**SUTD 50.001 Introduction to Information Systems and Programming**

**Problem set 2 (Part A)**

Note: Please submit your answer to eDimension

**Questions (Week 3)**

1. [10 points] Title: Comparable interface

Given the following Octagon class:

**public class** Octagon {  
 **private double side**;  
 **public** Octagon(**double** side){  
 **this**.**side** = side;  
 }  
 **public double** getSide() {  
 **return side**;  
 }  
}

Assume that all eight sides of the Octagon are of equal size.

Modify Octagon class to implement the Comparable<Octagon> interface to allow sorting of Octagon objects based on their perimeters.

Test code:

**public class** Test {  
 **public static void** main(String[] args) {  
 ArrayList<Octagon> l = **new** ArrayList<Octagon>();  
 l.add(**new** Octagon(2));  
 l.add(**new** Octagon(3));  
 l.add(**new** Octagon(1));  
 Collections.*sort*(l);  
 **for** (Octagon o:l)  
 System.***out***.println(o.getSide());  
 }  
}

Results:

1.0

2.0

3.0

2. [10 points] Title: Regular Expression

Given the following code, provide the value for “str” to use regular expression to generate output:

true

true

false

false

true

**Code:**

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** TestRegex {

**public** **static** **void** main(String[]args) {

// TODO: initialize str appropriately

String str;

Pattern p1 = Pattern.*compile*(str);

Matcher m = p1.matcher("aaabcc");

**boolean** b = m.matches();

System.*out*.println(b);

Matcher m2 = p1.matcher("accc");

**boolean** b2 = m2.matches();

System.*out*.println(b2);

Matcher m3 = p1.matcher("aaaabccccc");

**boolean** b3 = m3.matches();

System.*out*.println(b3);

Matcher m4 = p1.matcher("aabbccccc");

**boolean** b4 = m4.matches();

System.*out*.println(b4);

Matcher m5 = p1.matcher("aab");

**boolean** b5 = m5.matches();

System.*out*.println(b5);

}

}

3. [30 points] Title: Java Array

We represent matrices by two-dimensional arrays.

1. [10 points] Design a method **matMpy** that performs multiplication of the two matrices **A** and **B**, and returns their matrix product **AB**. The method returns **null** if the dimensions of the matrices are not compatible for matrix multiplication.
2. [20 points] Design a method **mergeSortedArray**:

public static void mergeSortedArray(int [] A, int m, int [] B, int n)

Given two sorted integer 1-d arrays **A** and **B** (in ascending order, i.e., from the smallest to the largest), the method merges **B** into **A** to form a single sorted array in ascending order. You can assume that **A** has enough space to hold the additional elements from **B**. Array **A** has **m** elements and array **B** has **n** elements, and **m** and **n** are passed as input arguments. The array **A** has size **m+n**, enough to hold the elements from **B**.

Full credits will be given only if: (i) you perform this in place, i.e., no additional array / storage is created to hold the elements; (ii) no additional sorting, since arrays **A** and **B** are sorted already. That is, your solution should be memory and computation efficient.

**Code:**

**public** **class** TestMatrix {

**public** **static** **void** main(String[] args) {

**double** [][] a = {

{1,2,3},

{4,5,6}

};

**double** [][] b = {

{1,2},

{3,4},

{5,6}

};

**double** [][] c = *matMpy* (a,b);

**int** [] f = **new** **int** [5];

f[0] = 2;

f[1] = 5;

**int** [] g = {1, 3, 7};

// f has 2 elements and g has 3 elements

*mergeSortedArray*(f, 2, g, 3);

}

**public** **static** **double** [][] matMpy ( **double** [][] a, **double** [][] b) {

// to be implemented

}

**public** **static** **void** mergeSortedArray(**int**[] a, **int** m, **int**[] b, **int** n){

// to be implemented

}

}

**Output:**

c = {

{22.0, 28.0},

{49.0, 64.0}}

f = {1, 2, 3, 5, 7}

4. [10 points] Title: Boolean Satisfiability

Is the following propositional formula of Boolean variables A, B, C, D satisfiable? Circle your answer, and give detail explanation and derivation. No credit will be given if explanation / derivation is not provided.

(¬A ∨ B) ∧ (A) ∧ (B -> C) ∧ (C->D) ∧ (¬D)

**Your Solution:**

a. Satisfiable

b. Not satisfiable

**Explanation / Derivation:**