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***.

Use the backside, if needed

OOP.M1.170317.1048

```
    for(int row=0; row<matrix.length;row++){
      for(int col=row+1; col<matrix[row].length;col++){
        matrix[row][col] = matrix[row][col] ^ matrix[col][row];
        matrix[col][row] = matrix[row][col] ^ matrix[col][row];
        matrix[row][col] = matrix[row][col] ^ matrix[col][row];
      }
    }
  }
  public static void main(String[] args){
    Scanner input = new Scanner(System.in);
    int size = input.nextInt();
    int[][] matrix = new int[size][size];
    for(int row=0; row<matrix.length;row++){
      for(int col=0; col<matrix[row].length;col++){
        matrix[row][col] = input.nextInt();
      }
    }
    transpose(matrix);
    for(int row=0;row<matrix.length;row++){
      for(int col=0;col<matrix[row].length;col++){
        System.out.print(matrix[row][col] + "\t");
      }
      System.out.println();
    }
  }
```

2

2) Write a Java program that computes the multiplication of 2 matrices.

```
public static int scalar(int[] a, int[] b){
  int result=0;
  for(int i=0;i<a.length;i++){
    result+= a[i] * b[i];
  }
  return result;
}
public static int[][] multiplication(int[][] mat1, int[][] mat2){
  int[][] result = new int[mat1.length][mat1.length];
  transpose(mat2);
  for(int row=0; row<mat1.length;row++){
    for(int col=0; col<mat2.length; col++){
      result[row][col] = scalar(mat1[row], mat2[col]);
    }
  }
  return result;
}
```

$2

Section, Name and ID#: _____

Problem 2: Write a Java method ***public static double[] lin(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 slope of the linear regression and the second element being the intercept. The linear regression approximates the data points by the linear formula

$$y = k\,x + b,$$

where the slope k and the intercept b are computed as

$$k = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\overline{x^2} - \bar{x}^2}, b = \bar{y} - k\,\bar{x}$$

Here $\bar{x}$ is the mean of the x coordinates, $\bar{y}$ is the mean of the y coordinates, $\overline{x^2}$ is the mean of the squares of the x coordinates, and $\overline{xy}$ is the mean of the products of the x and y coordinates. Use the element indices of the array ***double[] data*** as x coordinates and the element values as y coordinates. You may assume and use the method double ***mean(double[] a)***.

Use the backside, if needed

OOP.MT.17031.H048

3) Write a Java function that computes the reverse of an array.

```
public static void reverse(int[] array) {
   for(int i=0; i<array.length/2; i++) {
      array[i] = array[i] ^ array[array.length-1-i];
      array[array.length-1-i] = array[i] ^ array[array.length-1-i];
      array[i] = array[i] ^ array[array.length-1-i];
   }
}
public static void main(String[] args) {
   int[] array = {39, 29, 19, 9};
   reverse(array);
   for(int i=0; i<array.length; i++) {
      System.out.println(array[i]);
   }
}
```

1

4) Write a Java program that fills 2D arrays with successive integers in descending order.

```
public static void main(String[] args) {
   int[][] backward = new int[4][4];
   for(int row=0; row<backward.length; row++){
      for(int col=0; col<backward[row].length; col++) {
         backward[row][col] = backward[row].length * (backward.length - row) - col - 1;
      }
   }
   for(int row=0; row<backward.length; row++){
      for(int col=0; col<backward[row].length; col++) {
         System.out.print(backward[row][col]);
      }
      System.out.println();
   }
```

1

5) Write a C++ program that sorts the array elements, from smallest to biggest.

```
int main() {
   int val[8]={100, 30, 7, 14, 25, 60, 37, 99};
   for(int i=0; i<8; i++){
      for(int j=i+1; j<8; j++){
         if(val[i] > val[j]) {
            int temp= val[i];
            val[i] = val[j];
            val[j]=temp;
         }
      }
   }
   for(int i=0; i<8; i++) {
      cout << val[i] << endl;
   }
   return 0;
}
```

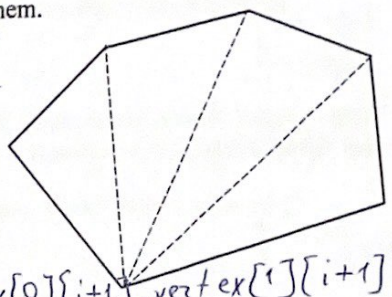
2

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 static double area (double[][] vertex) {
    double area = 0;
    for (int i=0; i<vertex.length-1; i++) {
        a = dist (vertex[0][0], vertex[1][0],
                  vertex[0][i], vertex[1][i]);
        b = dist (vertex[0][0], vertex[1][0], vertex[0][i+1], vertex[1][i+1]);
        c = dist (vertex[0][i], vertex[1][i], vertex[0][i+1], vertex[1][i+1]);
        double p = (a+b+c) / 2;
        area += sqrt (p*(p-a)*(p-b)*(p-c));
    }
    return area;
}
```

3

=>

Use the backside, if needed

OOP.MT.17031T.H048

6) Write a C++ function that takes as its argument a string and checks if it is palyndrome or not.

```
bool isPal (string arg) {
  int left=0, right= arg.length - 1;
  while (left < right) {
     if (arg[left] == arg[right]) {
        return true;
     }
     left++;
     right--;
  }
  return false;
}
int main()  {

   return 0;
}
```

1

7) Write a Java program that enterchanges the rows of 2 matrices. For example, the first row of the first matrix becomes the second row of the second matrix.

```
public static void main (String[] args) {
  int[][] mat1 = {{1,3,5},{9,4,6}};
  int[][] mat2 = {{10,13,15},{20,25,30}};

  mat1[0] = mat2[1];
}
```

Swap?

1

8) Write a Java program that fills 2D arrays with successive integers in ascending order.

```
int[][] forward = new int[6][6];
for(int row=0; row< forward.length; row++){
   for(int col=0; col < forward[row].length; col++){
      forward[row][col] = (row * forward.length) + col;
   }
}
for(int row=0; row < forward.length; row++){
   for(int col=0; col < forward[row].length; col++){
      System.out.print(foward[row][col]);
   }
   System.out.println();
}
```

1