|  |  |  |  |
| --- | --- | --- | --- |
| Test Suite | Test Case | Observed Failure | Fix |
| PointTester::testConstructorWithDoubles | Check for infinity | When checking for point z, the statement was == intead of !=, causing m\_valid to come back false incorrectlt | Change m\_z ==infinity to != |
| PointTester::testIsEquivalentTo | If the edge between two points is less than a min distance | Failure in comparing p0 and p0 -- should be equal, along with other points | In isEquivalentTo,  Change condition to <=mindistance instead of >mindDistance |
| PointTester::testIsEquivalentTo | If two points are the same | getLength() method of edge returned greater than edge minDistance when points were the same | Diffy variable in getLength() method mistakenly compared point y to point x, simplt change x to y so it compares correctly |
| EdgeTester::testEdge01 | Test to see if the slope is what should be expected | Failure in e.getSlopeZ(), slope=inf (expecting 0) | In statement:  **if** (xyOffset != 0) {  result = INFINITY; }  change != to == |
| TriangleTester::testFirstConstructer | On the get triangle type, it returned wrong value | Returned not a triangle, should have returned I | **if** (!isTriangle())  is incorrect, take out the ! so it returns N when not a triangle instead of the other way around |
| TriangleTester::testFirstConstructer | **else if** (approximatelyEquals(a, b, m\_edgeLengthThreshold) ||  approximatelyEquals(b, c, m\_edgeLengthThreshold) ||  approximatelyEquals(c, c, m\_edgeLengthThreshold)) {  ) | The test cases compared c to c, so every triangle would come out as I before s | Changed the cc compare to ac |
| TriangleTester::testFirstConstructor | **if** (!approximatelyEquals(t.computerArea(), 9.35414, 0.001)) {  std::cout << "Triangle: unexpected area of "  << t.computerArea() << std::endl;  **return**; } | Was returning NaN instead of correct area because the equation was wrong | When computing the s value, the code was  **double** s = ( a + b + b)/2;  The 2nd b had to be changed to a c. |
| TriangleTester::testEquilateralTriangles | m\_points = **new** Point\*[3]; m\_points[0] = **new** Point(values[0]); m\_points[1] = **new** Point(values[1]); m\_points[2] = **new** Point(values[1]); | Wasn’t setting up points correctly. | Change last 1 to a 2 |
| TriangleTester::testEquilateralTriangles() | **if** (t1.getTriangleType()!='E') {  std::cout << "Triangle: unexpected type of "  << t1.getTriangleType() << std::endl;  **return**; } | This one took a couple hours…. Stepping through multiple methods, finally found that the point:initialize function was broken | m\_x = convertStringToDouble(values[0], &m\_valid); m\_y = convertStringToDouble(values[1], &m\_valid); m\_x = convertStringToDouble(values[2], &m\_valid);  finally after hours, this was the reason why oints were read in wrong and triangles were decided on type incorrectly. That final m\_x member should be m\_z. And to think I tried to fix the find edge length method :’( |
|  | Equilateral and isocoles testing is mixed |  |  |

Ran out of room for custom test cases, new table:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Suite | Test Case | Observed Failure | Fix |
| TriangleTester::testIsoscelesTriangles | Where it tries to figure out if it is isosceles | It is pretty cleat that the isosceles test is not inclusive enough, it only allows for one type | Add more definitions so that it catches all types of isosceles triangles |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |