Oracle Testing

Scaling:
For my first oracle I decided to scale up the various parallelograms so that I could test hundreds of them at a time and know the expected output. Essentially a square will always be a square if you add 1 to certain points. By starting with a small square you can scale it up all the way to one hundred which is the limit of our program. I did this for square, rectangle, trapezoid and parallelograms. I could have made 100 different files but chose to compare one file with one hundred inputs. I saved the many many files for the random output oracle I chose to do next.

Random Outputs:
As mentioned above I outputted 1000 random files each with a unique set of six numbers between 0 and 100. This was to test the robustness of my program for sheer amount of inputs. Since there is no way I can make an output file to test against the randomly generated numbers for the files I looked through all the output files to see the ratio of errors to actual quadrilaterals. This output should classify more errors and quadrilaterals than squares and the more unique shapes.

Far and away the most common output for these randomly generated was error 3. Second most common was the generic quadrilateral classification. This makes reasonable sense because error 3 accounts for lines intersecting and random numbers don't go counter clockwise like my program wants. When it's not error 3 it would be quadrilateral. Very rarely did I see anything else in the output.

Address Sanitizer:
In the make file I added the address sanitizer to the clang compilation. This is to cover up any gaps in my logic for looping through various vectors or forgotten memory. Turns out I didn't turn on the address sanitizer for the entire project so when I ran it I had something like 2,000 errors. I pointed me to the loops I was using to determine if all slopes/lengths were equal. I refactored the code to not loop and made a few helper functions and the errors were fixed. Because I ran the program mostly through the terminal for testing I forgot to use the address sanitizer. Adding this line to the testing shell script ran it with it, clang++ -std=c++11 -fsanitize=address main.cpp -o main;