Section, Name and ID#:

17
2C

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

MIDTERM EXAM

15

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){
  float ans=0; float comp=0.0;
  for(int a=1; a<n+1; a++){
     int num=1;
     for(int c=1; c<=a; c++){
        num*=c;
     }
     ans=kahan(ans, 1./num, comp);
  }
  return ans;
}
```

3

Use the backside, if needed

Problem 1 of 4

OOP.MT.170317J.H93

Section, Name and ID#: [redacted]

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=0;
    double sum=0;
    int length=data.length;
    for(int ii=0;ii<length;ii++){
        sum+=data[ii];
    }
    mean=sum/length;
    double top=0;
    double ans=0;
    for(int ii=0;ii<length;ii++){
        top+=(data[ii]-mean)* →
                        →(data[ii]-mean);
    }
    ans=Math.sqrt(top/length);
    return (new double[]{mean,ans});

}
```

34

Use the backside, if needed

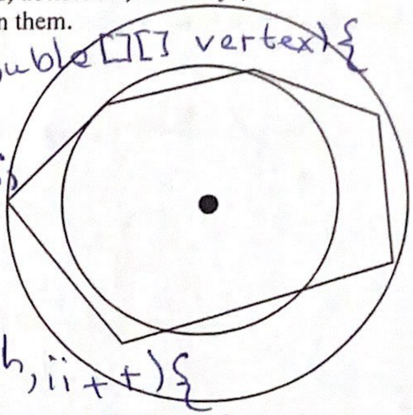
OOP.MT.17031J.H013

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 length=~~double~~vertex[0].length;
    double centerx=0;
    double center y=0;
    for(int ii=0;ii<length,ii++){
        Centerx+=vertex[0][ii];
        centery+=vertex[1][ii];
    }
    Centerx=centerx/length;
    centery=centery/Length;
    double min=dist(vertex[0][0];vertex[1][0];
                                        CenterX,centery);
    double max=min;
    for(int ii=1;ii<length;ii++){
        double localdist=dist(vertex[0][ii];vertex[1][ii];
                                        Centerx;centery);
        if(localdist> max){
            max=localdist;
        }
        if(localdist<min){
            min=localdist;
        }
    }
    return max-min;
}
```

Use the backside, if needed

4

Section, Name and ID#: [redacted]

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){
       int temp= a[row][col];
       ~~b[Row~~
       a[row][col]= b[row][col];
       b[row][col]= temp;
   }
 2. public static void swapCol(int[][] a,int[][] b,int col)
                                                      {
        int length= a[illegible]length;
        for(int ii=0;ii<length;ii++){
            int temp=a[ii][col];
            a[ii][col]= b[ii][col];
            b[ii][col] = ~~a[ii][col]~~ temp;
        }
    }
  3. public static void swapRow(int[][] a,int[][] b,int row){
        int length =a[0].length;
        for(int ii=0;ii<length;ii++){
            int temp= a[row][ii];
            a[row][ii] = b[row][ii];
            b[row][ii] = ~~a[row~~ temp;
        }
   }
```

4

Use the backside, if needed