Alex Sloan

***Note:** Inside of my directory, the primary files associated with this project are:main.cpp, tests.sh, coverage.sh, shapes.py, the fuzz executable, and all of the test files are located in the "testFiles" directory.

Checking against expected output:

- For this oracle, I made a series of tests that would run points through my program where I knew what the outcome should be. To do this, in the example of a square, I started with the smallest square possible: 0, 0, 1, 0, 1, 1, 0, 1. I then scaled the shape up to the max value (100). This created 100 sets of square coordinates for me to run through my program. I then made an expected-output file, which just has 100 lines that say "square". After that, I checked my program's output vs. the expected output file. If any errors were to pop up, this would show me that I'm calculating squares incorrectly. I repeated this process for other shapes, and it served to show me a few things that I needed to correct.

Assertions:

- An assertion is a way of adding some checks to your program. It basically involves a Boolean expression, and if the result is not what is expected, a message is written to the standard error device and abort is called, which terminates the program. I used some assertions within my program to re-check my logic. For example, if a shape was determined to be a square, I asserted that it is also a rectangle. If a shape was determined to be a rhombus, I asserted that it is also a parallelogram. If assertions like these were to fail, I would know that I have invalid logic within my program because, as everyone knows, a square is also a rectangle.

Address Sanitizer:

- For this oracle, I compiled my program with the address sanitizer enabled. Address Sanitizer is a fast memory error detector that consists of a compiler instrumentation module and a run-time library. It's capable of detecting the following bugs:
    - Out-of-bounds accesses to heap, stack, and globals
    - Use-after-free
    - Use-after-return
    - Double-free, invalid free
    - Memory leaks
- This oracle provides an extra layer of error-detection. It's so thorough that it slowed down the compilation of my project by around 2x.