

CS731 SOFTWARE TESTING

PROJECT REPORT

Topic: Design Integration Testing on a Scientific Calculator

Team Member:

Manish Reddy Koppula IMT2020019

Deep Shashank Pani IMT2020129

Introduction

Software testing is a method of determining whether the actual software product meets the expected requirements and ensuring that the software product is free of defects. It entails running software/system components through their paces using manual or automated tools to evaluate one or more properties of interest.

We have chosen the Design Integration Testing method for testing our code. We have used the JUnit tool for testing.

Contribution

Manish Reddy - Documentation, Making the Source code, Identifying all the call interfaces

Deep Shashank - Making Test Cases, Making the code testable, Identifying the Coupling pairs

Source Code Description

- ❖ The Source Code performs a plethora of Mathematical operations (more than 50 different types) and is akin to a Scientific Calculator.
- ❖ Some of those operations are:

I. Addition	IX. Tan
II. Subtraction	X. Secant
III. Multiplication	XI. Cosecant
IV. Division	XII. Cotangent
V. Square Root	XIII. Logarithm(Base 10)
VI. Power	XIV. Natural Logarithm
VII. Sine	XV. Logarithm (Base 2)
VIII. Cos	XVI. Absolute Value

In our code we have 5 main interfaces which handle all the operations:

- performUnaryOperation(): This handle all the operations that require just one input, eg. Square Root, Logarithm(Base 10) ,etc.
- performBinaryOperation(): This handle all the operations that require two input, eg. Addition, Subtraction,etc.
- performTernaryOperation(): This handle all the operations that require three input, eg. Power of Sum ($(a+b)^c$) ,etc.

- performTrigonometricOperation(): This handle all the operations that deal with the normal Trigonometric Operations , eg. Sine, Cos, etc.
- performInvTrigonometricOperation(): This handle all the operations that deal with Inverse Trigonometric Operations , eg. arcsin, arccos, etc.

These Main interfaces are in ScientificCalculator.java and they call the operations that are in Operations.java as and when required.

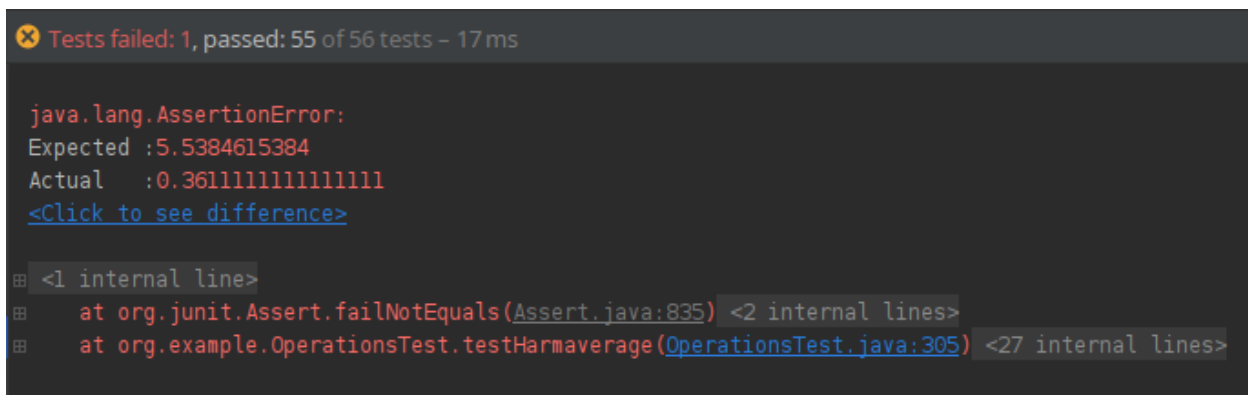
Design Integration Testing

Design Integration Testing is a comprehensive white-box testing approach that aims to assess the seamless integration and interaction of individual design components within a software system. This technique focuses on evaluating how these integrated components collectively function according to the specified design architecture. Unlike unit testing, which primarily validates individual units in isolation, Design Integration Testing addresses the collaboration and cooperation among these units when integrated into the larger system.

In our approach we find all the calls possible between the modules during execution and we note down the corresponding “Last-Def & First-Use” Coupling pairs in the code. And then we used those pairs as reference to create test cases that follow the du-paths that correspond to the Coupling pairs.

For the testing we used Bottom-Up Integration Testing, we first tested all the Operations we had in Operations.java.

Here we find a bug in one of the operations.



```

✖ Tests failed: 1, passed: 55 of 56 tests - 17 ms

java.lang.AssertionError:
Expected :5.5384615384
Actual   :0.3611111111111111
<Click to see difference>

<1 internal line>
at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>
at org.example.OperationsTest.testHarmaverage(OperationsTest.java:305) <27 internal lines>

```

This bug were caused due to faulty implementation of the operation itself, since the operation tester driving returned a failure, the tester program does not reach the main tester driver. We debug the operation accordingly, and re-run the testers.

Then we tested each of the 5 Main interfaces mentioned before after integrating them with the corresponding Operation after that we Integrated all the interfaces to the main() and did a full scale testing on it.

```

✓ Tests passed: 56 of 56 tests - 8 ms
/usr/lib/jvm/java-11-openjdk/bin/jav
Process finished with exit code 0

```

✓ testAdd()		Performing Addition: 2.0 + 3.0
✓ testCos()		Performing Cosine Calculation: cos(60.0 degrees)
✓ testCot()	15 ms	Performing Tangent Calculation: tan(45.0 degrees)
✓ testExp()		Performing Exponential Calculation: e^2.0
✓ testLog()		Performing Logarithm Calculation: ln(2.718281828459045)
✓ testSec()		Performing Sine Calculation: sin(60.0 degrees)
✓ testTan()		Performing Tangent Calculation: tan(45.0 degrees)
✓ testSubtract()		Performing Subtraction: 5.0 - 3.0
✓ testCosec()		Performing Cosine Calculation: cos(30.0 degrees)
✓ testFloor()		Performing Floor Calculation: floor(3.6)
✓ testLog10()		Performing Logarithm (Base 10) Calculation: log10(100.0)
✓ testPower()	16 ms	Performing Power Calculation: 2.0 ^ 3.0
✓ testRound()		Performing Round Calculation: round(3.5)
✓ testCeil()		Performing Ceiling Calculation: ceil(3.4)
✓ testCosh()		Performing Hyperbolic Cosine Calculation: cosh(0.5)
✓ testCube()		Performing Cube Calculation: 3.0 ^ 3
✓ testLog2()		Performing Logarithm (Base 10) Calculation: log10(8.0)
✓ testSine()		Performing Sine Calculation: sin(30.0 degrees)
✓ testSinh()		Performing Hyperbolic Sine Calculation: sinh(0.5)
✓ testSqrt()		Performing Square Root Calculation: sqrt(9.0)
✓ testTanh()		Performing Hyperbolic Tangent Calculation: tanh(0.5)
✓ testAverage()	15 ms	Performing Average Calculation: (5.0 + 3.0) / 2
✓ testFactorial()		
✓ testPowerOfDiff()		
✓ testPowerOfSum()		

The above the individual testing of all the operations in Operations.java.

```

BinaryTest
  Test Results: 80 ms
  BinaryTest: 80 ms
    testPerformBinaryOperation_Percentage: 49 ms
    testPerformBinaryOperation_HarmonicMean: 49 ms
    testPerformBinaryOperation_Division: 49 ms
    testPerformBinaryOperation_Maximum: 49 ms
    testPerformBinaryOperation_Subtraction: 16 ms
    testPerformBinaryOperation_Minimum: 16 ms
    testPerformBinaryOperation_Modulus: 16 ms
    testPerformBinaryOperation_Power: 16 ms
    testPerformBinaryOperation_Permutation: 16 ms
    testPerformBinaryOperation_Multiplication: 16 ms
    testPerformBinaryOperation_GeoAverage: 16 ms
    testPerformBinaryOperation_Combination: 15 ms
    testPerformBinaryOperation_SumOfFactorials: 15 ms
    testPerformBinaryOperation_LogCustom: 15 ms
    testPerformBinaryOperation_Addition: 15 ms
    testPerformBinaryOperation_Average: 15 ms
  
```

Enter first number: Enter second number: Performing Percentage(b is % of a) Calculation: (20.0 / 50.0) * 100
Result: 40.0

Enter first number: Enter second number: Performing Harmonic Mean Calculation: (2.0 + 8.0) / 2
Result: 3.2

Enter first number: Enter second number: Performing Division: 10.0 / 2.0
Result: 5.0

Enter first number: Enter second number: Performing Maximum Calculation: max(5.0, 3.0)
Result: 5.0

Enter first number: Enter second number: Performing Subtraction: 5.0 - 3.0
Result: 2.0

Enter first number: Enter second number: Performing Minimum Calculation: min(5.0, 3.0)
Result: 3.0

Enter first number: Enter second number: Performing Modulus Calculation: 10.0 % 3.0
Result: 1.0

Enter first number: Enter second number: Performing Power Calculation: 2.0 ^ 3.0
Result: 8.0

Enter first number: Enter second number: Performing Permutation Calculation: P(5.0, 2.0)
Performing Factorial Calculation: 5.0!
Performing Factorial Calculation: 5!
Performing Factorial Calculation: 3.0!

The above is the testing done for all the Binary operations after integrating them with performBinaryOperation().

```

Binary_Unary_Test
  Test Results: 159 ms
  Binary_Unary_Test: 159 ms
    testPerformUnaryOperation_Inverse: 46 ms
    testPerformUnaryOperation_Factorial: 46 ms
    testPerformBinaryOperation_Percentage: 13 ms
    testPerformBinaryOperation_Average: 13 ms
    testPerformBinaryOperation_Division: 3 ms
    testPerformBinaryOperation_Maximum: 3 ms
    testPerformUnaryOperation_AbsPower: 16 ms
    testPerformBinaryOperation_Subtraction: 16 ms
    testPerformUnaryOperation_SquareRoot: 16 ms
    testPerformBinaryOperation_Minimum: 16 ms
    testPerformBinaryOperation_Modulus: 16 ms
    testPerformBinaryOperation_Power: 16 ms
    testPerformBinaryOperation_Percentage: 15 ms
    testPerformBinaryOperation_Multiplication: 15 ms
    testPerformBinaryOperation_GeoAverage: 15 ms
    testPerformUnaryOperation_LogCustom: 16 ms
    testPerformUnaryOperation_Deg: 16 ms
    testPerformBinaryOperation_Combination: 16 ms
    testPerformBinaryOperation_SumOfFactorials: 16 ms
    testPerformUnaryOperation_Abs: 16 ms
    testPerformUnaryOperation_Exp: 16 ms
    testPerformUnaryOperation_Log: 16 ms
    testPerformUnaryOperation_RadiansToDegrees: 16 ms
    testPerformBinaryOperation_LogCustom: 16 ms
    testPerformBinaryOperation_Addition: 16 ms
    testPerformUnaryOperation_Floor: 16 ms
    testPerformUnaryOperation_Log: 16 ms
    testPerformBinaryOperation_Average: 16 ms
    testPerformUnaryOperation_Round: 16 ms
  
```

Enter a number: Performing Logarithm (Base 10) Calculation: log10(100.0)
Result: 2.0

Enter first number: Enter second number: Performing Average Calculation: (2.0 + 4.0) / 2
Result: 3.0

Enter a number: Performing Round Calculation: round(7.5)
Result: 8.0

Enter a number: Performing Ceiling Calculation: ceil(7.4)
Result: 8.0

Enter a number: Performing Hyperbolic Cosine Calculation: cosh(1.0)
Result: 1.543080634815244

Enter a number: Performing Logarithm (Base 10) Calculation: log10(8.0)
Result: 3.0

Enter a number: Performing Hyperbolic Sine Calculation: sinh(1.0)
Result: 1.1752011936438014

Enter a number: Performing Hyperbolic Tangent Calculation: tanh(1.0)
Result: 0.7615941559557649

Process finished with exit code 0

This image is for the testing done after integrating both Binary and unary Operations with performUnaryOperation() and performBinaryOperation().

Then we integrated performTernaryOperation(), performTrigonometricOperation() and performInvTrigonometricOperation() successively.

Then we did the final integration with the main file ScientificCalculator.java. This gave a few errors.

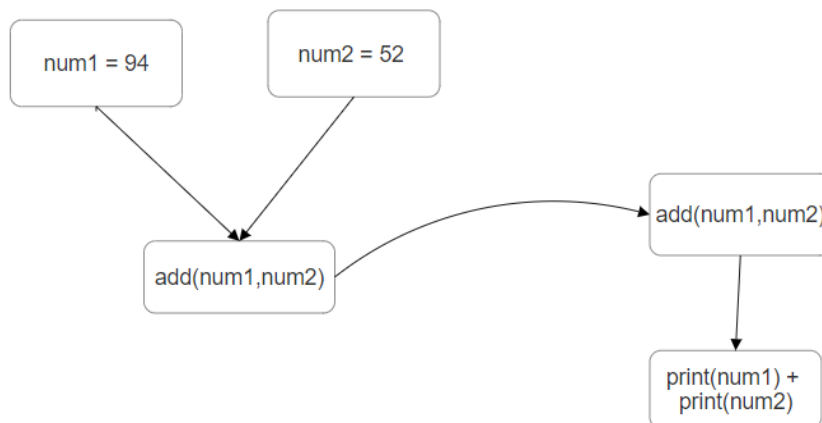
```
Testcase 31 failed
java.lang.AssertionError: Create breakpoint : expected:<16.0> but was:<5.333333333333333> <1 internal line>
    at org.junit.Assert.failNotEquals(Assert.java:834) <2 internal lines>
    at org.example.testcalc.testing(testcalc.java:139) <30 internal lines>
    at org.example.testrun.main(testrun.java:9)
Testcase 32 failed
java.lang.AssertionError: Create breakpoint : expected:<4.0> but was:<1.3333333333333333> <1 internal line>
    at org.junit.Assert.failNotEquals(Assert.java:834) <2 internal lines>
    at org.example.testcalc.testing(testcalc.java:139) <30 internal lines>
    at org.example.testrun.main(testrun.java:9)
```

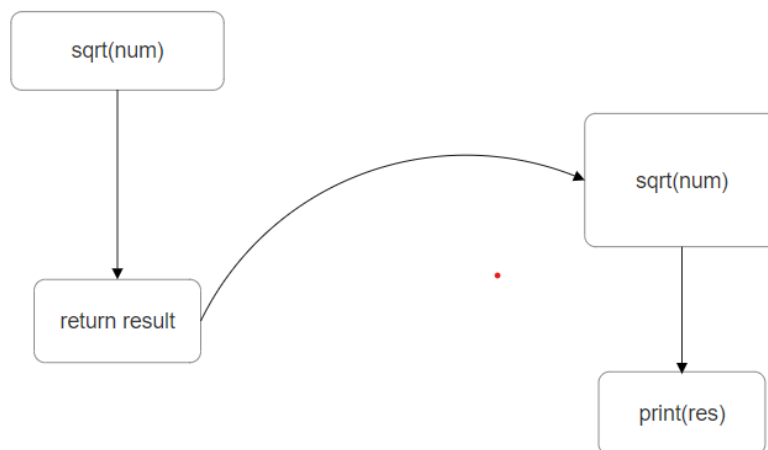
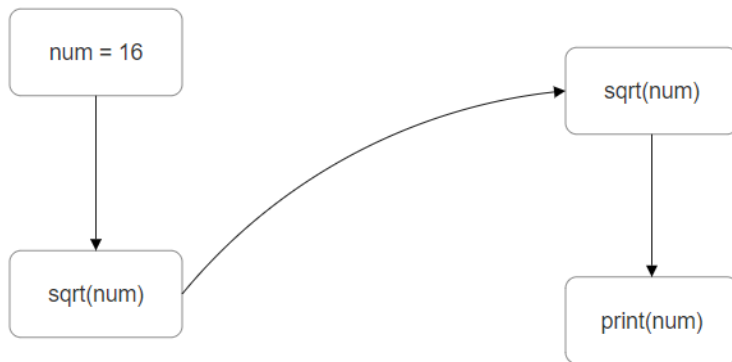
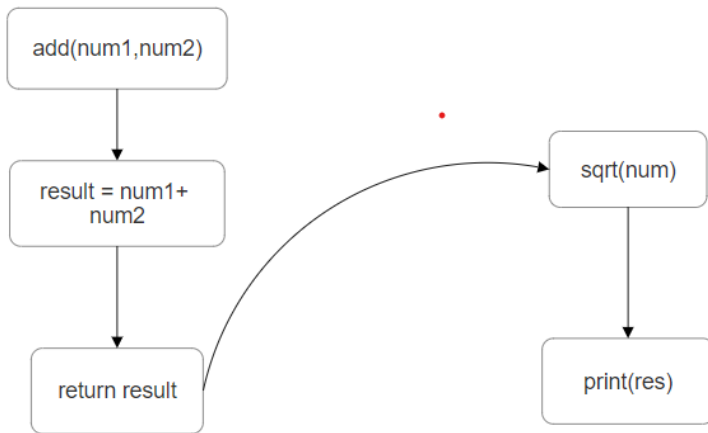
We found out that two test cases fail in the main tester driver but pass in the operation tester, this indicates an integration failure. After debugging the integrator function, we again run both testers and succeed this time.

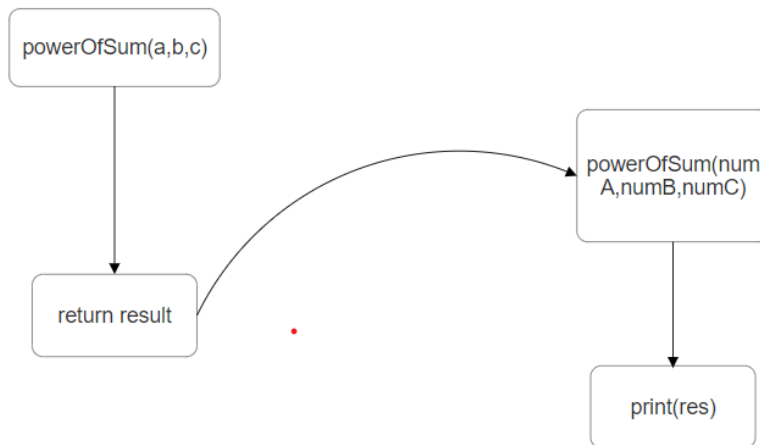
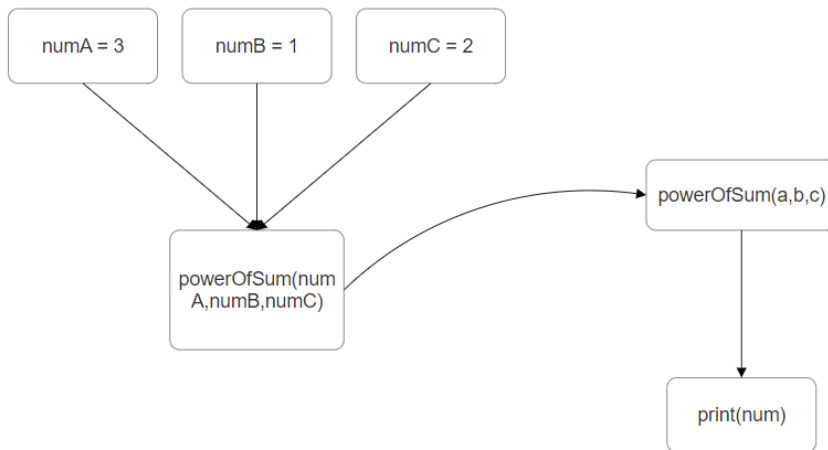
```
✓ Tests passed: 56 of 56 tests – 8 ms
/usr/lib/jvm/java-11-openjdk/bin/jav
Process finished with exit code 0
```

The above shows that 56 Test cases were successfully passed by the Scientific Calculator Program, showing no integration failures.

Coupling variables







Shared variables:

The below are all the instances of Last defs and first uses in the code for the relevant, i.e, shared variables.

Last defs:

In ScientificCalculator.java:

- Line 10,121: `angmode`
- Line 248: `num`
- Line 231,234: `num1,num2`
- Line 346,349,352: `numa,numb,numc`

In Operations.java:

- Line 269,272: res in factorialint()

Result defs:

- Line 9: result in add()
- Line 16: result in subtract()
- Line 23: result in multiply()
- Line 31: result in divide()
- Line 42: result in power()
- Line 178: result in sin()
- Line 185: result in cos()
- Line 192: result in tan()
- Line 199: result in sec()
- Line 206: result in cosec()
- Line 213: result in cot()
- Line 69: result in log10()
- Line 93: result in log()
- Line 81: result in log2()
- Line 104: result in abs()
- Line 111: result in ceil()
- Line 118: result in floor()
- Line 125: result in round()
- Line 133: result in logCustomBase()
- Line 145: result in factorial()
- Line 272: res in factorial(int)
- Line 156: result in exp()
- Line 163: result in degreesToRadians()
- Line 170: result in radiansToDegrees()
- Line 322: result in powerOfSum()
- Line 335: result in powerOfDiff()
- Line 345: result in absPower()
- Line 457: result in modulus()
- Line 364: result in square()
- Line 371: result in cube()
- Line 468: result in minimum()
- Line 475: result in maximum()
- Line 296: result in combination()
- Line 310: result in permutation()
- Line 353: result in inverse()
- Line 378: result in arcsin()
- Line 385: result in arcsinrad()
- Line 392: result in arccos()
- Line 399: result in arccosrad()

- Line 406: result in arctan()
- Line 413: result in arctanrad()
- Line 420: result in sinh()
- Line 427: result in cosh()
- Line 434: result in tanh()
- Line 441: result in roundUp()
- Line 448: result in roundDown()
- Line 482: result in average()
- Line 489: result in geoaverage()
- Line 500: result in harmaverage()
- Line 510: result in percentage()
- Line 521: result in harmnum()
- Line 534: result in sumofn()
- Line 547: result in sum2ofn()
- Line 560: result in sum3ofn()
- Line 574,577: result in sumkofn()

First uses:

In ScientificCalculator.java:

- Line 237: res in performBinaryOperation()
- Line 251: res in performUnaryOperation()
- Line 260: angmode in performTrigonometricOperation()
- Line 312: angmode in performInverseTrigonometricOperation()
- Line 355: res in performTernaryOperation()

In Operations.java:

- Line 7: num1, num2
- Line 15: num1, num2
- Line 22: num1, num2
- Line 29: num1, num2
- Line 41: base (num)
- Line 41: exponent (num)
- Line 177: angle (num)
- Line 184: angle (num)
- Line 191: angle (num)
- Line 198: angle(num)
- Line 205: angle (num)
- Line 212: angle (num)
- Line 67: num
- Line 91: num

- Line 79: num
- Line 103: num
- Line 110: num
- Line 117: num
- Line 124: num
- Line 131: num
- Line 131: base (num)
- Line 143: num in factorial()
- Line 262: n in factorial(int)
- Line 155: num
- Line 162: degrees (num)
- Line 169: radians (num)
- Line 321: a,b,c (numA,numB,numC)
- Line 331: a,b,c (numA,numB,numC)
- Line 344: base (num)
- Line 455: num1,num2
- Line 363: num
- Line 370: num
- Line 467: num1,num2
- Line 474: num1, num2
- Line 290: n,r (num1,num2)
- Line 296: a,b,c (res,res,res)
- Line 306: n,r (num1,num2)
- Line 310: a,b (res,res)
- Line 351: num (num)
- Line 377: num (num)
- Line 384: num (num)
- Line 391: num (num)
- Line 398: num (num)
- Line 405: num (num)
- Line 412: num (num)
- Line 419: num (num)
- Line 426: num (num)
- Line 433: num (num)
- Line 440: num (num)
- Line 447: num (num)
- Line 481: num1,num2 (num1,num2)
- Line 488: num1,num2 (num1,num2)
- Line 499: num1,num2 (num1,num2)
- Line 509: num1,num2 (num1,num2)
- Line 516: num (num)

- Line 532: num (num)
- Line 545: num (num)
- Line 558: num (num)
- Line 571: num,k (num1,num2)

Du Pairs:

The following are the du pairs for the above.

1. (ScientificCalculator::main(),angmode,10)
-(ScientificCalculator::performTrigonometricOperation(),angmode,260)
2. (ScientificCalculator::main(),angmode,10)
-(ScientificCalculator::performInverseTrigonometricOperation(),angmode,312)
3. (ScientificCalculator::main(),angmode,121)
-(ScientificCalculator::performTrigonometricOperation(),angmode,260)
4. (ScientificCalculator::main(),angmode,121)
-(ScientificCalculator::performInverseTrigonometricOperation(),angmode,312)

Unary operations: num:

5. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sqrt(), num, 54)
6. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sin(), angle, 177)
7. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cos(), angle, 184)
8. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::tan(), angle, 191)
9. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sec(), angle, 198)
10. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cosec(), angle, 205)
11. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cot(), angle, 212)
12. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sinrad(), angle, 219)
13. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cosrad(), angle, 226)
14. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::tanrad(), angle, 233)
15. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::secrad(), angle, 240)
16. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cosecrad(), angle, 247)
17. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cotrad(), angle, 254)
18. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::log10(), num, 67)
19. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::log(), num, 91)
20. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::log2(), num, 79)
21. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::abs(), num, 103)
22. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::ceil(), num, 110)
23. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::floor(), num, 117)
24. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::round(), num, 124)
25. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::factorial(), num, 143)
26. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::exp(), num, 155)
27. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::degreesToRadians(),
degrees, 162)
28. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::radiansToDegrees(),
radians, 169)
29. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::absPower(), base, 344)

30. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::square(), num, 363)
31. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cube(), num, 370)
32. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::inverse(), num, 351)
33. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::arcsin(), num, 377)
34. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::arcsinrad(), num, 384)
35. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::arccos(), num, 391)
36. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::arccosrad(), num, 398)
37. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::arctan(), num, 405)
38. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::arctanrad(), num, 412)
39. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sinh(), num, 419)
40. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::cosh(), num, 426)
41. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::tanh(), num, 433)
42. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::roundUp(), num, 440)
43. (ScientificCalculator::performUnaryOperation(), num, 248)
— (Operations::roundDown(), num, 447)
44. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::harmnum(), num, 516)
45. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sumofn(), num, 532)
46. (ScientificCalculator::performUnaryOperation(), num, 248)
— (Operations::sum2ofn(), num, 545)
47. (ScientificCalculator::performUnaryOperation(), num, 248) — (Operations::sum3ofn(), num, 558)

Binary operations: num1,num2:

48. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::add(), num1, 7)
49. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::add(), num2, 7)
50. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::subtract(), num1, 15)
51. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::subtract(), num2, 15)
52. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::multiply(), num1, 22)
53. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::multiply(), num2, 22)
54. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::divide(), num1, 29)
55. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::divide(), num2, 29)
56. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::power(), base, 41)
57. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::power(), exponent, 41)
58. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::logCustomBase(), num, 131)
59. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::logCustomBase(), base, 131)
60. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::modulus(), num1, 455)
61. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::modulus(), num2, 455)
62. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::minimum(), num1, 467)
63. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::minimum(), num2, 467)

64. (ScientificCalculator::performBinaryOperation(), num1, 231) — (Operations::maximum(), num1, 474)
65. (ScientificCalculator::performBinaryOperation(), num2, 234) — (Operations::maximum(), num2, 474)
66. (ScientificCalculator::performBinaryOperation(),num1,231)
-(Operations::combination(),n,290)
67. (ScientificCalculator::performBinaryOperation(),num1,231) -(Operations::permutation(),n,306)
68. (ScientificCalculator::performBinaryOperation(),num1,231)
-(Operations::average(),num1,481)
69. (ScientificCalculator::performBinaryOperation(),num1,231)
-(Operations::geoaverage(),num1,488)
70. (ScientificCalculator::performBinaryOperation(),num1,231)
-(Operations::harmaverage(),num1,499)
71. (ScientificCalculator::performBinaryOperation(),num1,231)
-(Operations::percentage(),num1,509)
72. (ScientificCalculator::performBinaryOperation(),num1,231) -(Operations::sumkofn(),num,571)
73. (ScientificCalculator::performBinaryOperation(),num2,234) -(Operations::combination(),r,290)
74. (ScientificCalculator::performBinaryOperation(),num2,234) -(Operations::permutation(),r,306)
75. (ScientificCalculator::performBinaryOperation(),num2,234)
-(Operations::average(),num2,481)
76. (ScientificCalculator::performBinaryOperation(),num2,234)
-(Operations::geoaverage(),num2,488)
77. (ScientificCalculator::performBinaryOperation(),num2,234)
-(Operations::harmaverage(),num2,499)
78. (ScientificCalculator::performBinaryOperation(),num2,234)
-(Operations::percentage(),num2,509)
79. (ScientificCalculator::performBinaryOperation(),num2,234) -(Operations::sumkofn(),k,571)

Ternary operations: numa,numb,numc:

80. (ScientificCalculator::performTernaryOperation(), numa, 346) — (Operations::powerOfSum(), a, 321)
81. (ScientificCalculator::performTernaryOperation(), numB, 349) — (Operations::powerOfSum(), b, 321)
82. (ScientificCalculator::performTernaryOperation(), numC, 352) — (Operations::powerOfSum(), c, 321)
83. (ScientificCalculator::performTernaryOperation(), numa, 346) — (Operations::powerOfDiff(), a, 331)
84. (ScientificCalculator::performTernaryOperation(), numB, 349) — (Operations::powerOfDiff(), b, 331)
85. (ScientificCalculator::performTernaryOperation(), numC, 352) — (Operations::powerOfDiff(), c, 331)

Factorial Using functions:

86. (Operations::factorial(),res,145) -(Operations::combination(),a,296)
87. (Operations::factorial(),res,145) -(Operations::combination(),b,296)
88. (Operations::factorial(),res,145) -(Operations::combination(),c,296)

89. (Operations::factorial(),res,145) -(Operations::permutation(),a,310)

90. (Operations::factorial(),res,145) -(Operations::permutation(),b,310)

Operations: result to the required one:

91. (Operations::add, result, 9) — (ScientificCalculator::performBinaryOperation(), res, 237)

92. (Operations::subtract, result, 16) — (ScientificCalculator::performBinaryOperation(), res, 237)

93. (Operations::multiply, result, 23) — (ScientificCalculator::performBinaryOperation(), res, 237)

94. (Operations::divide, result, 31) — (ScientificCalculator::performBinaryOperation(), res, 237)

95. (Operations::sqrt, result, 57) — (ScientificCalculator::performUnaryOperation(), res, 254)

96. (Operations::power(), result, 42) — (ScientificCalculator::performBinaryOperation(), res, 237)

97. (Operations::sin(), result, 178) — (ScientificCalculator::performUnaryOperation(), res, 254)

98. (Operations::cos(), result, 185) — (ScientificCalculator::performUnaryOperation(), res, 254)

99. (Operations::tan(), result, 192) — (ScientificCalculator::performUnaryOperation(), res, 254)

100. (Operations::sec(), result, 199) — (ScientificCalculator::performUnaryOperation(), res, 254)

101. (Operations::cosec(), result, 206) — (ScientificCalculator::performUnaryOperation(), res, 254)

102. (Operations::cot(), result, 213) — (ScientificCalculator::performUnaryOperation(), res, 254)

103. (Operations::sinrad(), result, 220) — (ScientificCalculator::performUnaryOperation(), res, 254)

104. (Operations::cosrad(), result, 227) — (ScientificCalculator::performUnaryOperation(), res, 254)

105. (Operations::tanrad(), result, 234) — (ScientificCalculator::performUnaryOperation(), res, 254)

106. (Operations::secrad(), result, 241) — (ScientificCalculator::performUnaryOperation(), res, 254)

107. (Operations::cosecrad(), result, 248) — (ScientificCalculator::performUnaryOperation(), res, 254)

108. (Operations::cotrad(), result, 255) — (ScientificCalculator::performUnaryOperation(), res, 254)

109. (Operations::log10(), result, 69) — (ScientificCalculator::performUnaryOperation(), res, 254)

110. (Operations::log(), result, 93) — (ScientificCalculator::performUnaryOperation(), res, 254)

111. (Operations::log2(), result, 81) — (ScientificCalculator::performUnaryOperation(), res, 254)

112. (Operations::abs(), result, 104) — (ScientificCalculator::performUnaryOperation(), res, 254)

113. (Operations::ceil(), result, 111) — (ScientificCalculator::performUnaryOperation(), res, 254)

114. (Operations::floor(), result, 118) — (ScientificCalculator::performUnaryOperation(), res, 254)

115. (Operations::round(), result, 125) — (ScientificCalculator::performUnaryOperation(), res, 254)

116. (Operations::logCustomBase(), result, 133) — (ScientificCalculator::performBinaryOperation(), res, 237)

117. (Operations::factorial(), result, 145) — (ScientificCalculator::performUnaryOperation(), res, 254)

118. (Operations::exp(), result, 156) — (ScientificCalculator::performUnaryOperation(), res, 254)

119. (Operations::degreesToRadians(), result, 163) —
(ScientificCalculator::performUnaryOperation(), res, 254)

120. (Operations::radiansToDegrees(), result, 170) —
(ScientificCalculator::performUnaryOperation(), res, 254)

121. (Operations::powerOfSum(), result, 322) — (ScientificCalculator::performTernaryOperation, res, 355)

122. (Operations::powerOfDiff(), result, 335) — (ScientificCalculator::performTernaryOperation(), res, 355)

123. (Operations::absPower, result, 345) — (ScientificCalculator::performUnaryOperation(), res, 254)

124. (Operations::modulus(), result, 457) — (ScientificCalculator::performBinaryOperation(), res, 237)
125. (Operations::square(), result, 364) — (ScientificCalculator::performUnaryOperation(), res, 254)
126. (Operations::cube(), result, 371) — (ScientificCalculator::performUnaryOperation(), res, 254)
127. (Operations::minimum(), result, 468) — (ScientificCalculator::performBinaryOperation(), res, 237)
128. (Operations::maximum(), result, 475) — (ScientificCalculator::performBinaryOperation(), res, 237)
129. (Operations::combination(),result,296)
-(ScientificCalculator::performBinaryOperation(),res,237)
130. (Operations::permutation(),result,310)
-(ScientificCalculator::performBinaryOperation(),res,237)
131. (Operations::inverse(),result,353)
-(ScientificCalculator::performUnaryOperation(),res,251)
132. (Operations::arcsin(),result,378) -(ScientificCalculator::performUnaryOperation(),res,251)
133. (Operations::arcsinrad(),result,385)
-(ScientificCalculator::performUnaryOperation(),res,251)
134. (Operations::arccos(),result,392) -(ScientificCalculator::performUnaryOperation(),res,251)
135. (Operations::arccosrad(),result,399)
-(ScientificCalculator::performUnaryOperation(),res,251)
136. (Operations::arctan(),result,406) -(ScientificCalculator::performUnaryOperation(),res,251)
137. (Operations::arctanrad(),result,413)
-(ScientificCalculator::performUnaryOperation(),res,251)
138. (Operations::sinh(),result,420) -(ScientificCalculator::performUnaryOperation(),res,251)
139. (Operations::cosh(),result,428) -(ScientificCalculator::performUnaryOperation(),res,251)
140. (Operations::tanh(),result,435) -(ScientificCalculator::performUnaryOperation(),res,251)
141. (Operations::roundUp(),result,441)
-(ScientificCalculator::performUnaryOperation(),res,251)
142. (Operations::roundDown(),result,448)
-(ScientificCalculator::performUnaryOperation(),res,251)
143. (Operations::average(),result,482)
-(ScientificCalculator::performBinaryOperation(),res,237)
144. (Operations::geoaverage(),result,489)
-(ScientificCalculator::performBinaryOperation(),res,237)
145. (Operations::harmaverage(),result,500)
-(ScientificCalculator::performBinaryOperation(),res,237)
146. (Operations::percentage(),result,510)
-(ScientificCalculator::performBinaryOperation(),res,237)
147. (Operations::harmnum(),result,521)
-(ScientificCalculator::performUnaryOperation(),res,251)
148. (Operations::sumofn(),result,534)
-(ScientificCalculator::performUnaryOperation(),res,251)
149. (Operations::sum2ofn(),result,547)
-(ScientificCalculator::performUnaryOperation(),res,251)

150. (Operations::sum3ofn(),result,560)
 -(ScientificCalculator::performUnaryOperation(),res,251)
 151. (Operations::sumkofn(),result,574)
 -(ScientificCalculator::performBinaryOperation(),res,237)
 152. (Operations::sumkofn(),result,577)
 -(ScientificCalculator::performBinaryOperation(),res,237)

Criteria:

All-coupling-DU-Path coverage criteria is used here since it subsumes all-coupling-defs and all-coupling-uses.

[k]: A simple path that covers the du-pair numbered k, starting from the first pointer and ending at the second.

Path index [k]:	Testcase:
1	2 7 45 54 3
2	2 44 1 54 3
3	2 23 7 0.875 54 3
4	2 23 44 1 54 3
5	2 5 16 54 3
6	2 7 45 54 3
7	2 8 45 54 3
8	2 9 45 54 3
9	2 10 45 54 3
10	2 11 45 54 3
11	2 12 45 54 3
12	2 23 7 0.875 54 3

13	2 23 8 0.875 54 3
14	2 23 9 0.875 54 3
15	2 23 10 0.875 54 3
16	2 23 11 0.875 54 3
17	2 23 12 0.875 54 3
18	2 13 50 54 3
19	2 14 50 54 3
20	2 15 50 54 3
21	2 16 -12.25 54 3
22	2 17 12.25 54 3
23	2 18 12.25 54 3
24	2 19 13.87 54 3
25	2 21 1 54 3
26	2 22 1 54 3
27	2 24 60 54 3
28	2 25 0.65 54 3
29	2 28 6 54 3
30	2 30 9 54 3
31	2 31 -9 54 3
32	2 37 5 54 3
33	2 44 0.866 54 3
34	2 23 44 0.866 54 3
35	2 45 0.5 54 3
36	2 23 45 0.5 54 3

37	2 46 1 54 3
38	2 23 46 1 54 3
39	2 41 2 54 3
40	2 42 3 54 3
41	2 43 0.5 54 3
42	2 39 2.2 54 3
43	2 40 8.8 54 3
44	2 53 1 54 3
45	2 49 12 54 3
46	2 50 13 54 3
47	2 51 9 54 3
48	2 1 94 52 54 3
49	2 1 94 52 54 3
50	2 2 94 52 54 3
51	2 2 94 52 54 3
52	2 3 59 6 54 3
53	2 3 59 6 54 3
54	2 4 54 9 54 3
55	2 4 54 9 54 3
56	2 6 4 7 54 3
57	2 6 4 7 54 3
58	2 20 7 2 54 3
59	2 20 7 2 54 3
60	2 29 8 3 54 3

61	2 29 8 3 54 3
62	2 32 86 78 54 3
63	2 32 86 78 54 3
64	2 33 54 67 54 3
65	2 33 54 67 54 3
66	2 48 1 0 54 3
67	2 47 1 0 54 3
68	2 34 1 2 54 3
69	2 35 1 2 54 3
70	2 36 1 2 54 3
71	2 38 5 9 54 3
72	2 52 1 4 54 3
73	2 48 1 0 54 3
74	2 47 1 0 54 3
75	2 34 1 2 54 3
76	2 35 1 2 54 3
77	2 36 1 2 54 3
78	2 38 5 9 54 3
79	2 52 1 4 54 3
80	2 26 3 1 2 54 3
81	2 26 3 1 2 54 3
82	2 26 3 1 2 54 3
83	2 27 3 1 2 54 3
84	2 27 3 1 2 54 3

85	2 27 3 1 2 54 3
86	2 48 1 0 54 3
87	2 48 1 0 54 3
88	2 48 1 0 54 3
89	2 47 1 0 54 3
90	2 47 1 0 54 3
91	2 1 94 52 54 3
92	2 2 94 52 54 3
93	2 3 59 6 54 3
94	2 4 54 9 54 3
95	2 5 16 54 3
96	2 6 4 7 54 3
97	2 7 45 54 3
98	2 8 45 54 3
99	2 9 45 54 3
100	2 10 45 54 3
101	2 11 45 54 3
102	2 12 45 54 3
103	2 23 7 0.875 54 3
104	2 23 8 0.875 54 3
105	2 23 9 0.875 54 3
106	2 23 10 0.875 54 3
107	2 23 11 0.875 54 3
108	2 23 12 0.875 54 3

109	2 13 50 54 3
110	2 14 50 54 3
111	2 15 50 54 3
112	2 16 -12.25 54 3
113	2 17 12.25 54 3
114	2 18 12.25 54 3
115	2 19 13.87 54 3
116	2 20 7 2 54 3
117	2 21 1 54 3
118	2 22 1 54 3
119	2 24 60 54 3
120	2 25 0.65 54 3
121	2 26 3 1 2 54 3
122	2 27 3 1 2 54 3
123	2 28 6 54 3
124	2 29 8 3 54 3
125	2 30 9 54 3
126	2 31 -9 54 3
127	2 32 86 78 54 3
128	2 33 54 67 54 3
129	2 48 1 0 54 3
130	2 47 1 0 54 3
131	2 37 5 54 3
132	2 44 0.866 54 3

133	2 23 44 0.866 54 3
134	2 45 0.5 54 3
135	2 23 45 0.5 54 3
136	2 46 1 54 3
137	2 23 46 1 54 3
138	2 41 2 54 3
139	2 42 3 54 3
140	2 43 0.5 54 3
141	2 39 2.2 54 3
142	2 40 8.8 54 3
143	2 34 1 2 54 3
144	2 35 1 2 54 3
145	2 36 1 2 54 3
146	2 38 5 9 54 3
147	2 53 1 54 3
148	2 49 12 54 3
149	2 50 13 54 3
150	2 51 9 54 3
151	2 52 0 4 54 3
152	2 52 1 4 54 3