5. A grayscale image is represented by a 2-dimensional rectangular array of pixels (picture elements). A pixel is an integer value that represents a shade of gray. In this question, pixel values can be in the range from 0 through 255, inclusive. A black pixel is represented by 0, and a white pixel is represented by 255. The declaration of the GrayImage class is shown below.

a) Write the method countWhitePixels that returns the number of pixels in the image that contain the value WHITE. For example, assume that pixelValues contains the following image.

	0	1	2	3	4
0	**255**	184	178	84	129
1	84	**255**	**255**	130	84
2	78	**255**	0	0	78
3	84	130	**255**	130	84

Figure 1: Example 2D array

A call to countWhitePixels method would return 5 because there are 5 entries (shown in boldface) that have the value WHITE. **[4 marks]**

b) Write the method processImage that modifies the image by changing the values in the instance variable pixelValues according to the following description. The pixels in the image are processed one at a time in row-major order. Row-major order processes the first row in the array from left to right and then processes the second row from left to right, continuing until all rows are processed from left to right. The first index of pixelValues represents the row number, and the second index represents the column number.

The pixel value at position (row, col) is decreased by the value at position (row + 2, col + 2) if such a position exists. If the result of the subtraction is less than the value BLACK, the pixel is assigned the value of BLACK. The values of the pixels for which there is no pixel at position (row + 2, col + 2) remain unchanged. You may assume that all the original values in the array are within the range [BLACK, WHITE], inclusive.

The following diagram shows the contents of the instance variable pixelValues before and after a call to processImage. The values shown in boldface represent the pixels that could be modified in a grayscale image with 4 rows and 5 columns. **[6 marks]**

Before Call to `processImage`

	0	1	2	3	4
0	**221**	**184**	**178**	84	135
1	**84**	**255**	**255**	130	84
2	78	255	0	0	78
3	84	130	255	130	84

After Call to `processImage`

	0	1	2	3	4
0	**221**	**184**	**100**	84	135
1	**0**	**125**	**171**	130	84
2	78	255	0	0	78
3	84	130	255	130	84

Figure 1: Example before and after a call to processImage