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 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***.

```
float pi(int n)
{
    float firstResult = 0;
    for(int i=0; i<n; i++) {
        firstResult += pow(-1,i)/((2i+1)*pow(5,2i+1));
    }
    float secondResult = 0;
    for(int j=0; j<n; j++) {
        secondResult += pow(-1,n)/((2i+1)*pow(239,2i+1));
    }
    float pi = 0;
    pi = 16*firstResult - 4*secondResult;
    return ?
}
```

2

Use the backside, if needed

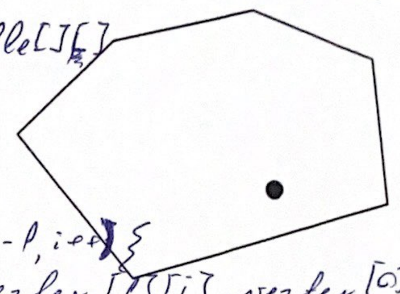
Problem 1 of 4

OOP.MT.170317.L081

Section, Name and ID# [redacted]

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.



```
public static boolean isInside(double[][]
vertex, double x, double y) {
[redacted]            int checking = 0
for ( i[redacted], [redacted] checking < vertex[0].length - 1; i++) {
!(boolean  toLeft( vertex[0][1], vertex [1][i], vertex [0][i+1],
   vertex[1][i+1] , x, y)
}
   return ( [redacted] checking = vertex[0].length);
```

0

Use the backside, if needed

OOP.MT:110317.L081