Montgomery College CMSC 203 Assignment 5 Design

1) Write the pseudo code for the methods of *TwoDimRaggedArrayUtility* and *HolidayBonus* class based on the Assignment 5 Description given to you. Refer to the <u>Pseudocode Guideline</u> on how to write Pseudocode.

Two Dim Ragged Array Utility	HolidayBonus
// Import necessary libraries	// Import necessary libraries
Import static methods from the	Import static methods from the
org.junit.Assert library	org.junit.Assert library
Import File, FileNotFoundException,	Import Before, After, and Test annotations
PrintWriter, Scanner classes	from the org.junit package
Import Before, After, and Test annotations	
from the org.junit package	// Define a class named HolidayBonusTest Class HolidayBonusTest:
// Define a class named	Class Heliaay zollas resti
TwoDimRaggedArrayUtilitySTUDENT_Test	// Declare data sets for testing
Class	Declare dataSet1 as 2D array of doubles
TwoDimRaggedArrayUtilitySTUDENT Test:	Declare dataSet2 as 2D array of doubles
	Declare dataSet3 as 2D array of doubles
// Declare variables	Declare dataSet4 as 2D array of doubles
Declare dataSet as 2D array of doubles	·
Declare inputFile, outputFile as files	// Method to set up test environment
	before each test
// Method to set up test environment	Before setUp():
before each test	// No setup required for this test
Before setUp():	
Initialize outputFile as a new file named	// Method to clean up after each test
"TestOut.txt"	After tearDown():
	// No cleanup required for this test
// Method to clean up after each test	
After tearDown():	// Test calculateHolidayBonus method
Set dataSet to null	with different parameters
Set inputFile and outputFile to null	Test testCalculateHolidayBonusA():
	Try:
// Test getTotal method	Call calculateHolidayBonus with
Test getTotal():	dataSet1, high of 5000, low of 1000, and
Assert that the total of all elements in	other as 2000
dataSet equals 3226.20	Assert the results are correct for each
// Tank and A arms and A	data set
// Test getAverage method	Catch any exceptions and fail the test if
Test getAverage():	encountered

Assert that the average of all elements in dataSet equals 201.6375

// Test getRowTotal method

Test getRowTotal():

Assert the total of each row in dataSet is correct

// Test getColumnTotal method

Test getColumnTotal():

Assert the total of each column in dataSet is correct

// Test getHighestInRow method

Test getHighestInRow():

Assert the highest number in each row in dataSet is correct

// Test getHighestInRowIndex method

Test getHighestInRowIndex():

Assert the index of the highest number in each row in dataSet is correct

// Test getLowestInRowIndex method

Test getLowestInRowIndex():

Assert the index of the lowest number in each row in dataSet is correct

// Test getLowestInRow method

Test getLowestInRow():

Assert the lowest number in each row in dataSet is correct

// Test getHighestInColumn method

Test getHighestInColumn():

Assert the highest number in each column in dataSet is correct

// Test getHighestInColumnIndex method

Test getHighestInColumnIndex():

Assert the index of the highest number in each column in dataSet is correct

// Test getLowestInColumn method

Test getLowestInColumn():

Assert the lowest number in each column in dataSet is correct

// Test calculateHolidayBonus method with different parameters

Test testCalculateHolidayBonusB():

Try:

Call calculateHolidayBonus with dataSet1, high of 1000, low of 250, and other as 500

Assert the results are correct for each data set

Catch any exceptions and fail the test if encountered

// Test calculateTotalHolidayBonus

method with different parameters

Test testCalculateTotalHolidayBonusA():

Assert the total holiday bonus calculated for each data set is correct

Call calculateTotalHolidayBonus with dataSet1, high of 5000, low of 1000, and other as 2000

Assert the results are correct for each data set

// Test calculateTotalHolidayBonus

method with different parameters

Test testCalculateTotalHolidayBonusB():

Assert the total holiday bonus calculated for each data set is correct

Call calculateTotalHolidayBonus with dataSet1, high of 1000, low of 250, and other as 500

Assert the results are correct for each data set

// Test getLowestInColumnIndex method

Test getLowestInColumnIndex():

Assert the index of the lowest number in each column in dataSet is correct

// Test getHighestInArray method

Test getHighestInArray():

Assert the highest number in dataSet is correct

// Test getLowestInArray method

Test getLowestInArray():

Assert the lowest number in dataSet is correct

// Test writeToFile method

Test writeToFile():

Create an empty array

Try to write dataSet to outputFile

Catch any exceptions and fail the test if encountered

Read from outputFile and assert the read data matches dataSet

// Test readFile method

Test readFile():

Create an empty array

Try to write some numbers to inputFile

Catch any exceptions and fail the test if encountered

Read from inputFile and assert the read data is in a ragged 2D array format

2) Complete the following test table. At this point you only need to complete the **Input** and **Expected Output** columns. Later when the implementation is complete, you will complete the **Actual Input** and **Actual Output** columns and compare them to see if the tests passed or not.

Make sure your tests cover all the possible scenarios.

Test Case #	Input	Actual Input	Expected Output	Actual Output	Did the test pass?
1	((-50.0, 100.0), (75.0, 125.0))		50		

2	((100.0, 200.0,	400.00	
	-300.0),		
	(400.0))		
3	Empty array	0	

Pseudocode Guideline

Pseudocode is code written for human understanding not a compiler. You can think of pseudocode as "English code," code that can be understood by anyone (not just a computer scientist). Pseudocode is not language specific, which means that given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

Pseudocode will be important to your future in Computer Science. Typically pseudocode is used to write a high-level outline of an algorithm.

As you may already know, an algorithm is a series of steps that a program takes to complete a specific task. The algorithms can get very complicated without a detailed plan, so writing pseudocode before actually coding will be very beneficial.

How to Write Pseudocode

There are no concrete rules that dictate how to write pseudocode, however, there are commonly accepted standards. A reader should be able to follow the pseudocode and hand-simulate (run through the code using paper and pencil) what is going to happen at each step. After writing pseudocode, you should be able to easily convert your pseudocode into any programming language you like.

We use indentation to delineate blocks of code, so it is clear which lines are inside of which method (function), loop, etc. Indentation is crucial to writing pseudocode. Java may not care if you don't indent inside your **if** statements, but a human reader would be completely lost without indentation cues.

Remember: Human comprehension is the whole point of pseudocode. So, what does pseudocode look like?

```
int n,k, f1, f2, sum;
Declare an integer variable called n
                                   if ( n < 2 )
Declare an integer variable sum.
                                      sum = n;
Declare an integer variable f1
                                   else
Declare an integer variable f2
                                   {
If n is less than 2
                                      sum=0;
       sum =n
                                      f1 = f2 = 1;
else
                                      for(k=2; k<n; k++)</pre>
       set sum to 0
       set f1 and f2 to 1
                                        sum = f1 + f2;
       repeat n times
                                        f2 = f1;
               sum = f1 + f2
                                       f1 = sum;
               f2 = f1
               f1 = sum
                                   System.out.println("Fibonacci of number " + n + " is "+ sum);
       end loop
print sum
```

Finding the Fibonacci numbers till n:

Remember that pseudocode is not language specific so we are not looking for "almost Java" code, but instead, we are looking for a strong understanding of the algorithm at hand.