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.
   1. public void preciseCopy(Picture fromPic, int startRow, int startCol, int endRow, int endCol) {
   2. Pixel fromPixel = null;
   3. Pixel toPixel = null;
   4. Pixel[][] toPixels = this.getPixels2D();
   5. Pixel[][] fromPixels = fromPic.getPixels2D();
   6. for (int fromRow = 0, toRow = startRow; toRow < endRow && fromRow < fromPixels.length; fromRow++, toRow++) {
   7. for (int fromCol = 0, toCol = startCol; toCol < endCol && fromCol < fromPixels[0].length; fromCol++, toCol++) {
   8. fromPixel = fromPixels[fromRow][fromCol];
   9. toPixel = toPixels[toRow][toCol];
   10. toPixel.setColor(fromPixel.getColor());
   11. }
   12. }
   13. }
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.
   1. public void myCollage()
   2. {
   3. Picture flower1 = new Picture("pixLab\\images\\flower1.jpg");
   4. flower1.mirrorDiagonal();
   5. this.preciseCopy(flower1, 30, 0, 100, 80);
   6. Picture caterpillar = new Picture("pixlab\\images\\caterpillar.jpg");
   7. caterpillar.mirrorHorizontalBotToTop();
   8. this.preciseCopy(caterpillar, 200, 100, 300, 200);
   9. Picture arch = new Picture("pixlab\\images\\arch.jpg");
   10. arch.negateImage();
   11. this.preciseCopy(arch, 400, 300, 450,500);
   12. this.write("mycollage.jpg");
   13. }
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.
   1. public void myEdgeDetection(int edgeDist)
   2. {
   3. Pixel leftPixel = null;
   4. Pixel rightPixel = null;
   5. Pixel[][] pixels = this.getPixels2D();
   6. Color rightColor = null;
   7. for (int row = 0; row < pixels.length; row++)
   8. {
   9. for (int col = 0; col < pixels[0].length-1; col++)
   10. {
   11. leftPixel = pixels[row][col];
   12. rightPixel = pixels[row][col+1];
   13. rightColor = rightPixel.getColor();
   14. if (leftPixel.colorDistance(rightColor) >
   15. edgeDist)
   16. leftPixel.setColor(Color.BLACK);
   17. else
   18. leftPixel.setColor(Color.WHITE);
   19. }
   20. }
   21. for (int col = 0; col < pixels[0].length; col++) {
   22. for (int row = 0; row < pixels.length - 1; row++) {
   23. leftPixel = pixels[row][col];
   24. rightPixel = pixels[row+1][col];
   25. rightColor = rightPixel.getColor();
   26. if (leftPixel.colorDistance(rightColor) >
   27. edgeDist)
   28. leftPixel.setColor(Color.BLACK);
   29. else
   30. leftPixel.setColor(Color.WHITE);
   31. }
   32. }
   33. }