Katie Robinson
Test Oracles
Quadrilaterals #6

The primary oracle I utilized was a nop; I generated 1000 text files based on one original file that had known classifications (I had generated them by hand) of small quadrilaterals, and each of the new text files contained coordinates that were scaled by a random factor. This was possible because no quadrilaterals change classification when they are scaled by a positive, nonzero integer. They were scaled by row so every new file, when tested, would still output the same classifications. This allowed me to use the “diff” tool to compare every test file result to a known output file and verify that there were no differences between them.

Note: I wanted to experiment with shifts and rotations since it would also not change a quadrilateral’s classification, but given the requirement that one point was at (0,0) and no values could be negatives or non-integers, this was not possible. These would have been great nops.

The second oracle I used was assertions; I sprinkled my original quadrilateral classifier C++ code with assert() statements in order to provide more locations where a bug from all the new random input would cause the program to crash in a controlled and informative way. I did not actually find any bugs with this technique, but I can see it is a powerful one given that it can limit the scope of poor assumptions and keep crashes close to the source of a bug.

A third oracle I used was another nop, which involved creating random test files guaranteed to fail and trigger error1, error2, error3, or error4 when run on the classifier. By preventing bad input from being entered into the classifier via error handling, I was able to cause the program to stop and report the error in a controlled manner instead of crash.