Section, Name and ID# [redacted]

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

CS 120 Introduction to Object-Oriented Programming

MIDTERM EXAM

6

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 e(int n)*** that computes the value ***e*** by the following formula:

$$e = \sum_{k=0}^{n} \frac{1}{k!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1*2} + \frac{1}{1*2*3} + \cdots$$

Recall that the factorial of non-positive numbers equals to *1* by definition.

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

float e (int n) {float compensation = 0, 0}

loop-e ~~for (n>0; n-1~~

~~while(n>K; n(n-1))~~ ~~while((n>K n-(n-1)(n-2...~~

~~calling function float.~~

~~float Kahen($\frac{1}{n}$~~

while (n>K; (n-1))

calling function float Kahan() to add

our loop nubers($\frac{1}{n!} + \frac{1}{(n-1)!}\frac{1}{(n-2)!}\cdots$)

2

Use the backside, if needed

Problem 1 of 4

full code on the back

OOP.MT.17OS17.L028

[illegible]

[illegible]

[illegible]

sort之后sum[illegible] a, b [illegible]

[illegible] sum.

Wait [illegible]

count << e72

[illegible]

Smart move

Section, Name and ID#: [redacted]

Problem 2: Write a Java method ***public static double[] mean(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 mean value of the data points and the second element being the standard deviation. The standard deviation σ of n numbers a_i is computed as:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n-1}(a_i - mean)^2}{n}}$$

```
public stlid ou ble [] date = { ~~[2][2]~~
  ~~Ru~~ for( ~~int~~ i>0 ; i<data ; i++){
      f = f+ data at((i) ; }
      ay = f(data.lenght) ;
  for(int j = 0 ; j ; j++)
```

$$\frac{\sum_{i=0}^{n}(n-1)}{n}$$ mean

we are computing nenn by loop according to theren then ~~a~~ by filling own array the first 1 of our values. by loop computing our 2nd sd by taekein on count the value of $\boxed{n ; j}$ (loop)

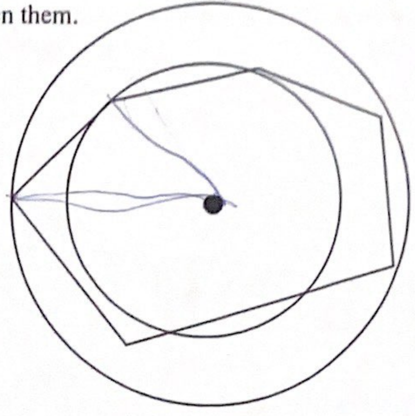
2

Section, Name and ID#: [redacted]

Problem 3: Write a Java function ***public static double thickness(double[][] vertex)*** that takes as its argument a ***2-by-n*** array of polygon's vertex coordinates ***double[][] vertex*** – the *x* coordinates in the first row and *y* coordinates in the second row. It returns polygon's boundary thickness as follows:

1. Computes the center – the mean *x* and *y* vertex coordinates;
2. Returns the difference between the maximal and minimal distances from the center to the vertices.

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.



double vertex = [2][n]

j.s.

Vor thnes = [2][n]

Von mbon = $\left(\frac{xy}{2}\right)$

$\frac{x+y}{2}$ = mem if infut is> thn zodiy

zetorn cerPa (new)

Von Cr.bengtt = α ~~distona~~

if (~~distnce~~ lest (x; nem) ≥ lsge (y; inyn)

zetrn α

if (lengt (y; nm ~~≥~~ lsent(x; nem) ,

1

Use the backside, if needed

Problem 3 of 4

Section, Name and ID#: [redacted]

Problem 4: Implement the following Java methods that swap element values between two 2D integer arrays of the same size ***int[][] a*** and ***int[][] b***:

1. ***public static void swap(int[][] a, int[][] b, int row, int col)*** – swaps element values from the specified row ***int row*** and column ***int col***;
2. ***public static void swapCol(int[][] a, int[][] b, int col)*** – swaps all element values from the specified column ***int col***;
3. ***public static void swapRow(int[][] a, int[][] b, int row)*** – swaps all element values from the specified row ***int row***. Get s bonus, if ***swapRow()*** performs faster than ***swapCol()***.

int[][]a

int[][]b

public static ~~v~~oid swap(int[][]a int[][]b, int row, int col) { for int[][]a(int[][]b(

1) (a[] = b[]' step in purge

int row a + int row b swap values of arr.

2) if int col a + int col b swap rows of a and b

1