**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.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Input** | **Actual Input** | **Expected Output** | **Actual Output** | **Did the test pass?** |
| 1 | Same as example | Same as example | Same as example | Same as example | Same as example |
| 2 | 5000.00, 2000.00 | 5000.00, 2000.00 | 7000.00 | 7000.00 | Yes |
| 3 | 4200.50, 8005.00 | 4200.50, 8005.00 | 12205.50 | 12205.50 | Yes |

**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?

|  |  |
| --- | --- |
| **Pseudocode** | **Real Code in Java** |
| Declare an integer variable called n  Declare an integer variable sum.  Declare an integer variable f1  Declare an integer variable f2  If n is less than 2  sum =n  else  set sum to 0  set f1 and f2 to 1  repeat n times  sum = f1 + f2  f2 = f1  f1 = sum  end loop  print sum | **int** n,k, f1, f2, sum;  **if** ( n < 2 )  sum =n;  **else**  {  sum=0;  f1 = f2 = 1;    **for**(k=2; k<n; k++)  {  sum = f1 + f2;  f2 = f1;  f1 = sum;  }  }  System.***out***.println("Fibonacci of number " + n + " is "+ 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.**

**Pseudocode:**

1. **Start.**
2. **Utility Class – TwoDimRaggedArrayUtility.**
   * 1. **Create a method called readFile, this method will pass in an array and return a 2D ragged array of doubles.**
     2. **Create a method called writeToFile, pass in a 2D array of doubles and a file, and write the ragged array into the file. Each row is on a separate line and each double is separated by a space.**
     3. **Create a method called getTotal, pass in a 2D ragged array of doubles and return the total of the elements inside the array.**
     4. **Create a method called getAverage, pass in a 2D ragged array of doubles and return average of elements inside the array.**
     5. **Create a method called getRowTotal, pass in a 2D ragged array of doubles and a row index and returns the total of that row. Row index 0 is the first row in the array.**
     6. **Create a method called getColumnTotal, pass in a 2D ragged array of doubles and a column index and returns the total of that column. Column index 0 is the first column in the array.**
     7. **Create a method called getHighestInRow, pass in a 2D ragged array of doubles and a row index and returns the largest element in that row. Row index 0 is the first row in the array.**
     8. **Create a method called getHighestInRowIndex, pass in a 2D ragged array of doubles and a row index and returns the largest element in that row. Row index 0 is the first row in the array.**
     9. **Create a method called getLowestInRow, a 2D ragged array of doubles and a row index and returns the smallest element in that row. Row index 0 is the first row in the array.**
     10. **Create a method called getLowestInRowIndex, a 2D ragged array of doubles and a row index and returns the index of the smallest element in that row. Row index 0 is the first row in the array.**
     11. **Create a method called getHighestInColumn, pass in a 2D ragged array of doubles and a column index and returns the largest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.**
     12. **Create a method called getHighestInColumnIndex, pass in a 2D ragged array of doubles and a column index and returns the index of the largest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.**
     13. **Create a method called getLowestInColumn, pass in a 2D ragged array of doubles and a column index and returns the smallest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.**
     14. **Create a method called getLowestInColumnIndex, pass in a 2D ragged array of doubles and a column index and returns the index of the smallest element in that column. Column index 0 is the first column in the array. If a row doesn’t contain that column, it is not an error, that row will not participate in this method.**
     15. **Create a method called getHighestInArray, pass in a 2D ragged array of doubles and returns the largest element in the array.**
     16. **Create a method called getLowestInArray, pass in a 2D ragged array of doubles and returns the smallest element in the array.**
3. **Utility Class – HolidayBonus.**
   * 1. **Create a method called calculateHolidayBonus, pass in a 2D ragged array of doubles. It will return an array of doubles which represents the holiday bonuses for each of the stores in the district. The first entry in the returned array [0] will represent the holiday bonus for the store at [0] in the two-dimensional ragged array of doubles. You will be using methods from the TwoDimRaggedArrayUtility when needed.**
     2. **Create a method called calculateTotalHolidayBonus, pass in a 2D ragged array of doubles, It will return a double which represents the total of all Holiday Bonuses for the District. You will be using methods from the TwoDimRaggedArrayUtility when needed.**
4. **GUI Application.**
5. **JUnit Test.**