Section, Name and ID#

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

8

**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 e (int n) {or fact [k] , arr num[k] , float num ,
int fact [0] = 1;
Cin >> int n;
for (k=0, k<=n , k++);{          // factorial of numbers from 0 to n
If (k>0) , fact[k] = 1* fact *k;   Else fact [k] = 1 ; }
for (k=0, k<=n, k++);            // division of 1 by every element of fact [k]
   num[k] = 1 / fact[k];
for (k=0, k<=n, k++);            // sum of all elements of num[k]
   final = Kahan (num[k]) ;
Cout << "e=" << final;
return 0, }
```

nice idea
wrong implementation

2

*Use the backside, if needed*



OOP.MT.iJo3iJ.M02

ection, 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 $\sigma$ 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 sum1=0;   arr a[]; arr b[] ; double c;
  double sum2=0;
  for (i=0, i < data.length , i++){
      sum1  +=  data[i] ;}  // sum of all elements in data[]
  mean value = sum1/(data.length +1) ; /*find the mean value by dividing
                                         the sum of all elements by the
                                         number of elements */
for (i=0, i < data.length , i++){
   a[i]= data[i] - mean value;
   b[i]  = a[i] * a[i] ;}  // square of all elements in a[]

for (i=0, i < data.length , i++)
      sum2 += b[i] ;  //sum of all elements in b[],
  c =  sum2 /(data.length +1) ;//divide sum2 by number of elements
  st dev =  c ^(1/2) ;   // square root of c

~~System.out.println(double [mean value ; st dev]) }~~
System.out.println(double [mean value ; st dev ]) }
```

return needed

3

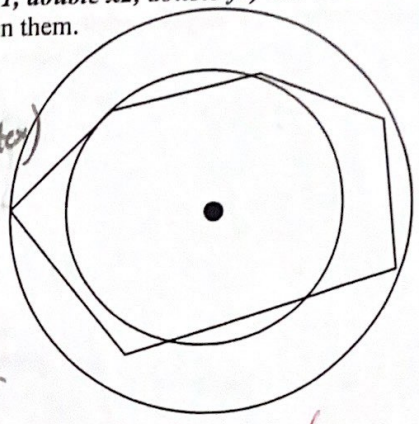
*Use the backside, if needed*

ection, 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)
row[0].length = row[1].length;
1) for (col[i]=col[0]; i < row[0].length, i++){
        sum x += col[i]; };
   mean x = sum x / row[0].length;
   for (col[i]=col[0], i < row[1].length, i++){
        sum y += col[i];}
   mean y = sum y / row[1].length;
```

Variables not declared

```
2) // find the distances from the center to the vertices.
   for ( i=0, i < row.length, i++) {
   ? = double dist ( double vertex[0][i], double vertex[1][i],
                     double mean x, double mean y) }

   // find the maximal distances;
   double max = 0.0; double min;
   If (dist[i] > max)                  for (i=0, i < row.length, i++){
       max = dist[i]; }
   If ( dist[i] < min)                 for (i=0, i < row.length, i++){
       min = dist[i]; }                // find the minimal distance

   thickness = max - min; // difference between maximal and
                                minimal distances
   System.out.println(double thickness); }
```

return needed

*Use the backside, if needed*

ection, 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 ;
}
```
0,6 ?

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]; }
```
// x[] are the elements x of
// the column col
// y[] are the elements y of
// the column col.

3)
```
Public static void swapRow (int[][]a ,int[][] b, int row) {
    x[] = int [row][]a ;
    y[] = int [row][]b,
    for (j=0 , j < col.[i] .length, j++){
        x [j]= x[j]+y[j];
        y[j] = x[j]- y[j];
        x[j] = x[j]- y[j]; }
```
// x[] are the elements x
// of the row row.
// y[] are the elements y
// of the row row.

bonus, in the back of the page →

1

*Use the backside, if needed*

Problem 4 of 4

OOP.MT.170317.M021

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

0