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:

$$\pi = 2\sum_{k=0}^{n} \frac{(2k-1)!!}{(2k)!!(2k+1)} = \frac{2}{1*1} + \frac{1}{2}*\frac{2}{3} + \frac{1*3}{2*4}*\frac{2}{5} + \frac{1*3*5}{2*4*6}*\frac{2}{7} + \cdots$$

Recall that **n!!** is the product of odd numbers from *1* to ***n***, if ***n*** is odd; and is the product of even numbers from *2* to ***n***, if ***n*** is even. The double factorial of non-positive numbers equals to *1* by definition.

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

float pi(int n)

Use the backside, if needed

OOP. MT. 170317. M069

```
using namespace std;


int number_of_lines=0;
void numberoflines();
   int main() {
    string line;
    ifstream myfile("textexample.txt");
        if (myfile.is_open()) {
          while (!myfile.eof()){
           getline(myfile,line);
            cout<< line <<endl;
          number_of_lines++; }
          myfile.close(); }
          number oflines(); }

        void numberoflines(){
            number_of_lines--;
          cout<<"number of lines in text file:"<< number_of_lines<<endl;
          }
```

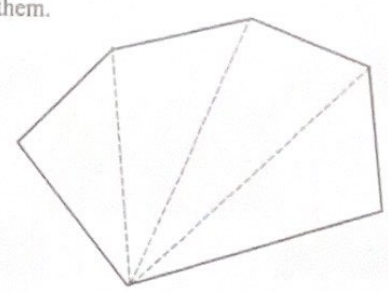
2

Section, Name and ID#:_____

Problem 3: Write a Java function ***public static double area(double[][] vertex)*** 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. It returns polygon's area as follows:

1. Divides the polygon into triangles by connecting the ***first*** vertex with the n^{th} and $(n+1)^{st}$ vertices;
2. Adds the areas of the constructed triangles using the formula $area = \sqrt{p(p-a)(p-b)(p-c)}$, where ***a***, ***b*** and ***c*** are the sides and $p = (a + b + c) / 2$.

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 class WidthLengthAreaMethods {
public static void main (String[]args) {
Scanner keyboard = new Scanner (System.in)
double length;
double width;
double area;
length = getLength();
width = getWidth();
area = getArea (double length, double width);
displayData (length, width, area);
public static double getLength() {
System.out.println("Enter length. ");
length = Keyboard.nextDouble();
System.out.println(" The length is" + length); }
public static double getWidth() {
System.out.println("Enter width. ");
width = Keyboard.nextDouble();
System.out.println(" The width is " + width); }
public static double getArea()
{ double length;
double width
double area = length * width
System.out.println(" the area is: " + area); }
public static void displayData(double length, double width, double area ) {
System.out.println(" the length is \t " + length);
System.out.println("The width is \t " + width);
System.out.println(" The area is \t " + area); } }
```

008.MT.170317.M069

Use the backside, if needed

2

Section, Name and ID#:_____

Problem 4: Write a Java method ***public static void magic4N(int[][] square)*** that creates a magic square of a ***4N-by-4N*** size using the following algorithm:

1. Creates an array of the same size as ***int[][] square*** and fills it forward with successive integers assigning *1* to the top-left element;
2. Creates anther array of the same size as ***int[][] square*** and fills it backward with successive integers assigning *1* to the bottom-right element;
3. Divides the original ***int[][] square*** into 16 blocks of the same size – 4 blocks per row and column. In the on-diagonal (shaded) blocks copies the elements from the first array, and in the off-diagonal blocks copies the elements from second array.

1	2					7	8
9	10					15	16
		19	20	21	22		
		27	28	29	30		
		35	36	37	38		
		43	44	45	46		
49	50					55	56
57	58					63	64

		62	61	60	59		
		54	53	52	51		
48	47					42	41
40	39					34	33
32	31					26	25
24	23					18	17
		14	13	12	11		
		6	5	4	3		

http://stackoverflow.com/questions/2592102/2d-array-diagonally-filling/25922901

```
public static void main (String[] args)
  // create an M x N array
  int rows = 16;
  int columns = 4;
  int [][] testData = new int[rows][columns];
    // iteratively add numbers
  int counter = 0;
  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
      testData [i][j] = ++counter;
    }
  }
  // print our test array
  printArray(testData);
  System.out.println("");
// print
```

Use the backside, if needed

Problem 4 of 4

OOP.MT.140317.M069

one way

```
int [][] convertToDiagonal(int [][] input) {
  int [][] output = new int[input.length][input.length];
  int i=0, j=0;  i count rows, j count columns

  int n = input.length
  for (int slice = 0, slice < 2*n-1; slice++){
  for (int k = 2, k ≤ slice - 2; ++k){ //store slice value in current row
    output[i][j++]=input[k][slice-k];
  }
  //if we reached end of row, reset column counter,
  if (j == n) {
    j = 0;
    i++;
  }
  }
  return output;
```

```
for (int k=0; k<dim; k++){ // loop extract top half of diagonals
  for (int j = 0; j ≤ k ; j++){
    int i = k-j;
    System.out.print(array[i][j]+ " ");
  }
  System.out.println();
  }
                                  // iterates on bottom half of diagonals
for (int k = dim-2; k>0; k--) {
  for (int j =0; j ≤ k; j++) {
    int i = k-j;
    System.out.print(array[dim-j-1][dim-i-1] + " ");
  }
  System.out.println();
}
```

0