|  |  |  |  |
| --- | --- | --- | --- |
| Test Suite | Test Case | Observed Failure | Fix |
| UtilsTester::testConvertToDouble() | Uses numeric\_limits | Not a member of std | #1 |
| PointTester::testConstructorWithDoubles() | Comparing p0 with p0 | Failure in constructing Point(0,0,0) isValid()=0 x=0 y=0 z=0 | #2 |
| PointTester::testIsEquivalentTo() | All of them | Failure in comparing all of the test cases | #3, #4 |
| Edge::testEdge01() | Create and test a edge between p0 and p1, then test all characteristics of that edge | Failure in e.getSlopeZ(), slope=inf (expecting 0) | #5 |
| EdgeTester::testParallelEdges() | Writing a representative set of test cases for edges that are parallel with other | e and e1 are Reading Not Parallel: Failure to read correctly -- expected answer: Parallel | #6 |
| EdgeTester::testNonLengthEdges() | Writing a representative set of test cases for edges have a length of zero or approximately zero | The reading is incorrect. Expected length is 0. | #7 |
| TriangleTester::testFirstConstructor() | getTriangleType() | Triangle: unexpected type of I | #8 |
| TriangleTester::testFirstConstructor() | !approximatelyEquals | Triangle: unexpected area of -21.9995 | #9 |
| TriangleTester::testSecondConstructor() | !t.isTriangle() | Failure to constructor a triangle from 3 points; unexpectedly not a triangle | #10 |
| TriangleTester::testEquilateralTriangles() | getTriangleType() | Triangle: unexpected type of I | #11 |
| TriangleTester::testIsoscelesTriangles() | getTriangleType() | Point c was being checked against itself | #8 |

#1. Added #include <limits> at the beginning

#2. Changed == to != in Point::testForInfinity()

#3. Changed a > to a < symbol in Point::isEquivalentTo()

#4. Changed a getX() into a getY() in Edge::getLength()

#5. Changed a != into a == in Edge::getSlopeZ()

#6. Changed a getSlopeX() into a getSlopeZ() in Edge::isParallelTo()

#7. Changed lengths that were approximately zero to zero in Edge::getLength()

#8. Changed a “c” to and “a” in the function getTriangleType()

#9. Changed a “b” to a “c” in the function computerArea()

#10. Changed a “1” to a “2” in the Second Constructor

#11. Added approximatelyEquals(a, c, m\_edgeLengthThreshold) to allow all three sides to be checked for equality