**CSCI 2251 – Programming Assignment**

**Matrix Addition – Part 1 of 2**

This assignment has the following objectives:

1. implement concurrent processing, Java multi-threading.
2. split a larger problem into smaller problems.
3. assign each sub-problem to a separate thread.
4. gather the results from all threads.
5. minimize system resource usage, use shared memory to eliminate memory copy, multi-threading to effectively utilize processor cycles (especially for multi-core computers).

**Problem Description**

Given two integer matrices, A and B, you are asked to write a program to perform matrix addition (A + B).

Both matrices will have the same number of rows and columns.

You need to divide A and B into four equal (or close to equal) size submatrices (I will refer to them as A00, A01, A10, A11 and B00, B01, B10, B11)

If the original matrices have R rows and C columns, then each submatrix should have dimensions close to (R/2) x (C/2). In other words, each submatrix should be about one-quarter the size of the original matrices.

You need to create four Java threads. Each thread performs addition on one pair of the submatrices. For example, thread 0 performs addition on A00 and B00, thread 1 performs addition on A01 and B01, . . . etc.

The final result should be stored in a matrix C of size R x C.

* You must divide the two-dimensional array into the form such as: . Same for *B* and *C*.

Using the above example, if

A = 2 3 1 2 5 1

3 1 2 2 2 4

1 2 3 2 7 2

3 6 1 5 1 3

Then the upper left corner is

= 2 3 1

3 1 2

One of your threads is responsible for adding

Same as

2 3 1 + 6 5 4 = 8 8 5

3 1 2 3 3 2 6 4 3

**List of classes that you will write:**

* Main – contains the main method.
* ThreadOperation – extends Thread and performs submatrix addition

**Instructions for Part 1**

For part 1 you need to create both of the above classes.

1. In the main method of Main, instantiate four ThreadOperation objects, start them, and join them. Each ThreadOperation will take as input (through the constructor) two matrices and a quadrant indicator. The indicator could be a String, an int, an enum or a set of indexes. It’s up to you.
2. In Main.java, write a static method named print2dArray that takes a two-dimensional array as input and prints it out with the rows and columns lined up. You must use System.out.printf.
3. Instantiate a test 2d array with any values you like in main and use it to verify that print2dArray works.
4. The filename should be given through the command prompt and passed into main via String[] args
5. Open and connect to the file using a Scanner.
6. Read in the number of rows and columns and save these in local variables in main.
7. Read in the first and second matrices (two-dimensional arrays) from the file. I recommend writing a method to accomplish this task and calling the method twice (once for each matrix). Consider using this method header:

public static int[][] matrixFromFile(int rows, int columns, Scanner file\_reader)

NOTE: if you are using a static scanner or an object-oriented approach then you may not need to pass these arguments to the method.

**Information on the file format**

1. the first line has two numbers, R and C (R rows, C columns), the size of both matrices A and B
2. the next R lines each has C elements for one of the rows of A
3. the next R lines each has C elements for one of the rows of B

Example:

4 6

2 3 1 2 5 1

3 1 2 2 2 4

1 2 3 2 7 2

3 6 1 5 1 3

6 5 4 1 4 3

3 3 2 2 1 1

7 5 4 3 2 5

2 1 8 4 8 4

For the above example, 4 is the number of rows, 6 is the number of columns. The first matrix values are highlighted in green and the second matrix is highlighted in red. The result of the sum should be as follow:

8 8 5 3 9 4

6 4 4 4 3 5

8 7 7 5 9 7

5 7 9 9 9 7

Example: The upper left quadrants of the corresponding matrices (highlighted in yellow) will be added together

4 6

2 3 1 2 5 1

3 1 2 2 2 4

1 2 3 2 7 2

3 6 1 5 1 3

6 5 4 1 4 3

3 3 2 2 1 1

7 5 4 3 2 5

2 1 8 4 8 4

For your convenience, three test cases are provided: matrix1.txt, matrix2.txt, and matrix3.txt.

One of the goals is to minimize the resource usage, such as memory and processor cycles. **Explain how multi-threaded code accomplishes this goal in your document.** **YOU MUST ANSWER THIS QUESTION IN A COMMENT AT THE TOP OF YOUR Main CLASS. Tell me about blocking on I/O, multicore machines, how sluggish humans are, etcetera, and then tell me how multi-threading helps. Compare threads to processes and tell me the advantages of multi-threading.** It doesn’t have to be long. Three sentences will suffice if they are good sentences.

**UML Diagram for Matrix Addition Part 1**

|  |
| --- |
| **Main** |
|  |
| + print2dArray(matrix: int[][]) : void |

|  |
| --- |
| **ThreadOperation** |
| - A : int[][]  - B : int[][]  - quadrant : String |
| <<constructor>>ThreadOperation(A : int[][], B : int[][], quadrant : String)  + run() : void |

**Compilation and Execution**

I will test your program as follows:

javac \*.java

java Main matrix1.txt

or

java Main matrix2.txt

java Main matrix3.txt