# Assignment 10: More Arrays

CS 140 with Dr. Sam Schwartz

Due: Sunday, November 23 at 11:59pm via Canvas Upload

## 1 Purpose

The purpose of this assignment is to gain more experience with single and multidimensional arrays

## 2 Tasks

In this assignment you will complete the following three tasks. These tasks will require different functions, but they should all live in the same Assignment10.java file for the purpose of this assignment.

A starter file is provided with test cases. You must use this starter file and the code already written.

Take a screenshot of your work and upload it alongside your code to Canvas.

### Task 1 – Array Compression

Implement a function to perform basic array compression using the counts of repeated values. Specifically:

Convert an integer array into a run-length encoding where each row contains: {value, count}.

This function should have a method signature like so:

```
public static int[][] compressRuns(int[] arr)
```

Example:

Input: {4,4,4,1,1,2}
Output: {{4,3},{1,2},{2,1}}

Also implement the corresponding decompress method.

This function should have a method signature like so:

```
public static int[] decompressRuns(int[][] compressedArr)
```

### Task 2 – Find All Local Maxima

A local maximum is an element strictly greater than its neighbors.

Example: In {2, 7, 5, 1, 3, 3, 9}], the local maxima are 7 and 9.

This function should have a method signature like so:

```
public static int[] localMaxima(int[] arr)
```

Return an array containing all the local maxima.

If there are no local maxima, return an empty array.

## Task 3 – Image Bluring

In the starter file provided are helper methods that open and reads data from a file called `lincoln.ppm` and saves data to a file called `blurredLincoln.jpg`.

You should save lincoln.ppm to your computer and note where it's saved to.

A PPM file is an uncompressed image file. Nearly all image files (GIF, JPG, PNG, etc.) use some sort of compression algorithm to store pixel data.

In a PPM file the image data is not compressed. The lincoln.ppm file we will be using has the form:

```
P2 # The string 'P2' indicates that this is a grayscale image.
# P1 is used for black and white only, P3 for color.
300 400 # In an inversion of the usual row, col format, these numbers indicate the image has
# 24 columns and 7 rows of pixels.
255 # This indicates the range of each pixel is in [0, 255]
# Pixel data goes here
161 179 162 165 171 171 171 159 176 170 162 173 173 173 164 175 166 166 164 162 176 177 160 165 158 18
152 163 167 182 169 152 171 182 163 168 163 171 146 159 170 157 163 170 165 160 164 146 150 167 173 17
158 166 172 177 178 165 179 193 158 169 188 156 174 160 180 155 164 171 160 154 180 162 178 171 168 17
161 148 153 178 171 163 164 171 161 171 158 173 151 171 176 175 174 160 166 157 171 156 162 161 173 16
175 154 170 149 183 167 186 180 181 171 155 182 158 160 148 152 168 172 170 167 166 157 167 170 158 16
...
```

While many computers can read PPM files, some newer versions of windows do not support this functionality out of the box. If that's the case, and you don't want to pay for photo editing software, `irfanview.com` has a free graphics viewer that can open PPM files.

Your task is to blur the photo of President Lincoln by doing the following:

Each entity becomes the average of itself and its (up to 8) neighbors, using integer division.

That is, suppose we have this array:

```
int[][] imgData = {
 {a, b, c, d},
 {e, f, g, h},
 {i, j, k, l},
 {m, n, o, p},
}
```

then aBlur = (a+b+f+e) / 4
and bBlur = (a+b+c+e+f+g) / 6
...
and kBlur = (f+g+h+j+k+l+n+o+p) / 9
... and so forth, all the way to pBlur.

Do this for all elements in the whole array, and return a new array composed of these newly blurred elements. That is, given the `imgData` example above, you should return

```
int[][] blurredImgData = {
 {aBlur, bBlur, cBlur, dBlur},
 {eBlur, fBlur, gBlur, hBlur},
 {iBlur, jBlur, kBlur, lBlur},
 {mBlur, nBlur, oBlur, pBlur},
}
return blurredImgData
```

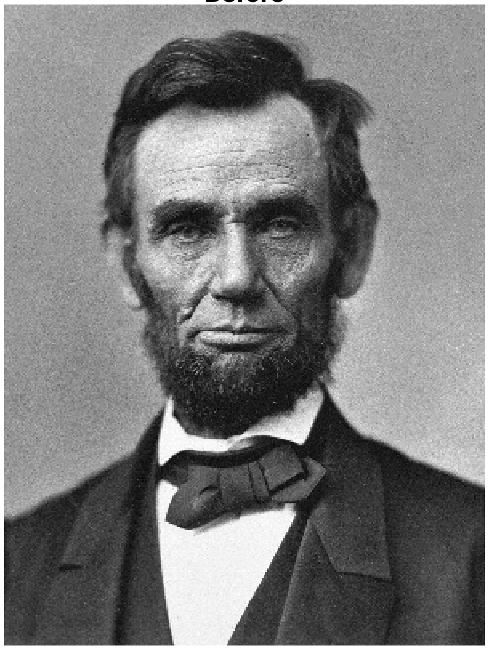
You should modify the starter code to put your own file locations in place, and upload the blurred image when done.

The starter code has this method signature:

```
public static int[][] blurImageData(int[][] data)
```

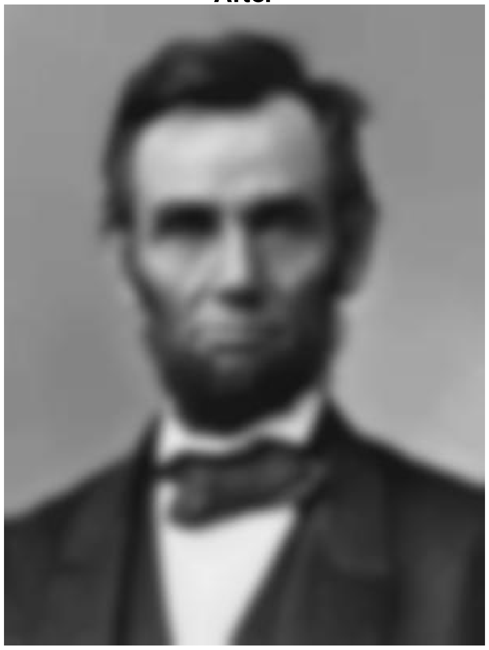
This is the only thing you're expected to implement. Everything else (opening and processing the file, writing the new file, etc.) has been done for you.

**Before**



`lincoln.ppb`

**After**



`blurredLincoln.jpg`

## 3 Grading Criteria

In general I am looking for these elements:

**Validity**

Student submitted a single Assignment10.java file
Which implemented the three tasks correctly
Handled possible errors gracefully
And did not modify code from the starter file that they were told to not modify.

Moreover, the student submitted a screenshot displaying the code's output (all of it, in one single screenshot.)

And also uploaded their blurred image of President Lincoln.

**Readability**

... which used professional English and typesetting throughout all documents.
... and had all methods annotated with JavaDoc

**Fluency**

... and the deliverables were executed in such a way that an experienced practitioner would not find the deliverable "weird-in-a-bad-way" or unduly jarring.