Google Doc Access Directions:

- Please click on **File** in the upper left corner.
- If you are working on a Chromebook or Google Docs, choose the **Make a copy option** and save a copy of the document to your Google Drive.
- If not, choose the **Download** as option and then the **Microsoft Word (.docx)** option to download an editable copy of the document to your computer.

Using the information you have gained so far, modify the code. Click “Submit Assignment” in the upper right corner of the screen to submit your work. Be sure and save the files as YourNameMod13PictureLabAssignmentNine

1. Create a second copy method that adds parameters to allow you to copy just part of the `fromPic`. You will need to add parameters that specify the start row, end row, start column, and end column to copy from. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

```
public static void testCopy() {
  Picture beach = new Picture(fileName:"beach.jpg");
  Picture motorcycle = new Picture(fileName:"motorcycle.jpg");
  beach.copy(motorcycle, startRow:10, endRow:100, startCol:20, endCol:300, toStartRow:50, …50);
  beach.explore();
}
```

```
public void copy(Picture fromPic, int startRow, int endRow, int startCol, int endCol, int toStartRow,
    int toStartCol) {
  Pixel fromPixel = null;
  Pixel toPixel = null;
  Pixel[][] toPixels = this.getPixels2D();
  Pixel[][] fromPixels = fromPic.getPixels2D();
  for (int fromRow = startRow, toRow = toStartRow; fromRow <= endRow && toRow < toPixels.length; fromRow++, toRow++) {
    for (int fromCol = startCol, toCol = toStartCol; fromCol <= endCol
        && toCol < toPixels[0].length; fromCol++, toCol++) {
      fromPixel = fromPixels[fromRow][fromCol];
      toPixel = toPixels[toRow][toCol];
      toPixel.setColor(fromPixel.getColor());
    }
  }
}
```

2. Create a `myCollage` method that has at least three pictures (can be the same picture) copied three times with three different picture manipulations and at least one mirroring. Write a class (static) test method in `PictureTester` to test this new method and call it in the `main` method.

Module13LessonTwoAssignmentNine Name_____

```
public void myCollage() {
  Picture flower1 = new Picture(fileName:"flower1.jpg");
  Picture flower2 = new Picture(fileName:"flower2.jpg");
  Picture flower3 = new Picture(fileName:"flower3.jpg");

  // Apply different manipulations to each picture
  flower1.zeroBlue();
  flower2.mirrorVertical();
  flower3.mirrorHorizontal();

  // Copy each picture into this picture
  this.copy(flower1, startRow:0, startCol:0);
  this.copy(flower2, startRow:100, startCol:0);
  this.copy(flower3, startRow:200, startCol:0);

  // Additional manipulation - mirror part of the collage
  this.mirrorVerticalRightToLeft();
}
```

```
public static void testMyCollage() {
  Picture canvas = new Picture();
  canvas.myCollage();
  canvas.explore();
}
```

3. Notice that the current edge detection method works best when there are big color changes from left to right but not when the changes are from top to bottom. Add another loop that compares the current pixel with the one below and sets the current pixel color to black as well when the color distance is greater than the specified edge distance.

Module13LessonTwoAssignmentNine Name_____

```
public void edgeDetection(int edgeDist) {
  Pixel currentPixel = null;
  Pixel rightPixel = null;
  Pixel bottomPixel = null;
  Pixel[][] pixels = this.getPixels2D();
  Color rightColor = null;
  Color bottomColor = null;

  // Detect horizontal edges
  for (int row = 0; row < pixels.length; row++) {
    for (int col = 0; col < pixels[0].length - 1; col++) {
      currentPixel = pixels[row][col];
      rightPixel = pixels[row][col + 1];
      rightColor = rightPixel.getColor();
      if (currentPixel.colorDistance(rightColor) > edgeDist) {
        currentPixel.setColor(Color.BLACK);
      } else {
        currentPixel.setColor(Color.WHITE);
      }
    }
  }

  // Detect vertical edges
  for (int row = 0; row < pixels.length - 1; row++) {
    for (int col = 0; col < pixels[0].length; col++) {
      currentPixel = pixels[row][col];
      bottomPixel = pixels[row + 1][col];
      bottomColor = bottomPixel.getColor();
      if (currentPixel.colorDistance(bottomColor) > edgeDist) {
        currentPixel.setColor(Color.BLACK);
      }
      // Note: There's no need to set it to white again as it might overwrite the
      // horizontal edge detection
    }
  }
}
```