows and columns. and then put at first ~~the element~~ the biggest element of a row.~~

~~we in order to do this we need to make new array ~~with n by m. but~~~~

public static void magicOdd(int[][] squar)

{

1) square[(the number of colums/2] = 1;

2) ~~the~~ make new array and put thoro element of first array but square[n-1]

i = rows

j = colums

new[i+1][j+1] = square[i][j];

3) if it is not replace that number to the first row of a new array. new[0][0] = elemet that is outside

4) row is the same and replace that to the new array

2

COP-MT. 17031J. L074