Practice Quiz: Functions

Programming Exercise Guide

Before attempting this quiz you should write the following JavaScript programs.

1) Write a JavaScript program that has a function named `swapRedGreen` with one parameter, a pixel. This function should swap the red and green values of the pixel. Pick an image, print the image, then apply the `swapRedGreen` to every pixel in the image, and print the new image. The choice of your image is important. For some images you may not notice any change. Think about what type of image you should use for testing your function.
2) Write a JavaScript program to make an image have more red in it, by adding a given value to the red, making sure it doesn't go over 255. Your program should have a function called `moreRed` with two parameters, a pixel and a value to increase the red by. Run your program on an image to see it get redder.
3) Write the complete JavaScript program to put the border around a picture and include the following functions that are included from the lesson. You should be able to write these functions without looking at the code from the lesson. Be sure to print the image so you can see it and run the program with different border values.
    a) `function setBlack(pixel)` - This function has a parameter `pixel` that represents a single pixel, and returns a pixel that has been changed to be the color black.
    b) `function pixelOnEdge(pixel, image, borderWidth)` - This function has three parameters where `pixel` is a single pixel, `image` is the complete image, and `borderWidth` is an integer specifying the thickness of the borders. This function returns true if the pixel's location is within `borderWidth` of any of the four borders, and thus on the border. Otherwise it returns false.
4) Now modify the border program to specify two thicknesses, one for the vertical borders and one for the horizontal borders. You should write the two boolean functions shown below. Be sure to print the image and run your program with different border values for the horizontal and vertical edges.
    a) `function pixelOnVerticalEdge(pixel, image, borderWidth)` - This function has three parameters where `pixel` is a single pixel, `image` is the complete image, and `borderWidth` is an integer specifying the thickness of the *vertical* borders. This function returns true if the pixel's location is within `borderWidth` of any of the two vertical borders, and thus on the border. Otherwise it returns false.
    b) `function pixelOnHorizontalEdge(pixel, image, borderWidth)` - This function has three parameters where `pixel` is a single pixel, `image` is the complete image, and `borderWidth` is an integer specifying the thickness of the *horizontal* borders. This function returns true if the pixel's location is within

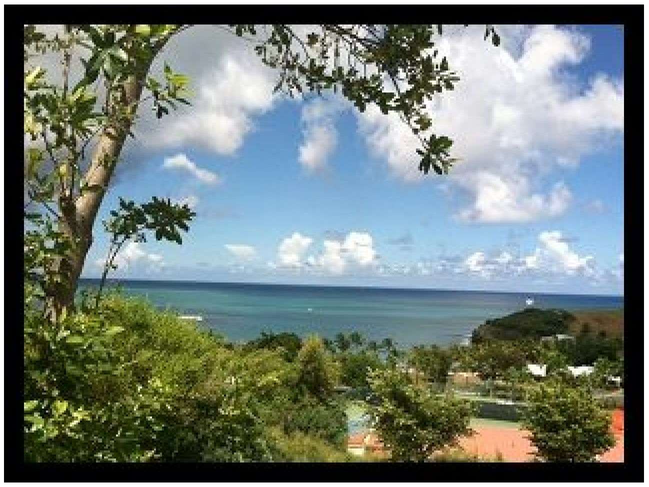
`borderWidth` of any of the two horizontal borders, and thus on the border. Otherwise it returns false.

5) Now modify the program one more time to replace the two functions `pixelOnVerticalEdge` and `pixelOnHorizontalEdge` with one function to do the same thing, called `pixelOnEdgeDifferentThicknesses()`. The parameters are not shown. You should decide on the parameters. Write the function and test it to make sure it works the same as the two functions it replaces, so you can generate pictures with borders that have different thicknesses for the vertical borders vs. the horizontal borders.

Here is an example:

Here is an island picture:

Here is the island picture with a border around it:



Here is a picture of the island with different thicknesses for the vertical versus horizontal borders: