Test Suite	Test Case	Observed Failure	Fix
PointTester::testConstructorWithDoubles()	A point with all values 0	Failure in construction point	#1
PointTester::testIsEquivalentTo()	Test the equality of p0 with all four points	Returned the wrong boolean for all points	#2
	Test the equality of p1,p2,p3 with themselves, and between p2 and p3	Points were not equal	#3
PointTester::testConstructorWithStrings()	Construct a point with a string of 3 different integer values	Point was not constructed correctly	#4
EdgeTester::testEdge01()	Test the slope of the Z value from two points	Infinity was returned instead of the expected 0	#5
EdgeTester::testEdge02()	Test the slope of the Z value from two points	Infinity was returned instead of the expected 1.118033989	#5
EdgeTester::testParallelEdges()	Test if an edge is parallel with itself	It was not parallel	#6
TriangleTester::testFirstConstructor()	Test if triangle is scalar	Test returned type ‘N’	#7
TriangleTester::testFirstConstructor()	Test triangles area	Test did not meet conditions of approximatelyEquals()	#8
TriangleTester::testSecondConstructor()	Construct a triangle from 3 points	Unexpectedly not a triangle	#9
EdgeTester::TestNonLengthEdges()	Edge with a length 0	Edge was valid	#10
	Edge with a length approx 0	Edge was valid	#10
TriangleTester::testInvalidTriangles()	Triangle with 2 points that were identical	isValid() returned true	#11

Fixes:
#1. The Point::checkForInfinity() method called in the constructor was fixed so that m_z was correctly identified as `m_z != INFINITY`

#2. The Point::isEquivalentTo return case for `edge.getLength() > m_minDistance` was correctly changed to `edge.getLength() < m_minDistance`

#3. The Edge::getLength() method was changed so that the difference between point1 y and point 2 y was correctly made: `m_point2->getY() - m_point1->getZ()` was changed to `m_point2->getY() - m_point1->getY()`

#4. Duplicate `m_y = convertStringToDouble(values[2], &m_valid)` was changed to `m_z = convertStringToDouble(values[2], &m_valid)`

#5. Changed the incorrect if condition `xyOffset != 0` to the correct `xyOffset == 0`

#6. Changed the areSlopesEquivalent function passed values in Edge::isParallelTo for `areSlopesEquivalent(getSlopeZ(), otherEdge.getSlopeX())` to `areSlopesEquivalent(getSlopeZ(), otherEdge.getSlopeZ())`

#7. In Triangle::getTriangleType(), condition to test if it was a triangle was changed correctly to `isTriangle()` from `!isTriangle()` and condition to find if two edges are the same from `approximatelyEquals(c, c, m_edgeLengthThreshold)` to `approximatelyEquals(a, c, m_edgeLengthThreshold)`

#8. Line in Triangle::computeArea() that was to calculate the semiperimeter was incorrect, changed `double s = ( a + b + b)/2` to `double s = ( a + b + c)/2`

#9. In second Triangle constructor, point[2] was assigned a repeated value values[1]; changed `m_points[2] = new Point(values[1])` to `m_points[2] = new Point(values[2])`

#10. Changed Edge::getLength() to set m_isValid to false if result <= 0.001 then updated constructor to call getLength()

#11. setupEdges() returned valid if any edge was valid instead of all of them; changed to `m_isValid = (m_edges[0]->isValid() && m_edges[1]->isValid() && m_edges[2]->isValid())`