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: 12

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***.

1) factorial function
2) for loop,

```
float e(int n){
  float result;
  float sum1, sum2, result, compensation=0.0;
  for(int i=0; i<n; i++){
    sum1 = factorial(i);
    sum2 = factorial(i+1)
    result += float(sum1, sum2, compensation)
  }
  return result;
}
```

sum1 = 0 sum1 = float(0, 1, com.
sum2 = 1. sum2 = i+1.

Assume we have factorial function

3

Use the backside, if needed

OOP.MT.170317.M022

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:

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

firstly in
(in main)

1) takes arg. double[] data
2) returns 2 elem arr
1 mean value, 2

mean = i+i+1+...
n

```
Scanner input = new Scanner(System.in);
int n;
n = input.nextInt();
double[] data = new double[n];
    for (int i=0; i<n; i++){
        data[i] = input.nextDouble;
    }
}
```

```
public static double[] mean(double[] data){
    double mean_val=0; stand_val =0;
    double sum=0; double result=new double [2];
  for (int i=0, i<data.length, i++)
    sum+= data[i];

    mean_val=sum/data.length;
? standard_val +=sqrt(data[i]-mean_val)*(data[i]-mean_val)/data.length,
    }
  result[0]=mean_val;
  result[1]=stand_val;
  return result;

    }
```

3

Use the backside, if needed

OOP.MT.1J0317.M022

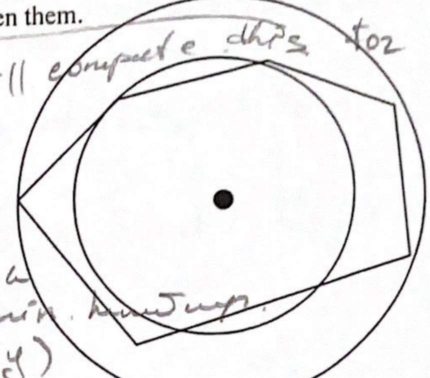
Section, Name and ID#: [illegible]

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.

compute this for vert.



1) ~~[illegible]~~ for loop that will

$meanx = \frac{x_1 + x_2 + ... x_n}{n}$

$meany = y_1 + y_2 + .. y_n$. for loop

2) [illegible] max_x. vertex, row 1 - [illegible]
max_y row 2 - [illegible] min. [illegible]
dist(max_x, max_y, min_x, min_y)

```
1). for(int col=0; col<vertex[0].length, col++){
mean_x+=vertex[0][col]/vertex[0].length;
men_y+=vertex[1][col]/vertex[1].length;
 }.

2). for(int col=0; col<vertex[0].length, col++){
   if(vertex[0][col]<vertex[0][col+1]) {
      max_x = vertex[0][col+1],
   if(vertex[1][col]<vertex[1][col+1]){.
      max_y = vertex[1][col+1];
    }.
  }.

result=dist(max_x, max_y, min_x, min_y);
   return result,
```

int division?

don't needed

[illegible]
min. [illegible]

2

Use the backside, if needed

OOP.MT.170317.M022

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) public static void swap(int[][] a, int[][] b, int row, int col){
    a[row][col] += b[row][col];
    b[row][col] = a[row][col] - b[row][col];
    a[row][col] = a[row][col] - b[row][col]
    }

2) public static void swapCol(int[][] a, int[][] b, int col){
    for(int row=0; row<col; row++){
        swap(a[row][col], b[row][col]; row, col)
    }

3) public static void swapRow(int[][] a, int[][] b, int row){
    for(int col=0; col<a[row].length; col++){
        swap(a[row][col]; b[row][col], row, col)
    }
```

4

Use the backside, if needed

OOP.MT.170317.M022