Section, Name and ID#

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

CS 120 Introduction to Object-Oriented Programming

MIDTERM EXAM

4

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 (int n) { float sum = 1 ; float product = 1 ;

for ( int i = 1 , i ≤ n , i++ ) {

    sum = kahan (sum ; 1/ (product * i )

    product = product * i ;

 }

   return sum ;

}
```

1

see AB

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 mean;
    double standard_devitation;

  double[] mean* = new double[2]
  for(int i = 0 ; i ≤ double length; i++){
      double sum += data[i]
   mean = sum/data length;
  }
   double [i] = mean;
    For(int j = 0 ; j ≤ double length; j++){
      int summention; sum mention += Math.Pow data[i]-mean){
      sum mention = summeation/data length;
   standard devitation = Math. sqrt(summention);
     mean[2] = standard_devitation
      return mean[2];
    }
```

0

AB

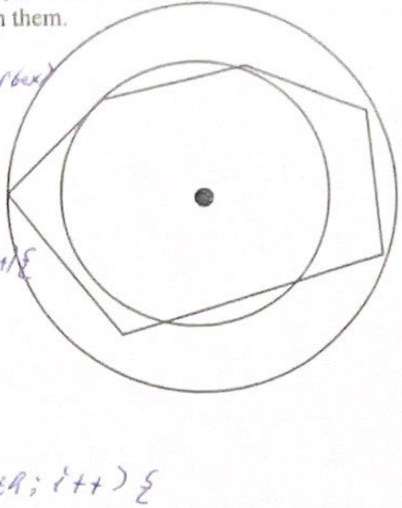
Use the backside, if needed

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)
rowLa].length = row[1]length;
1) for(col[i]=col[o];i<col[o].length, i++){
        Sum x += col[i]; };

    mean x = sumx / row[O].length;
 for(col[i]=col[o]; i<row[1].length; i++){
        sum y += col[i];};

    mean y = sum y / row[1].length;

2) for(i=0, i<row.length; i++){
     double dist (doubee vertex[i][i]; doublevertex[1][i],
                 double meanx, double mean y) }
 //from center to vertices
```

```
double max = 0,0 ; double min = 0,0;
    for(i= 0,i<row.length ;i++){
If (dist[i]>max)
    max = dist[i];}
if(dist[i] < min)
  for(i=0;i<row.length;i++){
  min = dist[i]; }
// ~~for~~ maximal distance
```

See NG

Use the backside, if needed

OOP.MT.17.03.17.LO73

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){

        x = int [row][col]a ;
        y = int[row ][col]b;

   ?  { x = x+y ;
        y = x-y ;
        x = x-y ;  }

2) public static void swapCol(int[][]a, int[][]b, int col) {
        x[] = int [ ][col ]a;
        y[] = int[][col]b;

     For ( i=0 ; i< row.length ; i++){
   ?  { x[ i ]= x[ i ]+ y[ i ] ;
        y[ i ]= x[ i ]- y[ i ] ;
        x[ i ]= x[ i ]- y[ i ],  }

3) public static void ~~Rowd~~ swapRow( int[][ ]a, int[][ ]b, int row)
        x[ ]= int[ ][row ]a ;
        y[ ] = int[ ][row ]b;
     For (i=0; i< col.length; i++ ) {
          x[ i ]= x[i]+y[i];
   ?      y[ i ]= x[i]-y[i];
          x[i ]= x[i]- y[i]; }
```

Use the backside, if needed

2

OOP.MT.170317.L073