**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***.

```
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;
}
```

*Use the backside, if needed*



OOP.MT.170317.L031

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 $\sigma$ of $n$ numbers $a_i$ is computed as:

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

```
public & statidc double [] mean(double [] data){
    double mean = 0;
    double sum = 0;
    int length = data length.;
for (int ii = 0; ii < length.; ii++){
    sum += data.[i i].
}
    mean.= sum / length
    double top = 0;
    double ans = 0;
for (int ii=0 ; ii<length; ii++){
    top+ = (data [ii]- mean)**(data[i i]mean);
} ans= Math Sqrt.(top/length);
    return (new. double [] { mean, ans});
}
```

*Use the backside, if needed*



OOP.MT.170317.L031

**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 [0] length.
    double centerX = 0.
    double centerY = 0.
for (int ii = 0; ii < length; i++) {
    centerX += vertex [0] [ii];
    centerY += vertex [1] [ii];
}
    center.x = centerX / length;
    centery = centerY / length;
double min = dist (vertex [0] [0]
                   ~~Counte~~ centerX, centerY.
    double max = min.
for (int ii = 1. ii < length. ii++.) {
    double mean. dist = vertex [0] [ii]
    if (local dist. > max.) {
        max = lead. dist;
    } if local dist < min {
        mean. = local. dist
}
    return max - min;
```

*Use the backside, if needed*



OOP. MT. 1703 1. 1031