Fault Injections

For our injected faults we went in and changed 5 of the methods that our drivers were testing. The directions injecting the faults can be found commented in the code as well as in this document. The first method that we injected a fault into was the getContrastRatio5DigitRound() found in the contrast checker class. This method takes in two colors as its parameters, one being foreground and the other the background. It then finds their luminosity and computes the contrast ratio of the two colors. The method assumes that the first input is the foreground color and the second is the background color. It does a simple check to see if this is true by testing to see if the luminosity of the foreground color is greater than the background color. Our fault changed this check by having the luminosity of the background color be greater than the foreground color. Simply change if(fgLuminosity > bgLuminosity) to if(fgLuminosity < bgLuminosity). This made test cases 3 through 5 fail. The first two passed as they were checking the same color and thus wouldn't change depending on which was the foreground vs background.

The second method that we inserted a fault into was the getLuminosity() method found in the contrast checker method. This method finds and returns the luminosity of a color. It does this by combining the composite red, green and blue colors multiplied by their respective multipliers. Our injection fault took these and mixed them up so that all the colors would be multiplied by the wrong color factor. To get the same results either uncomment the three statements in the code and comment out their respective correct lines, or do the following. Multiply the composent green value with the red factor, then add that to the product of the composent red value and the blue factor. Finally add this to the compsent blue value multiplied by the red factor. Having this fault will fail tests 8 through 10. Tests 6 and 7 won't fail from this fault as they test the luminosity of black and white.

The next test was conducted on the calculate() method in the distance converter class. We simply changed one of the additions to a subtraction. To get the same results uncomment the last line and comment out the line above or just change the addition on line 44 into a subtraction. This will fail test cases 13 through 15. Test cases 11 and 12 test black and white and thus won't change because of this fault.

The fourth test case injects a fault into the rgb2hsl() method found in the color converter class. This method converts RGB into HSL. HSL stands for hue, saturation and light. This is another way of categorizing colors. The fault we injected here switched a / for a *. Simply instead of having a division we changed it to a multiplication. The division on line 209 should be switched to multiplication or uncomment the line below and comment out line 209. This will cause test cases 16, 17, 18 and 20 to fail. Test case 19 is uses white and thus will not change because of this alteration.

Finally the last injection was in the method hex2Rgb() in the color converter class. There are two of these methods, we changed the one beginning on line 124. This method converts hex values into their RGB components. This injection fault comes in two parts. The first part on line 128 where you change colorStr.substring(1) to colorStr.substring(0). Following this on line 139 after colorStr.length() add a - 1 so it reads colorStr.length() - 1. By doing these two things test cases 20 through 24 will fail. Test case 25 will still pass as its testing whether incorrect input will be returned as null.

These are the 5 injection faults that have been added. It wasn't very hard to break some of them, but others like the hex2RGB were a lot harder. Overall injecting faults was easier than creating test cases, but not as interesting.