# Oracle 1: Double or halfing test

./python_scripts/doubleHalftest.py

In this oracle I do two tests that multiple/divide the result:

**Double testing:**

For valid inputs that produce a non-error result (prints a one of the quadrilaterals), I check if each and every point is less than or equal to 50. If every points are less than 50 I will multiply each point by 2 and check that the result is the same. If the result is the same as the original output, the test has passed. If the output is different, then the test has failed.

Valid example:
"1 22 3 4 50 6" -> "2 44 6 8 100 12"

Invalid example:
"1 22 3 4 **51** 6" -> invalid because 51 x 2 = 102 which produces an "error 1"

**Half testing:**

For valid inputs that produce a non-error result, I check that each point is an even number. Odd numbers would result in integer match that would produce a different result, therefore only even numbers are valid for this test. If every number is even in the input, I divide each input by 2 and check that the result is the same. If the result is the same as the original output, the test has passed. If the output is different, the test has failed.

Valid example:
"22 32 4 16 10 4" -> "11 16 2 8 5 2"

Invalid example:
"22 **33** 4 16 10 4" -> invalid because 33 is an odd number

# Oracle 2: Swap points 2 and 4

./python_scripts/swapTest.py

For valid inputs that produce a non-error result, I swap points 2 and 4. The result should be the same as the original result because the shape is the same, only the direction the shape is drawn is changed. If the result is the same as the original output, the test has passed. If the result is different, the test has failed.

Example:
"**1 2** 3 4 **5 6**" -> "**5 6** 3 4 **1 2"**

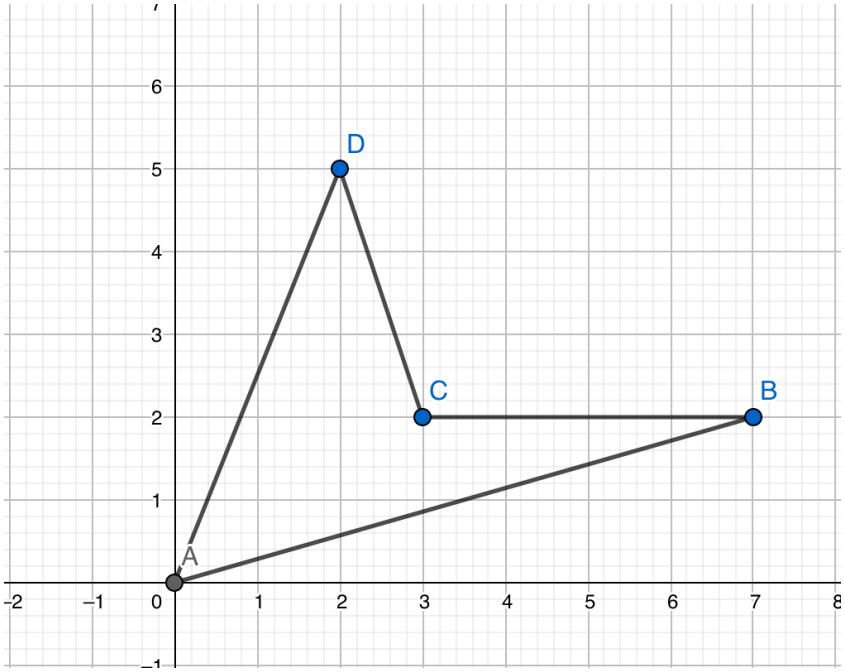
# Oracle 3: Assert statements

I employ 3 assert statements in my code as test oracles.

**assert(splittedLine.size() == 6);**
This assert statement ensures that the input contains 6 elements when split up by spaces.

```
for (string num : splittedLine) {
        assert(stod(num) >= 0 && stod(num) <= 100);
}
```
This assert statement ensures that all the inputs are greater than or equal to zero and less than or equal to 100, as per the spec discussed in class.

**assert(int(p1.angle + p2.angle + p3.angle + p4.angle) <= 360);**
This assert statement ensures that the sum of all angles is less than or equal to 360. Ideally, my assert statement should assert that all angles equal 360. However, my program prints out the outside angle (**angle C** shown below) instead of the inside angle which results in a sum less than 360.




# Oracle 4: Address and Undefined behavior sanitizer

Both the address and undefined behavior sanitizers in Xcode were turned on for this project.