

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:

$$e = \sum_{k=0}^n \frac{1}{k!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1*2} + \frac{1}{1*2*3} + \dots$$

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) {
    for(int i=0; i<n; i++) {
        int k, m;
        k = i*(i+1);
        m = i*(i+1)*(i+2);
        sum = sum + kahan(1/k, 1/m) - 12/m;
    }
    return sum;
}

int main() {
    float b = float e(sum);
    cout << b+2;
    return 0;
}
```

Use the backside, if needed

Problem 1 of 4

00P.M.T.170317.M017

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:

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

```
public static double[] mean(double[] data) {  
    double[] result = new double[2]; int sum = 0;  
    for (int i = 0; i < data.length; i++) {  
        int sum = 0 sum = sum + data[i];  
    }  
    double mean = sum / data.length;  
    result[0] = mean;  
    for (int k = 0; k < data.length - 1; k++) {  
        sum1 = sum1 + data[k];  
    }  
    alpha = sum1 / data.length alpha = Math.pow(sum1 - mean, 2) / data.length;  
    double deviation = Math.sqrt(alpha);  
    result[1] = deviation;  
    return result;  
}
```

3

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;

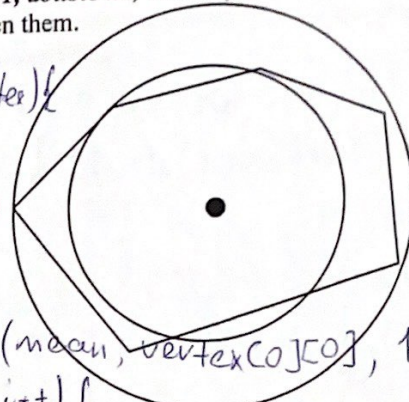
2. Returns the difference between the maximal and minimal distances from the center to the vertices.

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.

```

1. public static double thickness(double[][] vertex) {
    for (int i = 0; i < vertex.length; i++) {
        double mean = (vertex[0][i] + vertex[0][i]) /
            vertex.length;
    }
    2. int max; int min; max = 0; min = dist(mean, vertex[0][0], 1, 0)
    for (int i = 0; i < vertex[0].length; i++) {
        if (dist(x0, y0, vertex[0][i], 1, i) > max) {
            max = dist(x0, y0, vertex[0][i], 1, i);
        }
        if (dist(x0, y0, vertex[0][i], 1, i) < min) {
            min = dist(x0, y0, vertex[0][i], 1, i);
        }
    }
    return max - min;
}

```



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 a bonus, if `swapRow()` performs faster than `swapCol()`.

```

1. public static void swap(int[][] a, int[][] b, int row, int col) {
    int ab;
    ab = a[row][col];
    a[row][col] = b[row][col];
    b[row][col] = ab;
}

2. public static void swapCol(int[][] a, int[][] b, int col) {
    for (int i = 0; i < a.length; i++) {
        int k;
        k = a[i][col];
        a[i][col] = b[i][col];
        b[i][col] = k;
    }
}

3. public static void swapRow(int[][] a, int[][] b, int row) {
    for (int i = 0; i < a[0].length; i++) {
        int f;
        f = a[row][i];
        a[row][i] = b[row][i];
        b[row][i] = f;
    }
}

```

3