8. (40 points) A Line is a line defined by the equation $ax + by + c = 0$, where a is not equal to zero, b is not equal to zero, and a, b, and c are all integers. The slope of a Line is defined to be the double value $-a/b$. A point (represented by integers x and y) is on a Line if the equation of the Line is satisfied when those x and y values are substituted into the equation. That is, a point represented by x and y is on the line if $ax + by + c$ is equal to 0. Examples of two Line equations are shown in the following table.

Equation	Slope (a/b)	Is point $(5, -2)$ on the line?
$5x + 4y - 17 = 0$	$-5/4 = -1.25$	Yes, because $5(5) + 4(-2) + (-17) = 0$
$-25x + 40y + 30 = 0$	$25/40 = 0.625$	No, because $-25(5) + 40(-2) + 300$

Assume that the following code segment appears in a class other than Line. The code segment shows an example of using the Line class to represent the two equations shown in the table.

```
Line line1(5, 4, -17);
double slope1 = line1.getSlope();
bool onLine1 = line1.isOnLine(5, -2);

Line * line2 = new Line(-25, 40, 30);
double slope2 = line2->getSlope();
bool onLine2 = line2->isOnLine(5, -2);

Line line3(line1);
Line line4(1, 3, 5);
line4 = line1;
```

Write the Line class. Your implementation must include **a constructor that has three integer parameters that represent a, b, and c, in that order, a copy constructor, and an assignment operator= overloading function**. You may assume that the values of the parameters representing a and b are not zero. It must also include a function getSlope that calculates and returns the slope of the line, and a function isOnLine that returns true if the point represented by its two parameters (x and y, in that order) is on the Line and returns false otherwise. Your class must produce the indicated results when invoked by the code segment given above. You may ignore any issues related to integer overflow.

This page is left blank on purpose.

9. (50 points) A grayscale image is represented by a 2-dimensional rectangular array of pixels (picture elements). A pixel is represented by 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. You will write two unrelated member functions of the GrayImage class.

```
//Global variables
const int BLACK = 0;
const int WHITE = 255;

class GrayImage
{
public:
   // Return the total number of white pixels in this image.
   // Postcondition: this image has not been changed.
   int countWhitePixels();  //To be completed

   //Processes this image in row-major order and decreases the value of each
   //pixel at position (row, col) by the value of the pixel at position
   //(row + 2, col + 2) if it exists. Resulting values that would be less than
   //BLACK are replaced by BLACK. Pixels for which there is no pixel at
   //position (row + 2, col + 2) are unchanged.
   void processImage();  //To be completed

   //Return a pointer to a Pixel object at (row, col)
   Pixel * getPixel(int row, int col);
   int getRows();  // Return the total number of rows of this image
   int getCols(); // Return the total number of columns of this image

private:
   int rows;  // Total number of rows in this image
   int cols; // Total number of columns in this image

  // The 2-dimensional representation of this image.
  // That is a 2D array of pointers to Pixel objects
   Pixel *** pixels;
};

class Pixel
{
public:
    //To be completed
   //Parameterized constructor
```

```
   //Necessary getters and setters

private:
   int value;  // Pixel value
   int row;    // Row index of this pixel in an image
   int col;   // Column index of this pixel in an image
   GrayImage * img;  //The image to which this pixel belongs
}
```

(a) Add a constructor, and necessary getters and setters for the class Pixel that would be used in the processImage function in the next question.

(b) Write the function countWhitePixels() that returns the number of pixels in the image that contain the value WHITE. For example, assume that an image contains the following pixel values.

255	184	178	84	129
84	**255**	**255**	130	84
78	**255**	0	0	78
84	130	**255**	130	84

A call to countWhitePixels function would return 5 because there are 5 entries (shown in boldface) that have the value WHITE.

(c) Write the function processImage that modifies the image by changing the pixel values of an image 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 a pixel represents the row number, and the second index represents the column number.

The pixel 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 pixel values in the image are within the range of (0, 255) or (BLACK, WHITE), inclusive.

The following shows the contents of the pixel values 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.

Before Call to processImage

221	**184**	**178**	84	135
84	**255**	**255**	130	84
78	255	0	0	78
84	130	255	130	84

After Call to processImage

221	**184**	**100**	84	135
0	**125**	**171**	130	84
78	255	0	0	78
84	130	255	130	84

10. (15 points (bonus)) Add a function that returns a histogram of an image for class GrayImage. The histogram of an image describes the distribution of pixel values of the image, normally represented as a bar chart. The horizontal axis is used for pixel values from 0 to 255. The vertical axis or bar height represents the count of pixels having the cooresponding pixel value. For example, a pair of coordinates (126, 450) mean that there are 450 pixels in the image with a pixel value of 126.

```
// Return a one-dimensional array of integers with the indices representing
// pixel values and its elements representing the counts of cooresponding
// pixel values in this image.

int * GrayImage::getHistogram();
```