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

10

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 pi(int n)*** that computes the value π by the following formula:

$$\pi = 16\sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)5^{2k+1}} - 4\sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)239^{2k+1}} = \left(\frac{16}{1*5} - \frac{4}{1*239}\right) - \left(\frac{16}{3*5^3} - \frac{4}{3*239^3}\right) + \left(\frac{16}{5*5^5} - \frac{4}{5*239^5}\right) - \cdots$$

The initial value of ***float compensation*** is ***0.0***.

```
#include <cmath.lib>>
float pi(int n) {
  float comp=0.0; pi=0;
  for (int k=0; k<=n; k++) {
    pi+=kahan((pow(-1,k))/((2k+1)*pow(5,2k+1)))*16),
      -4((pow(-1,k))/((2k+1)*pow(239,2k+1))),
      &comp));}
  return pi;
}
```

3

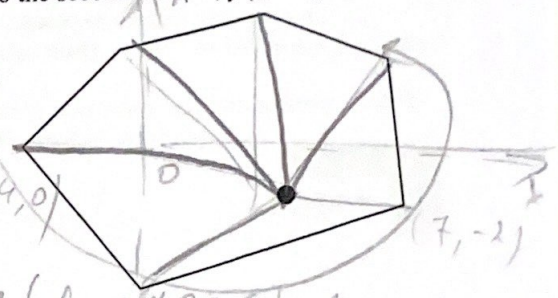
Use the backside, if needed

OOP.MT.170317.M026

Section, Name and ID#:_____

Problem 3: Write a Java function ***public static boolean isInside(double[][] vertex, double x, double y)*** that takes as its argument a ***2-by-n*** array of a convex polygon's vertex coordinates ***double[][] vertex*** – the ***x*** coordinates in the first row and ***y*** coordinates in the second row, and ***double x*** and ***double y*** coordinates of a point. It checks, if the point is inside the polygon.

Assume and use a method ***boolean toLeft(double x1, double y1, double x2, double y2, double x0, double y0)*** that takes as its arguments coordinates of three points and returns ***true***, if the third point ***(x0, y0)*** is in the left-hand side, when moving from the first point ***(x1, y1)*** to the second one ***(x2, y2)***; and ***false***, if it is in the right-hand side.



Assume

```
public static boolean isInside(double[][] vertex, ...) {
    boolean result = true;
    for(int i=0; i<vertex[0].length; i++) {
    if (vertex[...] toLeft
        for(int j=0; j<vertex[0].length; j++) {
        if (boolean toLeft(vertex[0][j], vertex[1][j],
        vertex[0][j+1], vertex[1][j+1], x, y)
          else return false; }
          return result;
    }
```

vertex[0][0], vertex[1][0]

[0][0] [1,1]

3

OOP.MT.170315.M026

Section, Name and ID#: _____

Problem 4: Write a Java method ***public static void magicOdd(int[][] square)*** that creates a magic square of an *odd* size using the following algorithm:

1. The number *1* goes in the middle of the top row;
2. All numbers are then placed one ***column to the right*** and ***one row up*** from the previous number;
3. Whenever the next number placement is above the top row, stay in the same column and place the number in the bottom row (note the place of *2* instead of the shaded location);
4. Whenever the next number placement is outside of the rightmost column, stay in the same row and place the number in the leftmost column (note the place of *3* instead of the shaded location);
5. When encountering an already filled-in square, place the next number directly below the previous number;
6. When the next number position is outside both a row and a column, place the number directly beneath the previous number (note the place of *7* instead of the shaded location).

	9	2	7
8	1	6	8
3	5	7	3
4	9	2	

```
public static void magicOdd(int[][] sq){
square[0][square.length/2]=1;
for(int i=0; i<square.length; ++r){
   for(int j=0; j<square.length; j++){
   i
```

Assume there is magic square function
Assume there is swapRows function
create magic square,
swap the top and bottom rows,
profit!
return sqr.

1

Use the backside, if needed

Problem 4 of 4

OOP.MT.070317.M026