Lab 9 – From the 2nd Dimension
CSCI 1101B – Spring 2015
Due: May 5, 10 pm

Objective: To gain experience working with multidimensional arrays and for loops.

Part 1: Row Count

In the `RowCount` class, write a method called `computeRowSums` that computes the sums of the rows of a 2-dimensional array. This method should be sent a 2D array of integers, compute the sum of the elements in each row of the 2D array, and store the row sums in a new 1D array. In other words, element `i` of the resulting array should contain the sum of all the elements in row `i` of the original 2D array.

Code to test the `computeRowSums` method is already provided in the `begin` method. While you are welcome to add your own code testing your method, you should not change the existing test code.

Note that, as in the provided testing code, the length of each row of the input array may not be same!

Part 2: Blur

Write a `Blur` class that creates a grid of randomly colored squares and has a method that blurs the colors. The constructor should create a grid of colored squares with the numbers of rows and columns passed as parameters. The upper left corner of the grid should be at coordinates (20, 20). Each square should be 8×8 pixels and should be randomly colored, **except** for the outer-most squares, which should all be black.

Every time the `makeBlurrier` method is called, the method should blur the pattern by performing a series of steps on each square **except** for those in the first or last row or column (i.e., the black squares). For each such square:

1. get the colors in the squares north, south, east, and west of the square,
2. calculate the average of each color level (red, green, and blue) of those four colors,
3. create a new color using those averages, and
4. set the color of the square to that new color.

In other words, the red, green, and blue levels of the new color for the square at row **r**, column **c** in this diagram would be the average of the red, green, and blue levels, respectively, of colors C1, C2, C3, and C4.

```
                 |----|
. . . . . . . .  | C1 |  . . . . . . . .
            |----|----|----|
. . . . . . | C4 |r,c | C2 | . . . . . .
            |----|----|----|
. . . . . . . .  | C3 |  . . . . . . . .
                 |----|
```

Notice that as soon as you blur the color of a square, that new blurred color will be used in the computation of the new colors of the squares East and South of it (assuming that you are processing the squares row by row and left to right within each row). If we did *not* want that to happen we would have to put the new colors in a *separate array* as we calculate them and then use those to reset the square colors in the original array when we are completely done calculating the new colors. However, it is fine to reuse the same (original) array here.

The starter project contain a `BlurEvents` class that uses the `Blur` class, which you should use to test your program. If your class is working correctly, when the mouse is dragged, the squares will fade to black, starting at the edges and progressing inwards. **You should not change the `BlurEvents` class.**

Submission: Submit your project in the usual way as a compressed file on Blackboard. Don't forget to put your name on your files and fill in the usual comment blocks!