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 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***.

The value e is known number ~ 2,75...

num 1 and num 2 return sum

$\sum_k$ → is a (sum) presented as $\frac{1}{1} \cdots \frac{1}{1*2*3}$

but using this formula we can take both non-positive and positive numbers factorials. discarded as we assumed that their value is 1.

float e(int n)

float e (int n) = $\sum_{k=0}^{n} \frac{1}{k!}$

float result;

$\frac{1}{k!}$ $k! \neq 0$ as $0! = 1$

$n! = n \cdot (n-1)$

Initializing is with 0.0 we then will add each 2,75...

Use the backside, if needed

1

OOP.MT.1503.L035

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: [mean value] ; [standard deviation]

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

#2

1. We need 'for loop' to find sum of array
2. Then dividing this sum into length of array
3. Next loop needed for deviation

```
public static double[]
(alternative example)
magic square [3;3]
public static void (
  int i, j;
  int sum-row, sum-col, sum-diagonal = 0, sum = 0;
  boolean magic = true;
  int [ ] [ ] square = new int [3][3];
  Scanner input = new Scanner (System.in);
  System.out.print ("Enters → ");
    for (i=0; i<3; i++)
    for ( j=0; j<3; j++)
      square [i] [j] = Input.nextInt ();
  System.out.print ("Square);
  for (i=0; i<3; i++) {
  for (j=0; j<3; j++)
  System.out.print (square [i] [j] + " ");
        System.out.println. }
```

//row

```
for (i=0; i<3; i++){
  sum-row = 0
for (j=0; j<3; j++)
  sum-row += square [i] [j];
if (sum-row != sum) {
    magic = false
    break;
```

//column

```
for (j=0; j<3; j++) {
  sum-col = 0;
for (i=0; i<3; i++)
if (sum-col != sum) {
  magic = false;
    break;
```

//diagonal

```
if (magic) {
for (i=0; i<3; i++)
if (i==j)
  sum-diagonal += square [i] [j];
if (sum-diagonal != sum) {
  magic = false;
```

//2nd diagonal

```
for (i=0; i<3; i++)
for (j=0; j<3; j++)
if ((i+j)==2)
  sum-diagonal = square
```

Use the backside, if needed

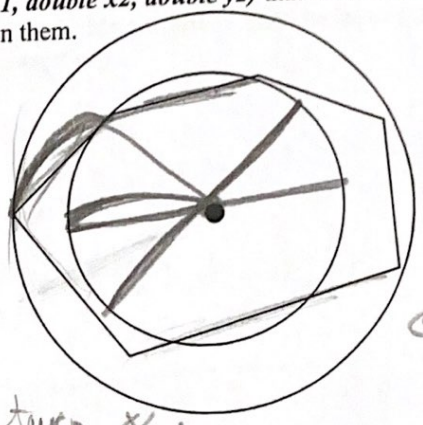
OOP.MT.[]031J.L035

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.



```
public static double thickness(double[][] vertex){
    int a = double[][]vertex;?
    int first row = x ;
    int second row = y ;
    for(x=0; x<vertex.length;x++)  → return x/r ;
    for(y=0; y<vertex.length;y++)
                                   → return y/r ;
. System.out.println.
```

```
public static double dist(double x1, double y1, double x2, double y2)
```

* we need to take into consideration 2 example : m and n.

For these elements → (sample elements m and n)

We can construct the needed distance

The boundary thickness is equal to the difference of the

sample points taken from the vertexes.

1

Use the backside, if needed

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 print (int [] array){
for(int i=0; i< array.length; i++)
        System.out.print(array[i] + " ");
    System.out.println();
}
public static void swap(int [] array; int i; int j){
array[i] += array[j];
array[j] = array[i] - array[j];
array[i] -= array[j];
}
```

2.
```
public static void swapRows (int array[][]; int rowA[]; int rowB[]) {
    int tempRow[] = array[rowA];
    array[rowA] = array[rowB];
    array[rowB] = int tempRowA;
}
```

3.
```
public static void swapColumns (int array[][]; int columnA[]; int columnB[])
{
    int tempColumn[] = array[columnA]; // row
    array[columnA] = array[columnB]; // row
    array[columnB] = int tempColumnA;
}
```

Conclusion: These are matrices with i'th and j'th elements the swaping is some kind of transpose but in this case we had specific cases what we needed to swap

1. rows } specified
2. columns
3. all matrix elements

Use the backside, if needed

Problem 4 of 4 2

OOP.MT.170317.0035