Section, Name and ID#:

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

14

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:

1 4 3

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 prod = 1; float sum = 1;
    for(int i = 1; i <= n; i++){
        sum = kahan(sum, 1/(prod*i), &co-
mpensation);
        prod = prod * i;
    }
    return sum;
}
```

3

Use the backside, if needed

Problem 1 of 4

OOP.MT.1703.H.M063

Section, Name and ID#: _____

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 static double[] mean(double[] data){
    double sum=0;
    double mean=0;
    double std=0;
    double[] arr=new ~~arr~~ double[2]; ~~doubl~~
    for(int i=0; i<data.length; i++){
        sum+=~~sum~~data[i];
    }
    mean=~~sum/n;~~ sum/~~data~~.length;
    double sum2=0;
    for(int j=0, j<~~arr~~data.length-1; j++){
        sum2=sum2+Math.pow(data.[j]-mean
                           ,2)
    }
    int ~~std=Math.sqrt(~~
    std=Math.sqrt(sum2/data.length);
    arr[0]=mean;
    arr[1]=std;
    return arr; }
```

4

Use the backside, if needed

Problem 2 of 4

OOP.MT.170317.M063

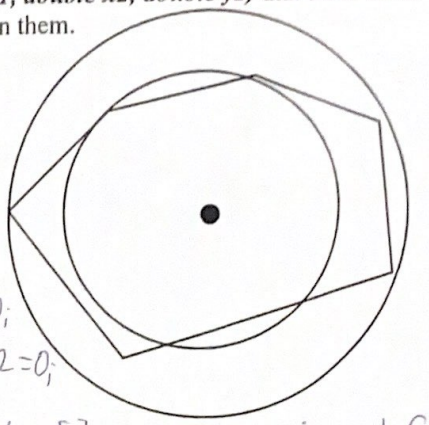
Section, Name and ID#: _____

00 10
01 11
02 12

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.



```
public static double
thickness (double [][] vertex) {
    double mx=0;   double sum1=0;
    double my=0;   double sum2=0;
    for (int i=0; i<vertex[0].length; i++) {
        sum1 = sum1 + vertex[0][i];
        sum2 = sum2 + vertex[1][i];
    }
    mx = sum1/vertex[0].length;
    my = sum2/vertex[0].length;
    double max=0;   double min=0;
    for (int j=0; j<vertex[0].length; j++) {
        if (dist(mx, my, vertex[0][j], vertex[1][j]) > max)
            max = dist(mx, my, vertex[0][j], vertex[1][j]);
        if (dist(mx, my, vertex[0][j], vertex[1][j]) < min)
            min = dist(mx, my, vertex[0][j], vertex[1][j]);
    }
    return thickness = max - min;
}
```

not needed

thickness

double dist=0;

double min=dist(mx, my, vertex[0][0], vertex[0][0]);

for (int i=0; i<2; i++) {

Use the backside, if needed

OOP. MT. 170317. M063

4

Section, Name and ID#: _____

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()***.

```
1 2 3     2 6 4
5 4 2     5 5 3
3 1 3     1 2 1
```

```
public static void swap(int[][]a, int[][] b,
1.  int row, int col){  int temp=0;
    temp = a[row][col];
    a[row][col] = b[row][col];
    b[row][col] = temp;
    int t2=0;
2.  for(int i=0; i<a.length; i++){
    t2 temp=a[i][col];
    a[i][col] = b[i][col];
    b[i][col] = t2;
    }
    int t3=0;
3.  for(int j=0, j<a[0].length; j++){
    t3 = a[row][j];
    a[row][j] = b[row][j];
    b[row][j] = t3;
    }
```

3

Use the backside, if needed

OOP. MT. 170817. M063