	Input	Expected	Actual - MathPackage.java	Actual - Commit.java	Actual - Fixed.java	MathPackage.java	Commit.java	Fixed.java
randomTest:	Generate 1000 arrays using the random function	all values contained within [a,b]	all values contained within [a,b]	all values contained within [a,b]	all values contained within [a,b]	PASS	PASS	PASS
maxTest	Generate 1000 arrays, sort them and compare the last element with the return value of max	last element of array == max	last element of array == max	last element of array == max	last element of array == max	PASS	PASS	PASS
minTest	Generate 1000 arrays, sort them and compare the first element with the return value of min	first element of array == min	first element of array == min	first element of array == min	first element of array == min	PASS	PASS	PASS
normalizeTest	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]	[0, -0.1, -0.2, -0.3, -0.4, -0.5, -0.6, -0.7, -0.8, -0.9, -1.0]	[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]	[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]	FAIL	PASS	PASS
	[0, 33, 66, 100]	[0, 0.33, 0.66, 1.0]	[0, -0.33, -0.66, -1.0]	[0, 0.33, 0.66, 1.0]	[0, 0.33, 0.66, 1.0]	FAIL	PASS	PASS
sumTest	Generate 1000 arrays, compare the sum against Java's built in method for calculating sum	sum	sum	sum	sum	PASS	PASS	PASS
stddev	[0.8448535275473217, 0.7356655820407797, 0.9431584337138527, 0.9885039386933584, 0.26752295318703023, 0.30118478380150293, 0.5614795756611817, 0.27410185661451103, 0.3727308526619961, 0.9350197461150006, 0.3430515987014403, 0.14325655442623153, 0.09474732578722389, 0.8118925398379901, 0.7889524243808093, 0.6174482765472112, 0.08282552694523304, 0.9906861613488818, 0.1006718689797883, 0.07076982982755198]	0.33383429942515 +/- 1e-10	0.33383429942515 +/- 1e-11	0.33383429942515 +/- 1e-12	0.33383429942515 +/- 1e-13	FAIL	FAIL	FAIL
	[24.351879794282766, -1.2369080960006045, 21.33618933269294, 38.40700888177969, 81.07224984081901, 14.773395426428706, -23.465737641671126, -29.74766797865162, -80.53273274920562, 82.69835022556401, 16.765612733609586, 85.27325420746189, -25.56842729615137, -85.17577250342885, 91.14866661682103, 47.56747687575623, 71.62517118559902, 26.263337092658247, -78.61956099645082, -18.809917650844213, 31.415431556119216, 15.021802528535659, -45.50777659561043, 76.15463305868838, 71.73129701754698, 42.00719896644702, 98.93219072662015, 29.911361379866946, -7.304486822031862, 72.46129116466469, 25.17287827553399, 6.610331737331364, 34.267027445622745, 10.796643684552663, -44.6877977606748, 99.08890435845663, 18.379318834519594, 54.01219466707613, -11.329038630356479, -23.230019858073646, 33.03020694951988, -59.205375268977974, -8.495912401158435, 83.71104890003406, 80.19371195739095, 70.17088288687546, -27.756415190854483, 67.22645204084466, 2.237307845125258, 90.69398038338696, 13.575182881518089, 66.0876889384611, -30.1366615029317, -20.084766272685243, 10.62213430500914, 70.12939720918465, -53.19292944066913, -84.43071331549132, -53.30861375844398, 36.75691360838863, 4.481177488201666, 89.23026437856967, 28.481208229172694, 4.041593290473799, -19.144188062781325, 71.48911820023875, -97.3407210733165, -82.95697886617639, -52.019808046281035, -20.500795201979386, -65.75240723052059, -76.4564498827019, -48.255740133965006, 20.75249609534157, -45.919570907416606, 55.70989509490502, -30.432532466301396, -51.16117165251921, -7.213072379759879, 62.480197271840865, -14.266598956238383, -14.496858737726953, -6.199152015512155, 86.73375309074737, 24.576298742407715, -50.335938636762336, -98.21921344917936, -97.29469852961215, 92.13635661651074, -6.363254915974139, 34.622868801612384, -47.36683311550864, -57.59764296582546, 53.28248250936096, 26.007936605144735, -86.505212849109, -7.68902146399742, -18.99542397736265, -5.991562076158388, 82.91883854718108]	54.00891607	54.00891607	54.00891607	54.00891607	PASS	PASS	PASS
arrayAddTestSameLength	Generate 1000 * 2 arrays with the same length. Compare the result against an alternative way to sum the arrays using Java 8 lambdas	element-wise sum	element-wise sum	element-wise sum	element-wise sum	PASS	PASS	PASS
arrayAddDifferentLength	d1: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1] d2: [1, 2, 3, 4, 5, 6]	Assertion error different lengths	[11, 11, 11, 11, 11, 11]	[11, 11, 11, 11, 11, 11]	Assertion error different lengths	FAIL	FAIL	PASS
	d1: [1, 2, 3, 4, 5, 6] d2: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]	Assertion error different lengths	IndexOutOfBoundsException	IndexOutOfBoundsException	Assertion error different lengths	FAIL	FAIL	PASS
negateTest	Generate 1000 arrays, and negate them, compare against return result	array negation	array negation	array negation, but 1 is also subtracted from each element	array negation	PASS	FAIL	PASS

arraySubtractTestSameLength	Generate 1000 * 2 arrays with the same length. Compare the result against an alternative way to subtract the arrays using Java 8 lambdas	element-wise subtraction	n/a	element-wise subtraction	element-wise subtraction		PASS	PASS
arraySubtractTestDifferentLength	d1: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1] d2: [1, 2, 3, 4, 5, 6]	Assertion error different lengths	n/a	[-9, -7, -5, -3, -1, 1]	Assertion error different lengths		FAIL	PASS
	d1: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1] d2: [1, 2, 3, 4, 5, 6]	Assertion error different lengths	n/a	[9, 7, 5, 3, 1, -1]	Assertion error different lengths		FAIL	PASS
distanceTest	d1 or d2 are not of length 2	Assertion error different lengths	n/a	0	Assertion error different lengths		FAIL	PASS
	d1: [-1, -2] d2: [3, 4]	7.211102551	n/a	0	7.211102551		FAIL	PASS
	d1: [6, 7] d2: [-8,-9]	21.26029163	n/a	0	21.26029163		FAIL	PASS
arrayDeviationTest	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	[-4.5, -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, 4.5]	n/a	null	[-4.5, -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, 4.5]		FAIL	PASS
	[4, 8, 1, 3, 9, 5, 10, 2, 7, 6]	[-1.5, 2.5, -4.5, -2.5, 3.5, -0.5, 4.5, -3.5, 1.5, 0.5]	n/a	null	[-1.5, 2.5, -4.5, -2.5, 3.5, -0.5, 4.5, -3.5, 1.5, 0.5]		FAIL	PASS

	input	expected	actual	
reverseTest	[1, 2, 3, 4, 5, 6, 7, 8, 9]	[9, 8, 7, 6, 5, 4, 3, 2, 1]	[9, 8, 7, 6, 5, 4, 3, 2, 1]	PASS
uniqueTest	[1, 1, 2, 3, 4, 5, 6, 7, 8, 9, null]	[1, 2, 3, 4, 5, 6, 7, 8, 9]	[1, 2, 3, 4, 5, 6, 7, 8, 9]	PASS
intersectionTest	a: [1, 2, 3, 4, 5, 6, 7, 8, 9] b: [3, 4, 5]	[3, 4, 5]	[3, 4, 5]	PASS
unionTest	a: [1, 2, 3, 4] b: [5, 6, 7, 8, 9]	[1, 2, 3, 4, 5, 6, 7, 8, 9]	[1, 2, 3, 4, 5, 6, 7, 8, 9]	PASS
indexOfTest	a: [1, 2, 3, 4, 5, 6, 7, 8, 9, null] b: 5	4	4	PASS
	a: [1, 2, 3, 4, 5, 6, 7, 8, 9, null] b: null	10	10	PASS
withoutTest	array: [1,2,3,4, 4,5,6,7,8,9], remove: [3, 4, 5, 10]	[1, 2, 6, 7, 8, 9]	[1, 2, 6, 7, 8, 9]	PASS
withoutTestRemoveTwo	array: [1,2,3,4,5,6,7,8,9], remove: [3, 4, 4, 5, 10]	[1, 2, 6, 7, 8, 9]	[1, 2, 4, 6, 7, 8, 9]	FAIL
withoutTestRemoveFirstElement	array: [1,2,3,4,5,6,7,8,9], remove: [1, 3, 4, 5]	[2, 6, 7, 8, 9]	[1, 2, 6, 7, 8, 9]	FAIL
intersectionTestDuplicate	a: [1, 2, 2, 3, 4] b: [2, 2, 3, 4]	[2, 2, 3, 4]	IndexOutOfBoundsException	FAIL