Address sanitizer:

I decided to employ address sanitizer as one of my testing oracles. I learned about the address sanitizer within xcode last semester. We were taught when programming in c++ that we should always employ address sanitizer for our own benefit and code safety. Once I learned address sanitizer was a type of oracle I learned how to run it with clang++ and again employed that technique. From my research about Asan and how it works:

Helps prevent things like memory corruption, dangling pointers, buffer overflow, etc.

Assertions:
The second oracle i employed was assertions. I ended up removing them from my code altogether due to them not really making a whole lot of sense within my project. The context in which I was using them was to test for basically impossible situations.
Ex:
Checking to see if my quadrilateral object was null, or if the quadrilateral slopes where blank.

I understand assertions to be tests to prove that something is what the programmer believes it to be. Inserting them strategically within my code from the the start I think would've been useful as I could have checked certain conditions for validity early on. Again I removed them because the just didn't make much sense in my code at this point considering the other error checking that I had implemented.

Expected output vs Program output key:

I'm not sure of the proper naming convention but one of the testing oracles I employed in my automated testing was to generate shapes such as square, rectangle, trapezoid, etc. I then scale the shapes and output their values to a text file. After I generated a test file of the same number of outputs, let's call that expected I then compared the expected output with the program output by running each through a series of shell script tests. If the test failed I know there is a bug with my program. Ex: when scaling each shape I knew or at least expected the output to be whatever shape I am currently scaling. Testing square, I expect the output to be "SQUARE", you run that against the key and if there is a difference, you know there is a bug in your program somewhere.
I took that concept and applied it to all the various shapes. I also chose to scale co-linear points as they were difficult to generate randomly.