Section, Name and ID#:

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

~~0~~ 7

Date / Time: Friday, March 17 2017 at 17:30
Duration: 2 hours
Attention: <u>**ANY TYPE OF COMMUNICATION IS STRICTLY PROHIBITED**</u>
<u>***Write down your section, name and ID# at the top of all used pages***</u>

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:

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

num1+ num2 -n

3 - 1 - 2 - 2

Recall that the factorial of non-positive numbers equals to ***1*** by definition.
The initial value of ***float compensation*** is ***0.0***.

```
float kahan(float num1, flat num2, float& compensation)
{
 float result;
 num2 - = compensation;
 result = num1 + num2
 compensation = (result - num1) - num2;
 return result;

~~float e(int n)~~
```

factorial of a number is the previous number +1; n, n+1, n+1+1, n+1+1... so the result that is giving us is every previous number +1 as the result = num1 + num2

we keep 2 element array which has a[0] and a[1]
a[0] - is the sum of all element divided by the quantity of elements
a[1] each element - a[0] ~~%a~~ divid by n and take the square root of it.

~~other problem.~~

```
for (int c = 1; c < n+1, c++) {
  int num = 1
  for(int b >= 1, b < c, b++){
 num *= b
 ~~num~~ kahan
```

Use the backside, if needed

1

OOP.MT.17031J.M024

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){
for int double a=0
double mean=0
double c=0
for (i=0, i<data.length, i++){
a+=data[i];
}
a=a/(data.length);
mean=a-c
double q=0;
for (j=0, j<data.length; j++){
q+=pow((data[i]-mean); 2);
}
double formula=sqrt(q/data.length);
double[] array=new double[2];
double[0]=mean.
double[i]=formula;
return double[];
}
```

3

```
public class TwoDimentionalArrayOperations {
public static double getTotal (double[][] array){
  double total=0;
for(int row=0, row<array.length, row++){
 for(int column=0, col<array[row].length, col++){
     total+=array[row][col];
}
}
public static double getAverage (double[][] array){
 return getTotal(array)/getElementCount(array);
}
public static double getRowTotal(double[][] array, int row);
 {double total=0
 for (int col=0, col<array[row].length; col++){
 total+=array[row][col];
 }return total;
  }
```

Use the backside, if needed

Problem 2 of 4

OOP.MT.17031.M024

```
for(int row=0, row<array.length; row++){
    total += array[row][col];
  } return total;

}
public static double getHighestRow(double[][] array, int row){
        double highest = array[row][0];
for(int col=1; col<array[row].length, col++){
  if (array[row][col] >highest ){
     highest =array[row][col];
}

} return highest;

}public static double getLowestRow(double [][] array, int row){
       double lowest= array[row][0];

 for(int col= 1, col<array[row].length; col++){
     if (int col array[row][col]< lowest){
       lowest =array[row][col];
   }
  }
  return lowest;
 }.
 public static int getElementCount(double[][] array){
    int count=0;
for (int row=0; row<array.length; row++){
      count += array [row]. length
 }return count;
}
public static void main (String [] args){
double[][] studentTestScores = {{67, 53, 24, 98}
                                 {33, 23, 54, 12}};
```

TwoDimensionalArrayOperation.java ✓

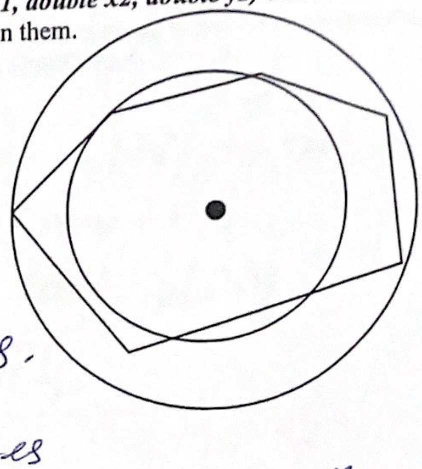
https://github.com/levelupLunch/levelup-java-exercises/blob/master/src/main/java/com/levelup/java/exercises/beginner/

Section, Name and ID#

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.



1. We take the sum of x-es and divide it by the ~~quantity~~ length of x-es and we take the sum of y-es divide it by the ~~quantity~~ length of y-es.

2. we take all the x-es and y-es and we dist() all the numbers that we get from the centre as x's and y's and the biggest and the smallest numbers on the answers.

2/1

```
public static double thickness(double[][] vertex){
  double[][] vertex = 0
for(int ~~double x1~~
  int length = double[0]
  double centerx = 0;
  double centery = 0;
  for(int b = 0, b < lengh, b++){
    center ++ = vertex[0][b];{
  center ++ = vertex [1] ?
```


OOP.MT.1J0317.M024

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)
  for (int row=0; row<array.length, row++){
  for (int col=0; col<array[row].length, col++){
      swap += array[row][col];
```

a[row][col] = ? [row][col] ^ b[row][col]

b[row][col] = ? [row][col] ^ a[row][col]

we take int[][] a, int[][] b, and swap the values for int row and int col, first we compute the length of both rows and columns, specify them and then create a function which will swap their elements,

2. only take the int colum compute its lenght with function

```
for (int col=0; col<array.length, col++){
    and then swap int col elements int a and
    int b;
```

3. the same is done for int row

```
for (int row=0; row<array.length, row++){
```

2

Use the backside, if needed

OOP.MIT.18318.M024