Section, Name and ID#

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:

$$\pi = 2\sum_{k=0}^{n}\frac{(2k-1)!!}{(2k)!!(2k+1)} = \frac{2}{1*1}+\frac{1}{2}*\frac{2}{3}+\frac{1*3}{2*4}*\frac{2}{5}+\frac{1*3*5}{2*4*6}*\frac{2}{7}+\cdots$$

Recall that **n!!** is the product of odd numbers from *1* to ***n***, if ***n*** is odd; and is the product of even numbers from *2* to ***n***, if ***n*** is even. The double factorial of non-positive numbers equals to *1* by definition.

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

```
float pi(int n) {
    result = 0;
    for(int k=0, k<=n, k++){
        result = result + 2(odd(2k-1)/even(2k)*(2k+1));
    }
    return result;
```

```
double odd(int n){ i = i+2
    double result = 1;
    for(int i=1, i<=n, i++)
        result = result * i;
    }
    return ?
double even(int n){
    double result = 1;
    for(int j=2, i<=n, i=i+2)
        result = result * i;
    }
    return ?
```

Kahan?

Q3

Use the backside, if needed

2(odd(2k-1)/even(2k)*(2k+1))

OOP.MT.170317.M062

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)***.

```
#include math.h
public static double[] lin(double [] data){
    for(int i=0; i<= data.length; i++){
    for(int j=data[0]; j<=data[data.length]; j++){
    double k=[mean(i*j) - (mean(i)*mean(j))] /
                array needed
        [mean(pow(i,2)) - mean(pow(mean(i),2))];

    double b= mean(j)- k*mean(i);
    }
}
return double [] arr(k,b);
}
```

2

Use the backside, if needed

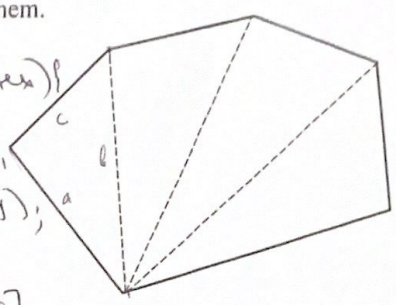
COP. MT. 14831Y. M062

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 ***n^{th}*** 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 A=0; for(int i=0, i<=n, i++){
int a = double dist( vertex[i]_x, vertex[i]_y,
                     vertex[i+1]_x, vertex[i+1],y);

int b = double dist ( vertex[i]_x, vertex[i]_y,
                      vertex[i+2]_x, vertex[i+2]_y);

int c = double dist (a_x, a_y, b_x, b_y);

double p = (a+b+c)/2;

double area = sqrt( p*(p-a)*(p-b)*(p-c));

A = A + area;
}
return A;
```

2.

correct idea
wrong implementation

Use the backside, if needed

Problem 3 of 4

OOP.MT.170317.M062