**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 [**Pseudocode Guideline**](#PSGdline)on how to write Pseudocode.

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.**

Data set 1

A picture containing graphical user interface

Description automatically generated

Data Set 2

A picture containing graphical user interface

Description automatically generated

Data set 3

A picture containing graphical user interface

Description automatically generated

Data set 4

A picture containing table

Description automatically generated

District 3

Timeline

Description automatically generated

District 4

A picture containing graphical user interface

Description automatically generated

District 5

A picture containing timeline

Description automatically generated

**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?

|  |  |
| --- | --- |
|  |  |
|  | readFile :  Get a file called file as a parameter  Create a scanner called input that reads from file  Declare an int called rows and initialize to 0  While input has next line  Add one to rows  Read in a line from input  Close input  Declare a ragged array of doubles called data with int rows as the row length  Create a scanner called input2 that reads from file  For int j set to 0, j less than data row length, j plus one  Declare a string called line and set equal to a line read from input2  Declare a one-dimensional string array called separated and set equal to String line split by “ ”  Set data column size to separated length at row j  For int k set to 0, k less than data at row j column length, k plus one  Set data at row j column k to seperated at index k converted to double  Close input2  Return data  writeToFile:  Get a ragged array of doubles called data as a parameter  Create a PrintWriter called writer set to write to file  For int j set to 0, j less than data row length, j plus one  For int k set to 0, k less than data at row j column length, k plus one  Add data[j][k] converted to a string plus “ ” to file  Go to next line in file  Close writer  getTotal :  Get a ragged array of doubles called data as a parameter  Declare a double called total and initialize to 0  For int j set to 0, j less than data row length, j plus one  For int k set to 0, k less than data at row j column length, k plus one  Add data[j][k] to total  Return total  getAverage :  Get a ragged array of doubles called data as a parameter  Declare a double called count and initialize to 0  For int j set to 0, j less than data row length, j plus one  For int k set to 0, k less than data at row j column length, k plus one  Add one to count  Declare a double called average and set to getTotal of data divided by count  Return average  getRowTotal :  Get a ragged array of doubles called data and an int called row as a parameter  Declare a double called total and initialize to 0  For int j set to 0, j less than data at row row column length, j plus one  Add data at row row and column j to total  Return total  getColumnTotal :  Get a ragged array of doubles called data and an int called col as a parameter  Declare a double called total and initialize to 0  Declare an int called row and initialize to 0  While row is less than data row length  If data at row row column length greater than col  Add data at row row column col to total  Add one to row  Return total  getHighestInRow :  Get a ragged array of doubles called data and an int called row as a parameter  Declare a double called highest and initialize to data at row row column 0  For int j set to 1, j less than data at row row column length, j plus one  If data at row row column j is greater than highest  Set highest to data at row row column j  Return highest  getLowestInRow :  Get a ragged array of doubles called data and an int called row as a parameter  Declare a double called lowest and initialize to data at row row column 0  For int j set to 1, j less than data at row row column length, j plus one  If data at row row column j is less than lowest  Set lowest to data at row row column j  Return lowest  getHighestInColumn :  Get a ragged array of doubles called data and an int called col as a parameter  Declare a double called highest and initialize to 0  Declare an int called row and initialize to 0  While highest is equal to 0  If data at row row column length greater than col  Set highest to data at row row column col  Add one to row  While row is less than data row length  If data at row row column length greater than to col  If data at row row column col is greater than highest  Set highest to data at row row column col  Add one to row  Return highest  getLowestInColumn :  Get a ragged array of doubles called data and an int called col as a parameter  Declare a double called lowest and initialize to 0  Declare an int called row and initialize to 0  While lowest is equal to 0  If data at row row column length greater than col  Set lowest to data at row row column col  Add one to row  While row is less than data row length  If data at row row column length greater than col  If data at row row column col is less than lowest  Set lowest to data at row row column col  Add one to row  Return lowest  getHighestInArray :  Get a ragged array of doubles called data as a parameter  Declare a double called highest and initialize to data at row 0 column 0  For int j set to 0, j less than data row length, j plus one  For int k set to 0, k less than data at row j column length, k plus one  If data at row j column k is greater than highest  Set highest to data at row j column k  Return highest  getLowestInArray :  Get a ragged array of doubles called data as a parameter  Declare a double called lowest and initialize to data at row 0 column 0  For int j set to 0, j less than data row length, j plus one  For int k set to 0, k less than data at row j column length, k plus one  If data at row j column k is less than lowest  Return lowest  getHighestInColoumnIndex :  Get a ragged array of doubles called data and an int called col as a parameter  Declare an int called index and initialize to 0  Declare a double called highest and initialize to 0  Declare an int called row and initialize to 0  While highest is equal to 0  If data at row row column length greater than col  Set highest to data at row row column col  Set index to row  Add one to row  While row is less than data row length  If data at row row column length greater than to col  If data at row row column col is greater than highest  Set highest to data at row row column col  Set index to row  Add one to row  Return index  getLowestInColumnIndex :  Get a ragged array of doubles called data and an int called col as a parameter  Declare an int called index and initialize to 0  Declare a double called lowest and initialize to 0  Declare an int called row and initialize to 0  While lowest is equal to 0  If data at row row column length greater than col  Set lowest to data at row row column col  Set index to row  Add one to row  While row is less than data row length  If data at row row column length greater than col  If data at row row column col is less than lowest  Set lowest to data at row row column col  Set index to row  Add one to row  Return index  HolidayBonus.java  calculateHolidayBonus :  Get a ragged array of doubles called data, a double called high, a double called low and a double called other as parameters  Create a one-dimensional array of doubles called bonuses with a length of data row length  For int j set to 0, j less than data row length, j plus one  Set bonuses at index j to 0;  For int k set to 0, k less than data at row j column length, k plus one  If getLowestInColumnIndex given data and k is equal to j  Add low to bonuses at index j  Else if getHighesttInColumnIndex given data and k is equal to j  Add high to bonuses at index j  Else  Add other to bonuses at index j  Return bonuses  calculateTotalHolidayBonus :  Get a ragged array of doubles called data, a double called high, a double called low and a double called other as parameters  Create a one-dimensional array called bonuses and set equal to calculateHalidayBonus of data, high, low, and other  Declare a double called total and initialize to 0  For int j set to 0, j less than bonuses length, j plus one  Add bonuses at index j to total  Return total |

**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.**