Section, Name and ID#: data - array

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

9

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;
}
```

0.0

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 num1 = 0
    float num2 = 0
    for (int k=0; k<=n; k++) {
        float n1 = 16*( pow(-1,i) ) / (2*i+1)*pow(5,2*i+1)
    } float num1 += n1
    for(int j {                                   Kahan-?
    for (int j=0; j<=n; j++) {
        float n2 = 4*pow(-1,j) / (2j+1) pow(239, 2*k+1)
        num2 += n2;
    }.
```

Use the backside, if needed արդյունքում 2 բոլոր կոդերը Problem 1 of 4

Kahan ֆունկցիան կարդալուց 2

3

OOP.MT.1Fo3(7.HOSS

բոլոր ցանկացած => Π - ն

Section, Name and ID#: data = array

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

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

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) {
    double[] result = new double[2];
    for (int i=0; i < data.length; i++) {
        if (data[i] ≤ 0) {
            result = {0,0}; only once
        }
        else {
            double meanX = 0;
            double meanY = 0;
            double meanX2 = 0;
            double XY = 0;
            for (int j=0; j < data.length; j++) {
                meanX += j;
            }
            meanX = meanX / data.length;
            for (int i=0; i < data.length; i++) {
                meanY += Math.log(data[i]);
            }
            meanY = meanY / data.length;
```

Use the backside, if needed

OOP.MT.170317.Hoss

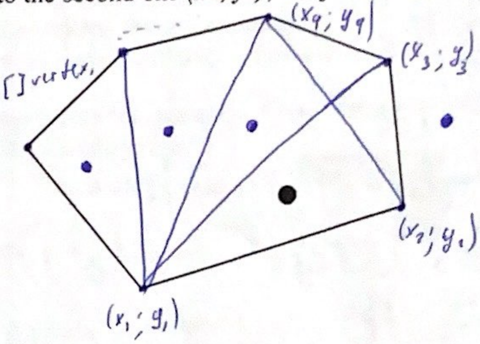
meanX2

```
for (int i=0; i < data.length; i++) {
    meanX2 += pow(i, 2);
}
meanX2 = meanX2 / data.length;

for (int i=0, i < data.length; i++) {
    meanXY += i * Math.log(data[i]);
}
meanXY = meanXY / data.length;

result[0] = (meanXY - meanX * meanY) / (meanX2 - pow(meanX, 2));
result[1] = meanY - meanX * result[0];
return result;
}
}
}
```

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)

~~[illegible]~~

I assume that for the given matrix ~~of values~~ ~~each~~ if we join vertices starting from the 1st (x,y) pair to the end ([illegible]) we will get a convex poligon.

⇓

if we take points $(x_i; y_i)$ ~~and~~ $(x_{i+1}; y_{i+1})$ [line 1]

$(x_{i+1}; y_{i+1})$ $(x_{i+2}; y_{i+2})$ [line 2]

$(x_i; y_i)$ $(x_{i+2}; y_{i+2})$ [line 3]

$x_1 \to x_2$ – [illegible]
$x_2 \to x_3$ – [illegible]
$x_3 \to x_1$ – [illegible]

[illegible]
$x_1 - x_2 \to$ [illegible]
$x_2 - x_3$ – [illegible]
$x_3 - x_1$ – [illegible]

OOP.MT.170317.1055

if for the [line 1] and [line 3] {

the returned boolean values of function toLeft should be opposite. ([illegible] line 1 -[illegible] toLeft-[illegible] line 2-[illegible] toLeft-[illegible], [illegible] line 2 [illegible] line 3 -[illegible] line 1 [illegible] line 2 -[illegible] => polygon-[illegible];

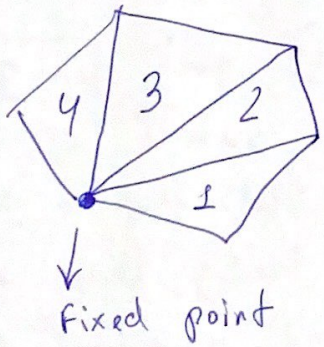
[illegible] for-[illegible]

վիճակ շրջան խնդիրը: Երբ որևէ 3 կետ
հոսար պատկան կետ ունի, ապա ջրային polygon-ի
ներսում է: հակառակ կետում այդ տարբ է ջրային
polygon-ից:

P.S. Օգտվում ենք այն փաստից, որ երբ ջրային polygon-ի ներսում է, ապա այդ երեք կետերի կազմված եռանկյունու ներսում է: Եվ ապա որևէ ենք ջրային ջրային ենք երբ ոչ մի եռանկյունու ներսում չի: Երբ ոչ ապա polygon-ի մեջ չէ:

ամեն օպտիմալ ալգորիթմ ջրին հոսար վերլուծում ենք եռանկյունների և հետո ներկում ենք այդ կետով ու երբ հանդիպում է այլ կետերով կազմված. հետևանքներ բոլոր եռանկյունները: Պատկերում տրված Polygon-ի տեսքերը ներկված կլինեն: Օգտագործելով ենք վերևում նկարագրված ծրագիր արդյունք ստանալ ջրային եռանկյունների ներսում է թե ոչ

2



Section, 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	

square [0][0]

squaremagic [0][0] = square [0][0] + 1

```
public static void magicOdd (int[][] square) {
    int[][] magic_square ;
    for (int i = 0 ; i
```

այդպես պրոցեսը է 1-ի փոխել գնում
ուղիղ վերև աջ գրելուց է առաջին
row-ի middle տեղին.
Ցիկլերի միջ է պետք magic_square-ի
առաջին column-ի տեղի փոխելուց.

Որպեսզի պրոցեդուրան աշխատեցնելու համար
օգտագործի

square_magic [i][j] = square [i][j] + 1.

Use the backside, if needed

Problem 4 of 4

1

OOP.MI.17031T.Hoss