**Software Testing**

**Project Report**

Session: Spring 2021

**Danish Hasan** MSCS-20001

**Abu Bakar** MSCS-20013

**Musa Khan** MSCS-20065

**Awais** MSCS-20074

**Employee Time Reporting**

Department of Computer Science

**Information Technology University Lahore**

**Pakistan**

****

## Project Contribution

|  |  |  |
| --- | --- | --- |
| **Member** | **Code** | **Report** |
| **Danish** | Wrote test cases in test suites  test\_1 – test\_10  total 16 test cases written.  Setup combined Junit eclipse workspace in our code repository. | Made report for test cases in  test\_1 – test\_10 |
| **Abu Bakar** | Wrote test cases in test suites  testBinaryGCD\_1  testBinaryGCD\_2  testdivWord\_1  testdivWord\_2  testdivWord\_3  total 17 test cases written  Compiled all the code in a single file. | Made report for test cases in  testBinaryGCD\_1  testBinaryGCD\_2  testdivWord\_1  testdivWord\_2  testdivWord\_3 |
| **Awais** | Wrote test cases in test suites  MathContextTest\_1  MathContextTest\_2  MathContextTest\_3  passesMillerRabinTest1  passesMillerRabinTest2  passesMillerRabinTest3  passesMillerRabinTest4  subtractTest1  subtractTest2  subtractTest3  total 10 test cases written | Made report for test cases in  MathContextTest\_1  MathContextTest\_2  MathContextTest\_3  passesMillerRabinTest1  passesMillerRabinTest2  passesMillerRabinTest3  passesMillerRabinTest4  subtractTest1  subtractTest2  subtractTest3 |
| **Musa** | No contribution | No contribution |

## White-Box Testing with Junit

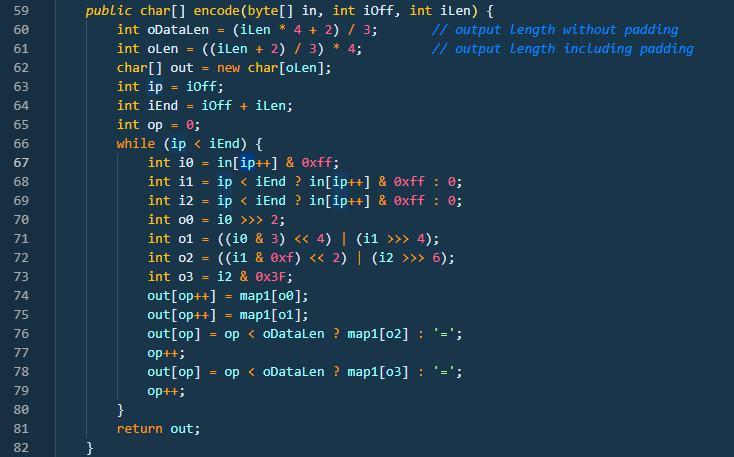
### **Function 1**:

Encodes a byte array into Base64 format.

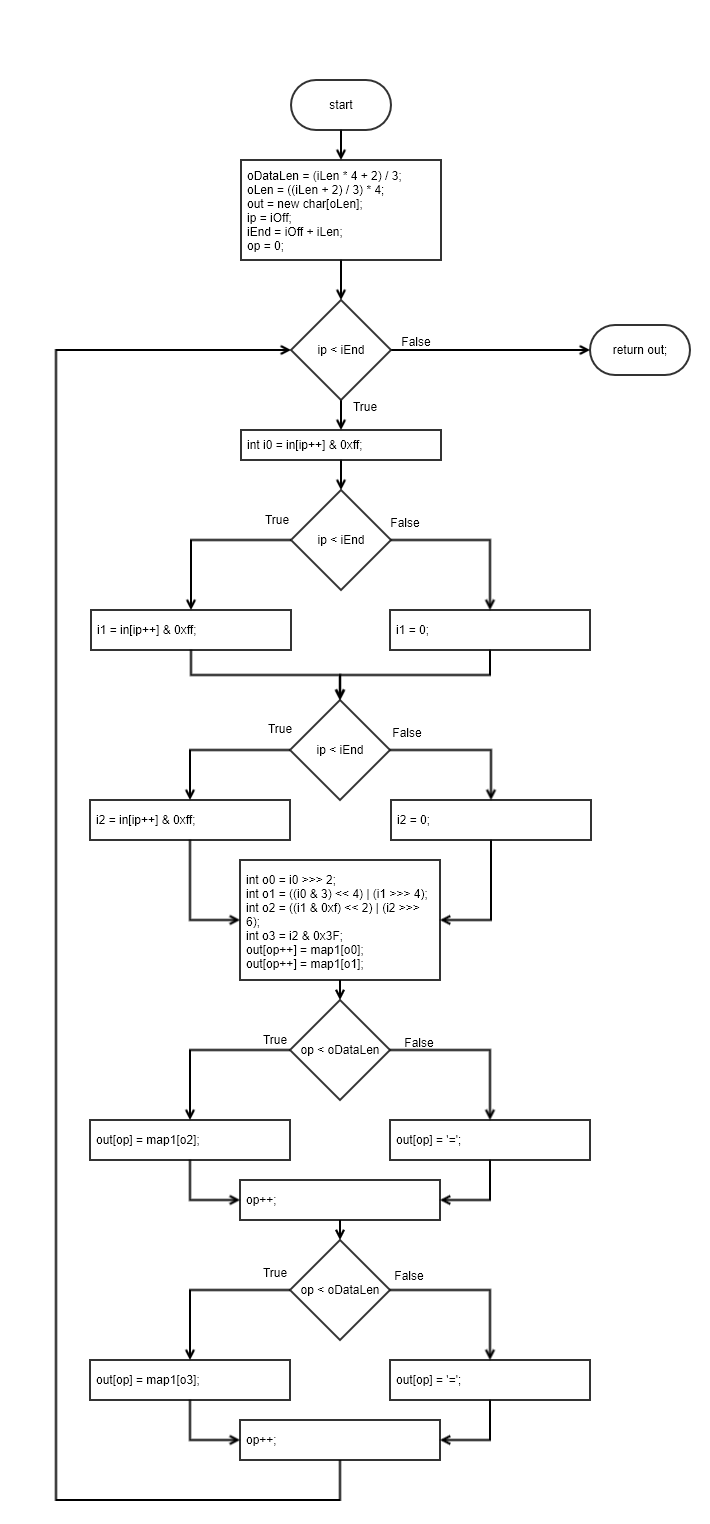
Note: map[] table is populated in another constructor function.

**Source Code:**

timesheet-master\src\main\java\timeSheet\util\properties\Base64Coder.java



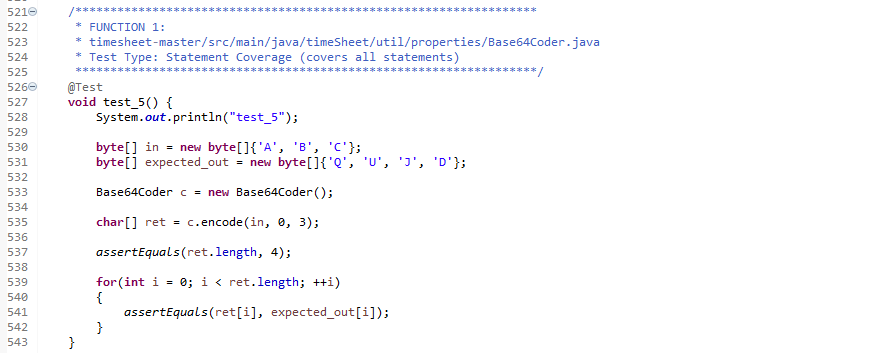
**CFG:**

****

**Statement Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | QUJD | Pass | Covers all statements |

**Test Case Code:**



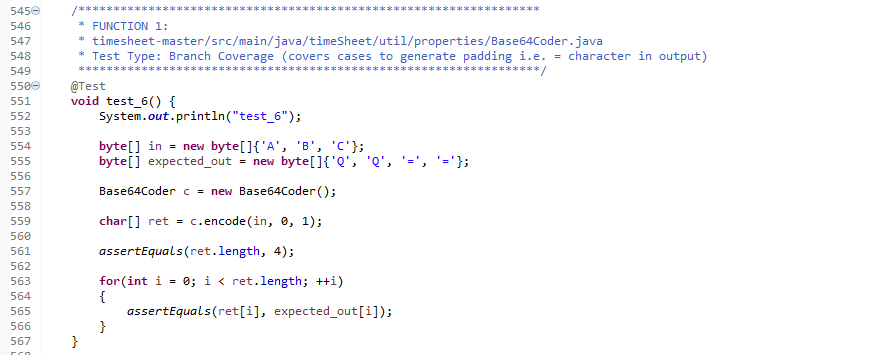
**Test Case Result:**



**Branch Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 1; | QQ== | QQ== | Pass | Covers 66TF, 68F, 69F, 76F, 78F |

**Test Case Code:**



**Test Case Result:**



**Basis Path:**

Edges - Nodes + 2 = 22 – 18 + 2 = 6

Path 1: 66F

Path 2: 66T, 68T, 69T, 76T, 78T

Path 3: 66T, 68T, 69F, 76T, 78F

Path 4: 66T, 68F, 69F, 76F, 78F

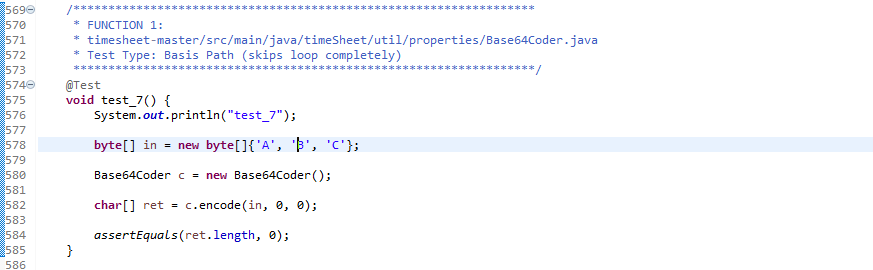
Path 5: 66T, 68F, 69F, 76F, 78T

Path 6: 66T, 68F, 69T, 76F, 78F

Note that no logical path is possible to cause 69T while 68F. Same is the case with 76F and 78T. Similarly, conditions in 76 and 78 also depend upon the same factor as 68, 69 so it is not possible for 68T but 76F and vice versa.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 0; | Empty String | Empty String | Pass | Covers Path1 |

**Test Case Code:**



**Test Case Result:**



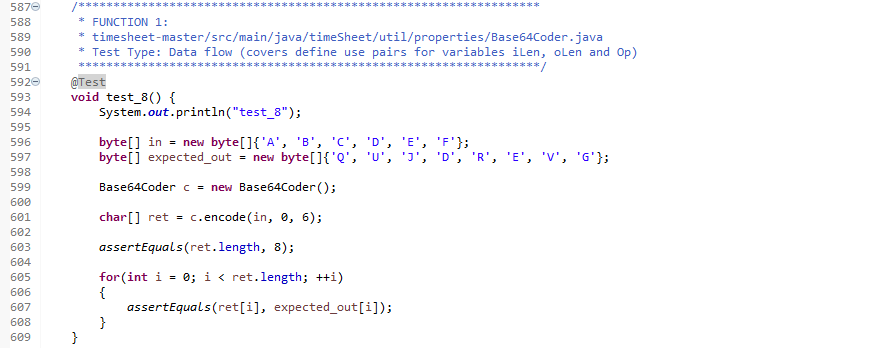
**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | iLen | 59 | 60, 61, 64 |
| 2 | oLen | 61 | 62 |
| 3 | Op | 65, 74, 75, 77, 79 | 74, 75, 76, 77, 78, 79 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | iLen | <59, 60>, <59, 61>, <59, 64> |
| 2 | oLen | <61, 62> |
| 3 | Op | <65,74>, <74,75>, <75,76>, <75,77>, <77,78>, <77,79>, <79,74> |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’};  iOff = 0;  iLen = 6; | QUJDREVG | QUJDREVG | Pass | iLen = Covers <59, 60>, <59, 61>, <59, 64>  oLen = Covers <61, 62>  op = Covers <65,74>, <74,75>, <75,76>, <75,77>, <77,78>, <77,79>, <79,74> |

**Test Case Code:**



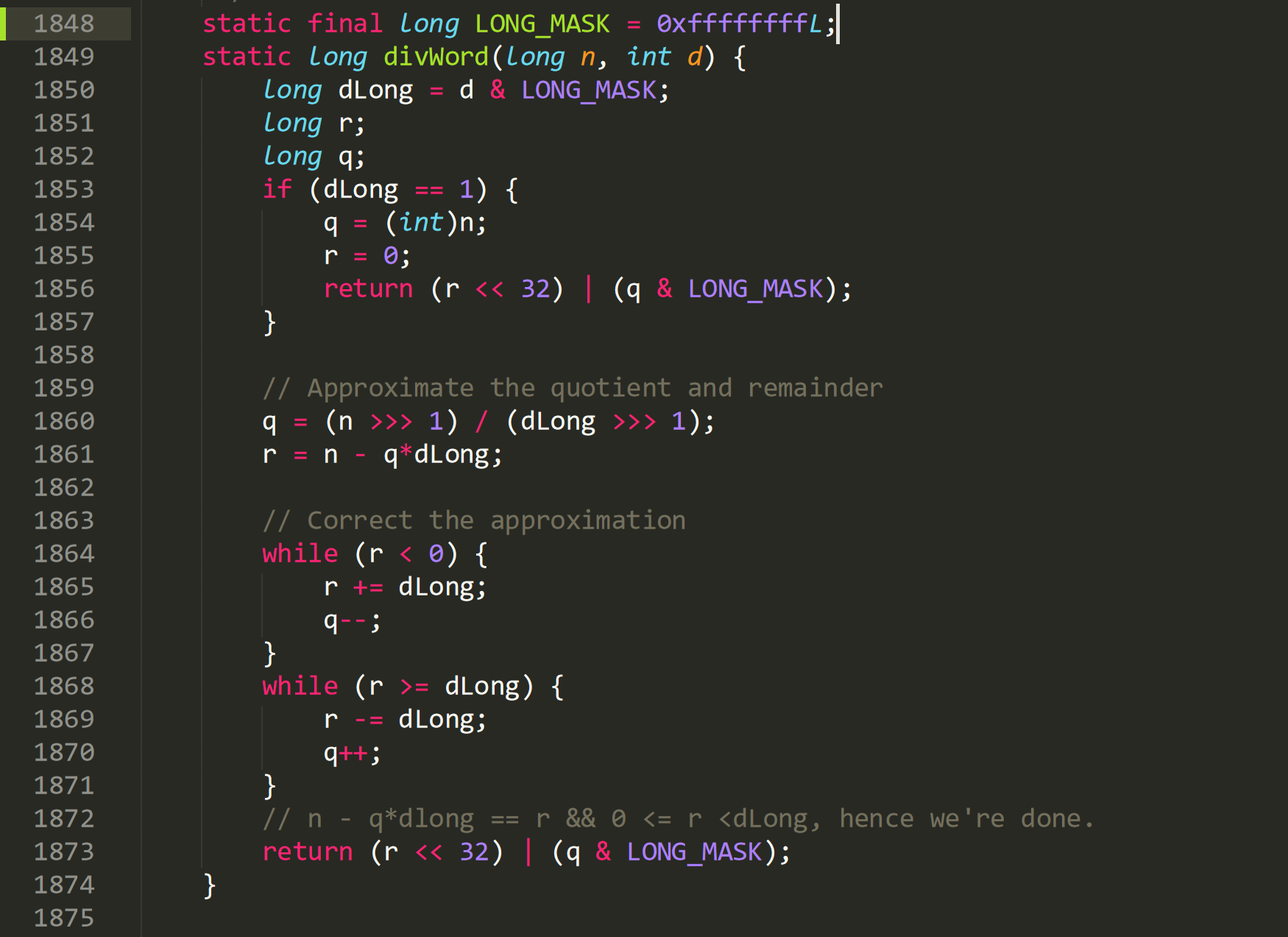
**Test Case Result:**



### **Function 3:**

**Source Code:**

https://github.com/openjdk/jdk/tree/master/src/java.base/share/classes/java/math/ MutableBigInteger.java

****

****

**CFG:**

Diagram

Description automatically generated

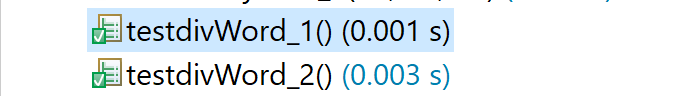
**Test Case Code:**



**Branch Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | n = 16  d = 1 | 16 | 16 | Pass | Covers Statement 1850-1857 |
| **2** | n = 10  d = 3 | 4294967299 | 4294967299 | Pass | Covers Statement 1850,1851,1852, 1860-1868, 1873 |
| **3** | - | - | - | - | Statement 1869- 1870 I think this is a dead code, I could not find any such case in which the condition at 1868 becomes True |

**Test Result**



**Loop Boundary:**

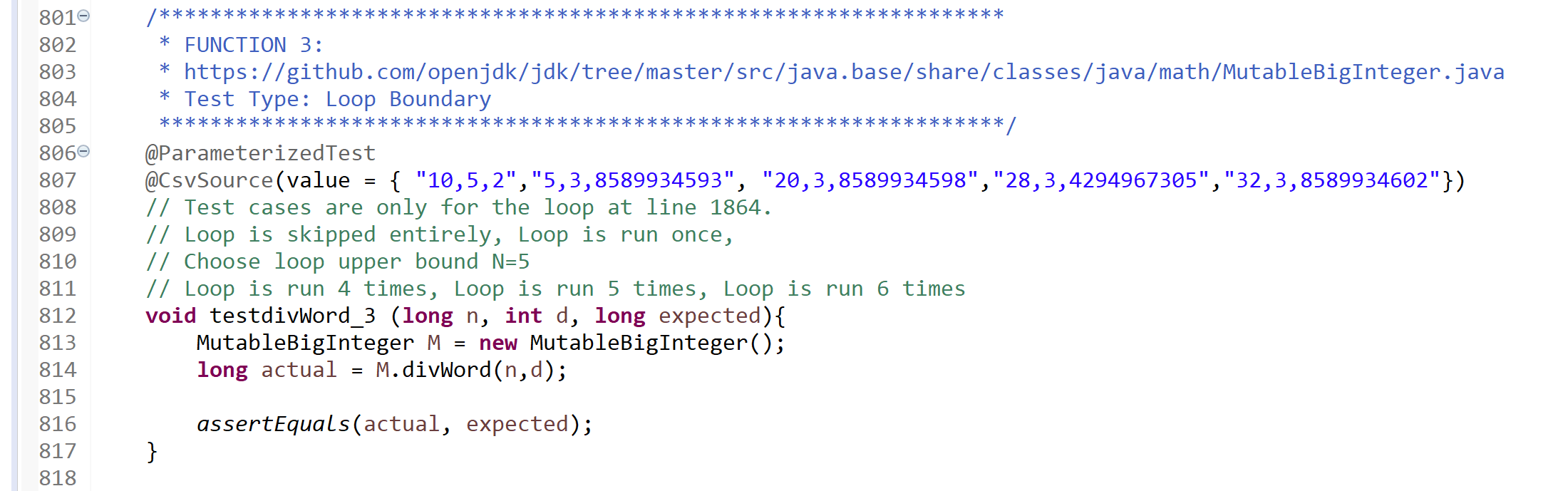
I think Loop at line 1868 is a dead code, I could not find any such case in which the condition at 1868 becomes True.

**Test cases are only for the loop at line 1864.**

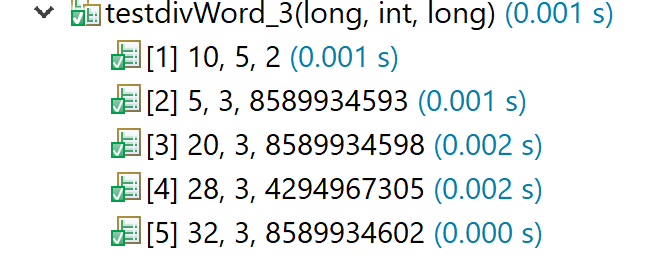
I choose loop upper bound = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | n =10  d = 5 | 2 | 2 | Pass | Loop at line 1864 is skipped entirely. |
| **2** | n =5  d = 3 | 8589934593 | 8589934593 | Pass | Loop at line 1864 is run only once |
| **3** | n =20  d =3 | 8589934598 | 8589934598 | Pass | Loop at line 1864 is run 4 times |
| **4** | n = 28  d = 3 | 4294967305 | 4294967305 | Pass | Loop at line 1864 is run 5 times. |
| **5** | n = 32  d = 3 | 8589934602 | 8589934602 | Pass | Loop at line 1864 is run 6 times. |

**Test Case Code:**

****

**Test Result:**

****

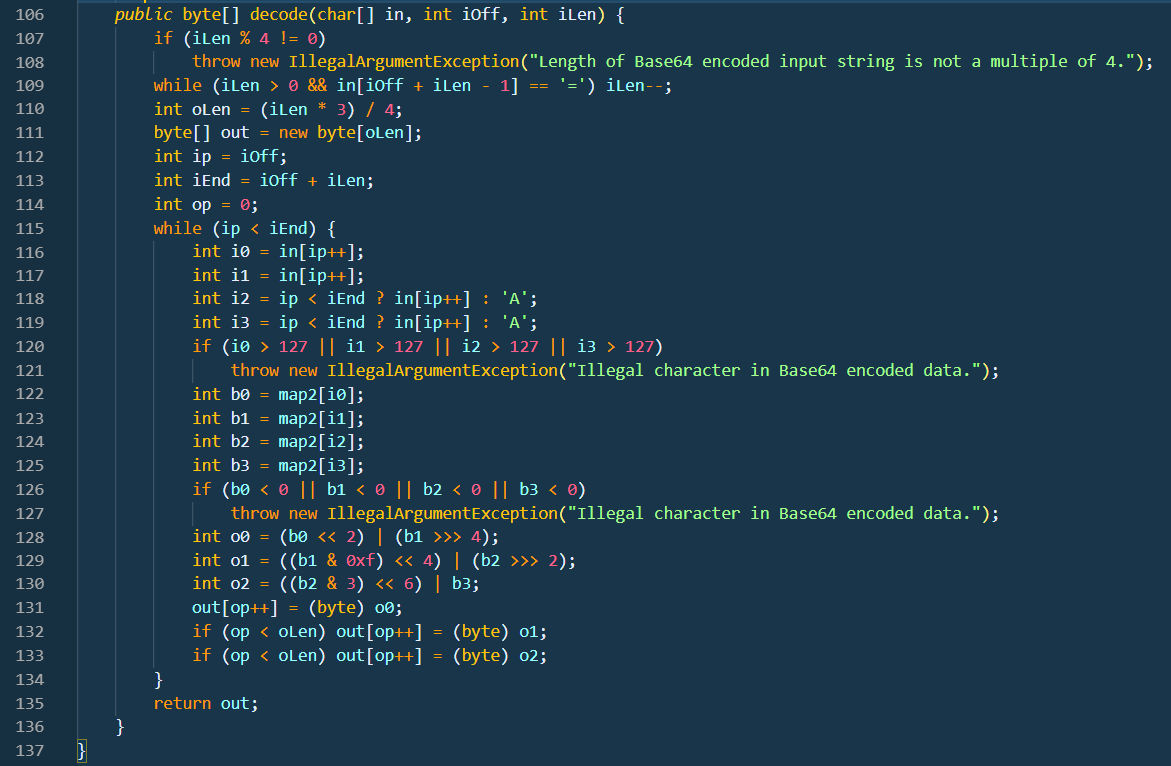
### **Function 4:**

Decodes a byte array from Base64 format.

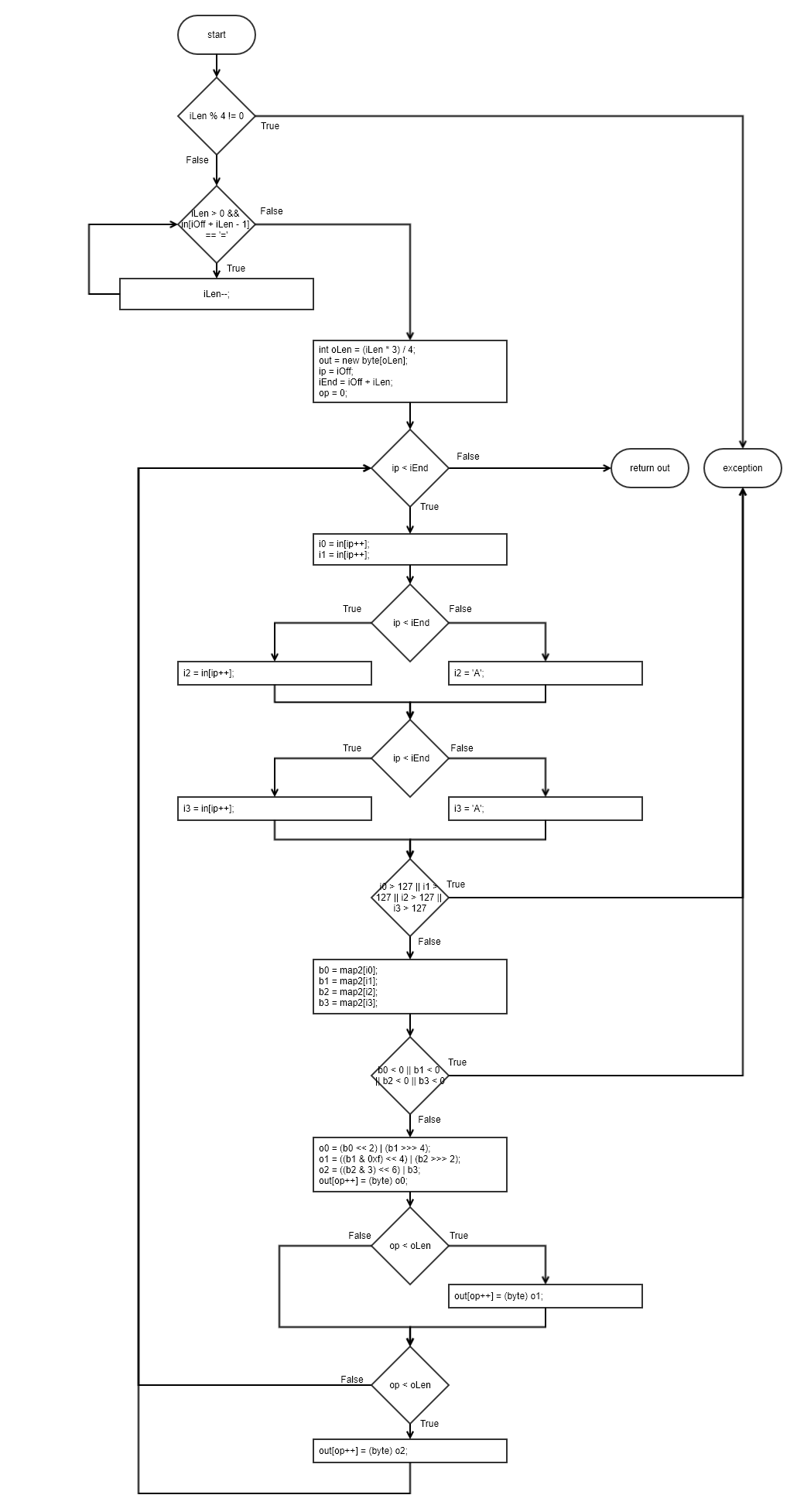
Note: map2[] table is populated in another constructor function.

**Source Code:**

timesheet-master\src\main\java\timeSheet\util\properties\Base64Coder.java



**CFG:**



**Boundary Interior:**

Exception cases are not covered under sir’s guidance.

Possible logical paths:

* A: 118T->119T-> 132T-> 133T
* B: 118T-> 119F-> 132T->133F
* C: 118F-> 119F-> 132T-> 133F

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | ‘ABC’ | Pass | Covers Path A |
| **2** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | ‘A’ | Pass | Covers Path B |
| **3** | In[] = ‘QUI=’  iOff = 0  iLen = 4 | ‘AB’ | ‘AB’ | Pass | Covers Path C |

**Test Case Code:**



**Test Case Result:**

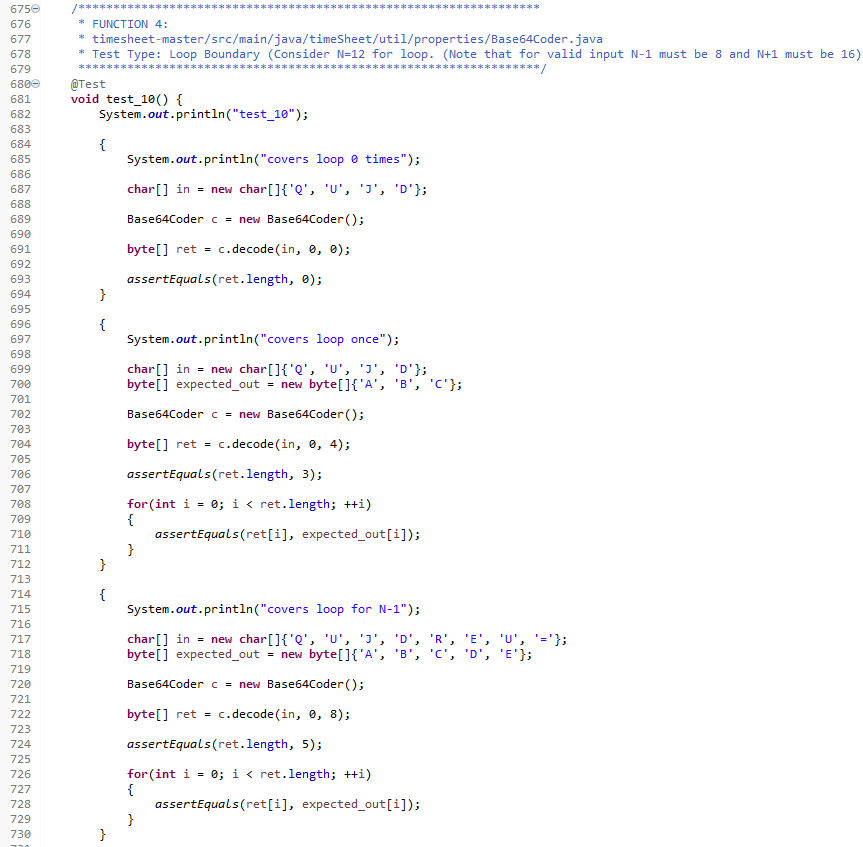


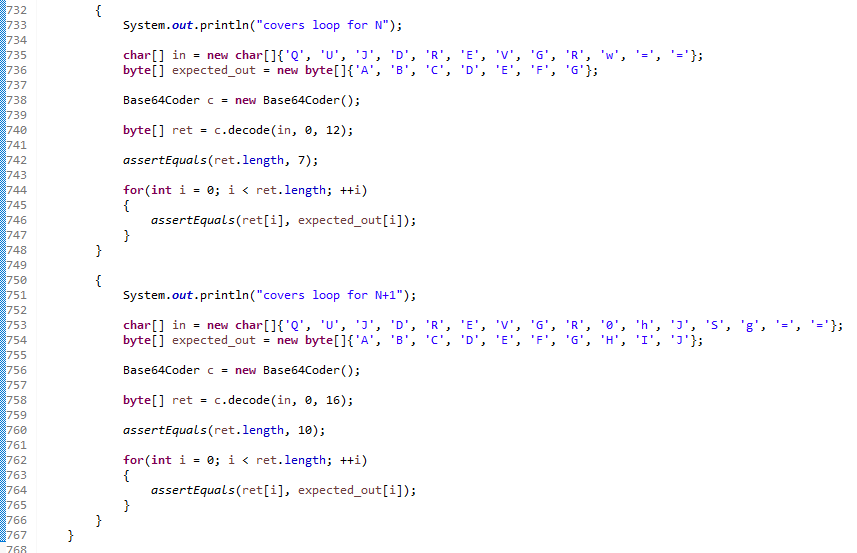
**Loop Boundary:**

Consider N=12 for loop. (Note that for valid input N-1 must be 8 and N+1 must be 16)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 0 | Empty String | Empty String | Pass | Covers 115F |
| **2** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | ‘ABC’ | Pass | Covers 115F once |
| **3** | In[] = ‘QUJDREU=’  iOff = 0  iLen = 8 | ‘ABCDE’ | ‘ABCDE’ | Pass | Covers 115T for N-1 |
| **4** | In[] = ‘QUJDREVGRw==’  iOff = 0  iLen = 12 | ‘ABCDEFG’ | ‘ABCDEFG’ | Pass | Covers 115T for N |
| **5** | In[] = ‘QUJDREVGR0hJSg==’  iOff = 0  iLen = 16 | ‘ABCDEFGHIJ’ | ‘ABCDEFGHIJ’ | Pass | Covers 115T for N+1 |

**Test Case Code:**





**Test Case Result:**



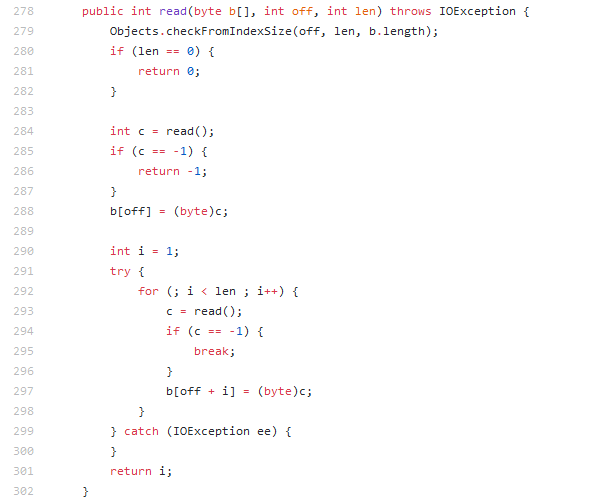
### **Function 5:**

**Source Code:**

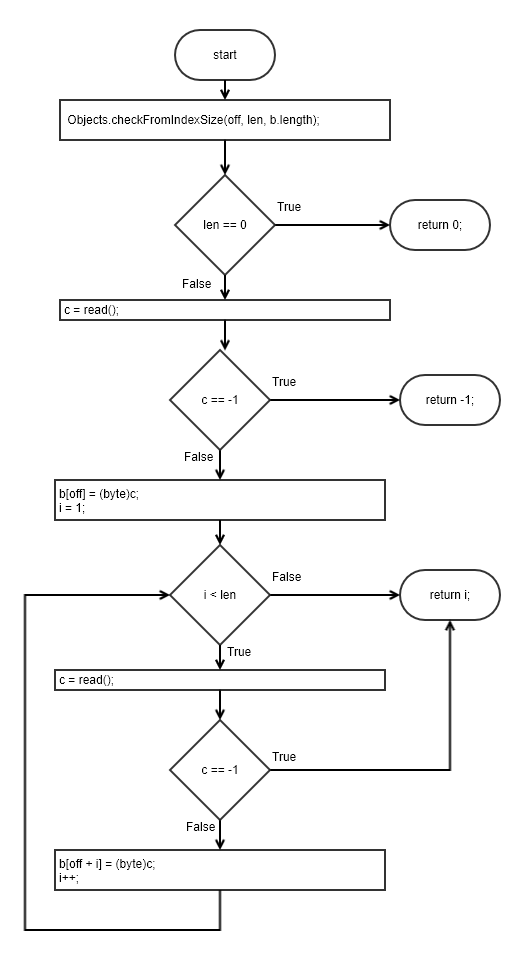
https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/io/InputStream.

Java

checkFromIndexSize and read are external APIs. checkFromIndexSize can be implemented as dummy stub while read is implemented as needed by each test case.



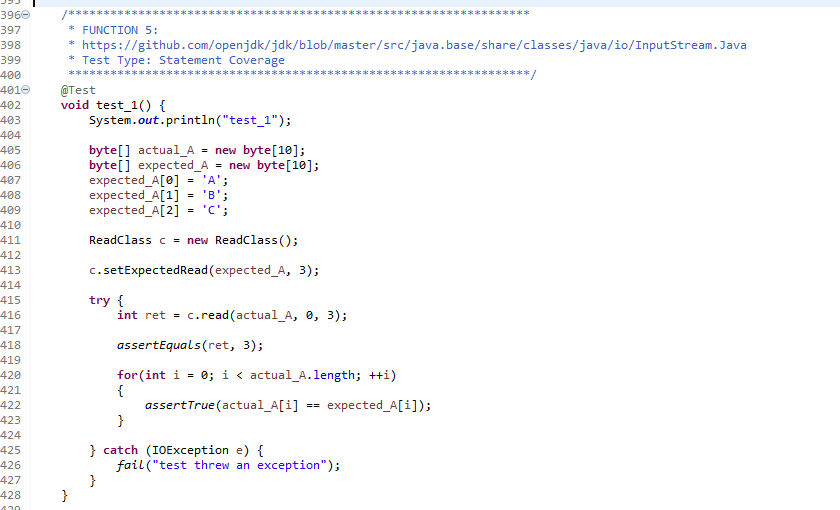
**CFG:**



**Statement Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | 3,  b[] =‘ABC’ | Pass | External module API read() returns ‘A’, ‘B’, ‘C’ in consecutive calls. |

**Test Case Code:**



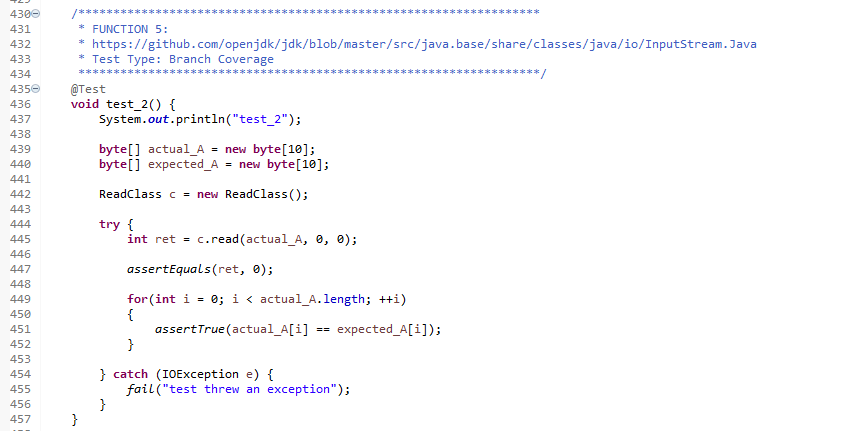
**Test Case Result:**



**Branch Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 0 | 0,  b[] = Empty Array | 0,  b[] = Empty Array | Pass | External module API read() is never called.  280T |

**Test Case Code:**



**Test Case Result:**

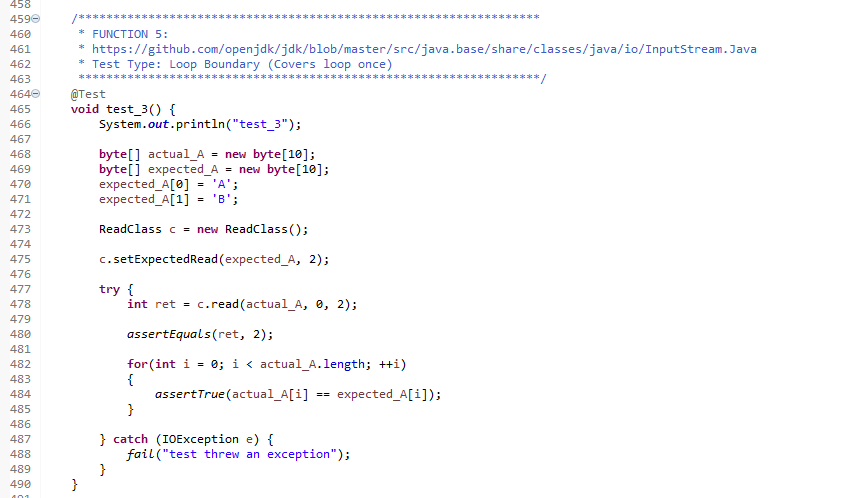


**Loop Boundary:**

Consider N=4 for loop boundary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 2 | 2,  b[] =‘AB’ | 2,  b[] =‘AB’ | Pass | External module API read() returns ‘A’, ‘B’ in consecutive calls.  Covers 292T once |

**Test Case Code:**



**Test Case Result:**



**Basis Path:**

Decision points + 1 = 4 + 1 = 5

Path 1: 280T

Path 2: 280F, 285T

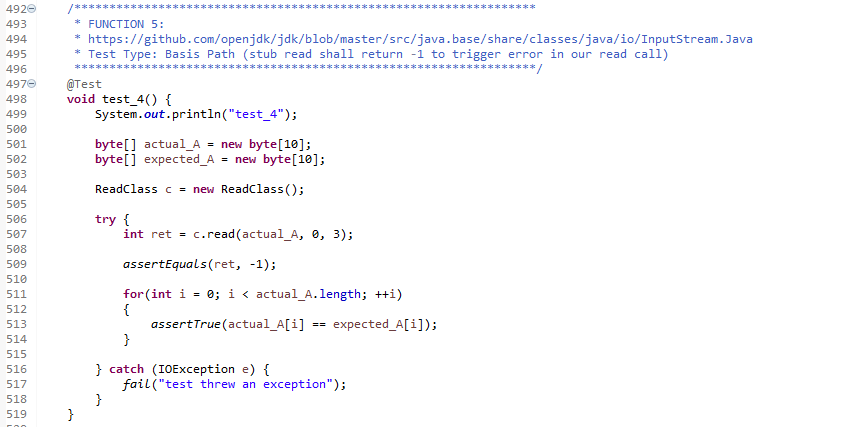
Path 3: 280F, 285F, 292F

Path 4: 280F, 285F, 292TF, 294F

Path 5: 280F, 285F, 292T, 294T

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected**  **Output** | **Actual**  **Output** | **Result** | **Comments /**  **Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | -1,  b[] = Empty Array | -1,  b[] = Empty Array | Pass | External module API read() returns -1 to notify an error at first call.  **Covers Path2** |

**Test Case Code:**



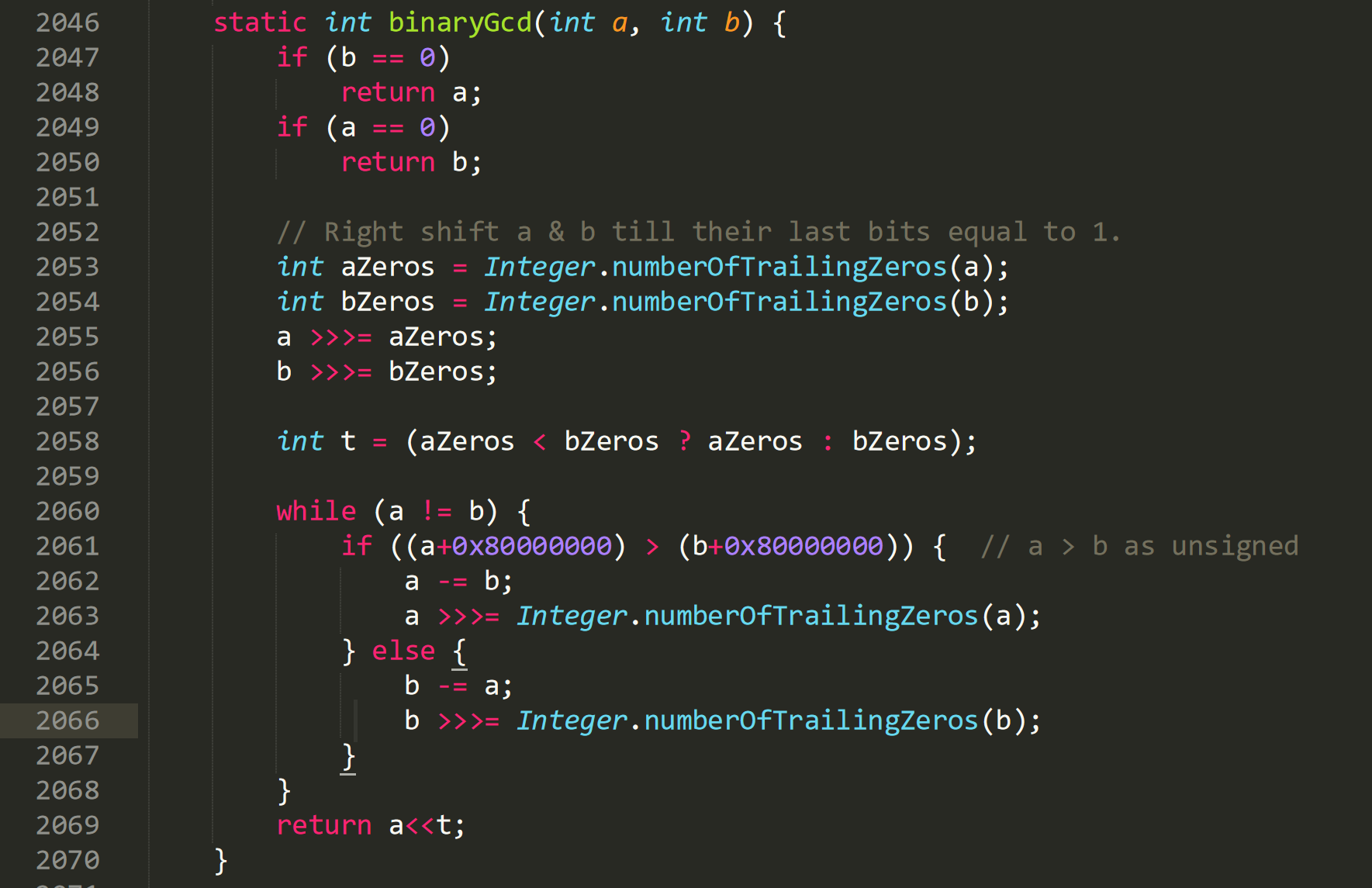
**Test Case Result:**



### **Function 6:**

**Source Code:**

https://github.com/openjdk/jdk/tree/master/src/java.base/share/classes/java/math/ MutableBigInteger.java

****

**CFG:**

Diagram

Description automatically generated.

**Statement Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | 15 | Pass | Covers statement 2047-2048 |
| **2** | a = 0  b =15 | 15 | 15 | Pass | Covers statement 2049-2050 |
| **3** | a = 98  b =56 | 14 | 14 | Pass | Covers statement 2047,2049, 2051-2069 |

**Branch Coverage:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | 15 | Pass | Covers B2047T |
| **2** | a = 0  b =15 | 15 | 15 | Pass | Covers B2049T, B2047F |
| **3** | a = 98  b =56 | 14 | 14 | Pass | Covers B2047F, B2049F, B2060TF, B2061T |
| **4** | a = 56  b =98 | 14 | 14 | Pass | Covers B2047F, B2049F, B2060TF, B2061F |

**Boundary Interior:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | a = 98  b =56 | 14 | 14 | Pass | Covers boundary interior path  DEG |
| **2** | a = 56  b =98 | 14 | 14 | Pass | Covers boundary interior path  DEF |

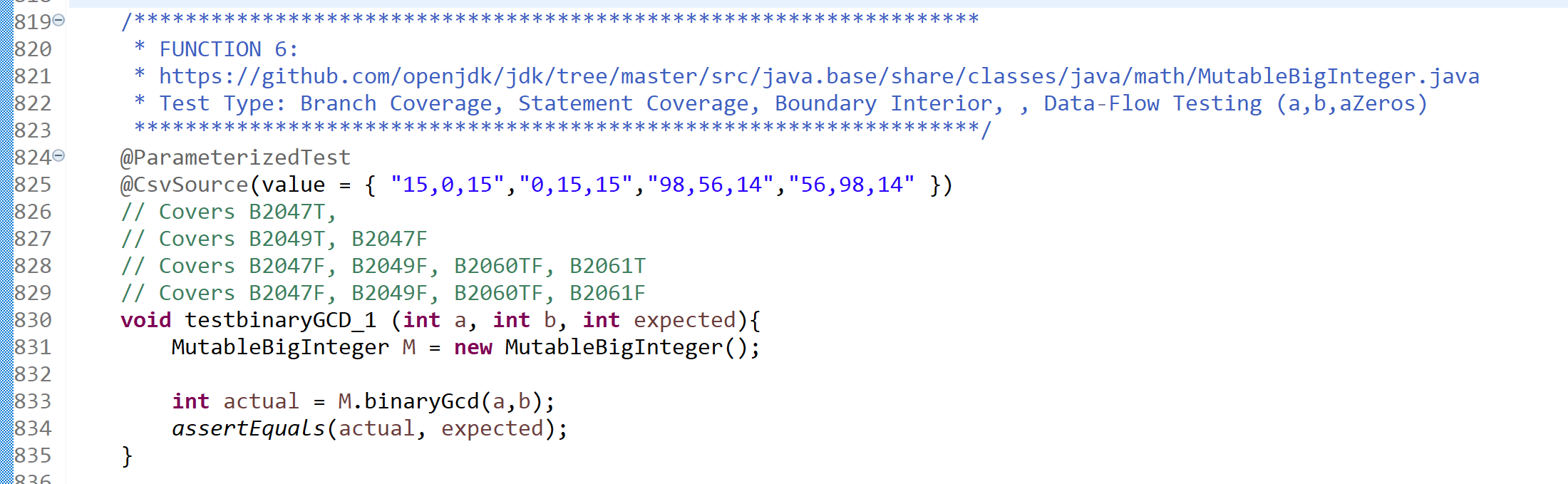
**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | A | 2046, 2055, 2062, 2063 | 2048, 2049, 2053, 2055, 2060, 2061, 2062, 2063, 2065, 2069 |
| 2 | b | 2046, 2056, 2065, 2066 | 2047, 2050, 2054, 2056, 2060, 2061, 2062, 2065, 2066 |
| 3 | aZeros | 2053 | 2055, 2058 |

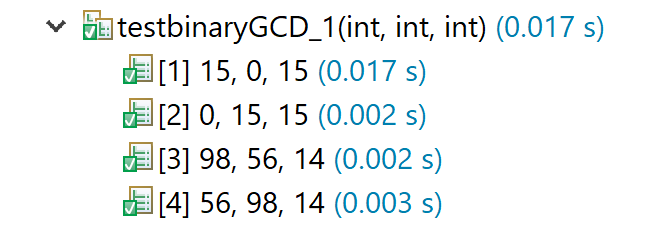
|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | a | <2046, 2048> <2046, 2049>  <2046, 2053> <2046, 2055>  <2055, 2060> <2055, 2061>  <2055, 2062> <2055, 2065>  <2055, 2069>  <2062, 2063>  <2063, 2060> <2063, 2061>  <2063, 2062> <2063, 2069> |
| 2 | b | <2046, 2047> <2046, 2050>  <2046, 2054> <2046, 2056>  <2056, 2060> <2056, 2061> <2056, 2062> <2056, 2065>  <2065, 2066>  <2066, 2060> <2066, 2061> <2066, 2062> |
| 3 | aZeros | <2053, 2055> <2053, 2058> |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | 15 | Pass | For a covers  <2046, 2048>  For b covers  <2046, 2047> |
| **2** | a = 0  b =15 | 15 | 15 | Pass | For a covers  <2046, 2049>  For b covers  <2046, 2047>  <2046, 2050> |
| **3** | a = 98  b =56 | 14 | 14 | Pass | For a covers  <2046, 2049>  <2046, 2053>  <2046, 2055>  <2055, 2060>  <2055, 2061>  <2055, 2062>  <2062, 2063>  <2063, 2060>  <2063, 2061>  <2063, 2062>  <2063, 2069>  For b covers  <2046, 2047>  <2046, 2054>  <2046, 2056>  <2056, 2060>  <2056, 2061> <2056, 2062>  For aZeros covers:  <2053, 2055>  <2053, 2058> |
| **4** | a = 56  b =98 | 14 | 14 | Pass | For a covers  <2046, 2049>  <2046, 2053>  <2046, 2055>  <2055, 2060>  <2055, 2061>  <2055, 2065>  <2055, 2069>  For b covers  <2046, 2047>  <2046, 2054>  <2046, 2056>  <2056, 2060>  <2056, 2061>  <2056, 2065>  <2065, 2066>  <2066, 2060>  <2066, 2061> <2066, 2062>  For aZeros covers:  <2053, 2055>  <2053, 2058> |

**Test Code**



**Test Result**



**Loop Boundary:**

I choose loop upper bound = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | a = 12  b = 12 | 12 | 12 | Pass | Loop is skipped entirely. |
| **2** | a = 4  b = 2 | 2 | 2 | Pass | Loop is run only once |
| **3** | a = 6  b = 2 | 2 | 2 | Pass | Loop is run twice. |
| **4** | a = 10  b = 2 | 2 | 2 | Pass | Loop is run 4 times |
| **5** | a = 12  b = 2 | 2 | 2 | Pass | Loop is run 5 times. |
| **6** | a = 14  b = 2 | 2 | 2 | Pass | Loop is run 6 times. |

**Basis Path:**

No. of Basis Paths = No. of decision points + 1

No. of Basis Paths = 4 + 1 = 5

Path 1: AI

Path 2: ABJ

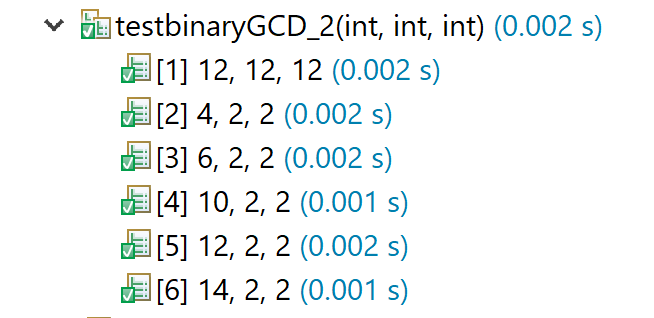
Path 3: ABCDH

Path 4: ABCDEFH

Path 5: ABCDEGH

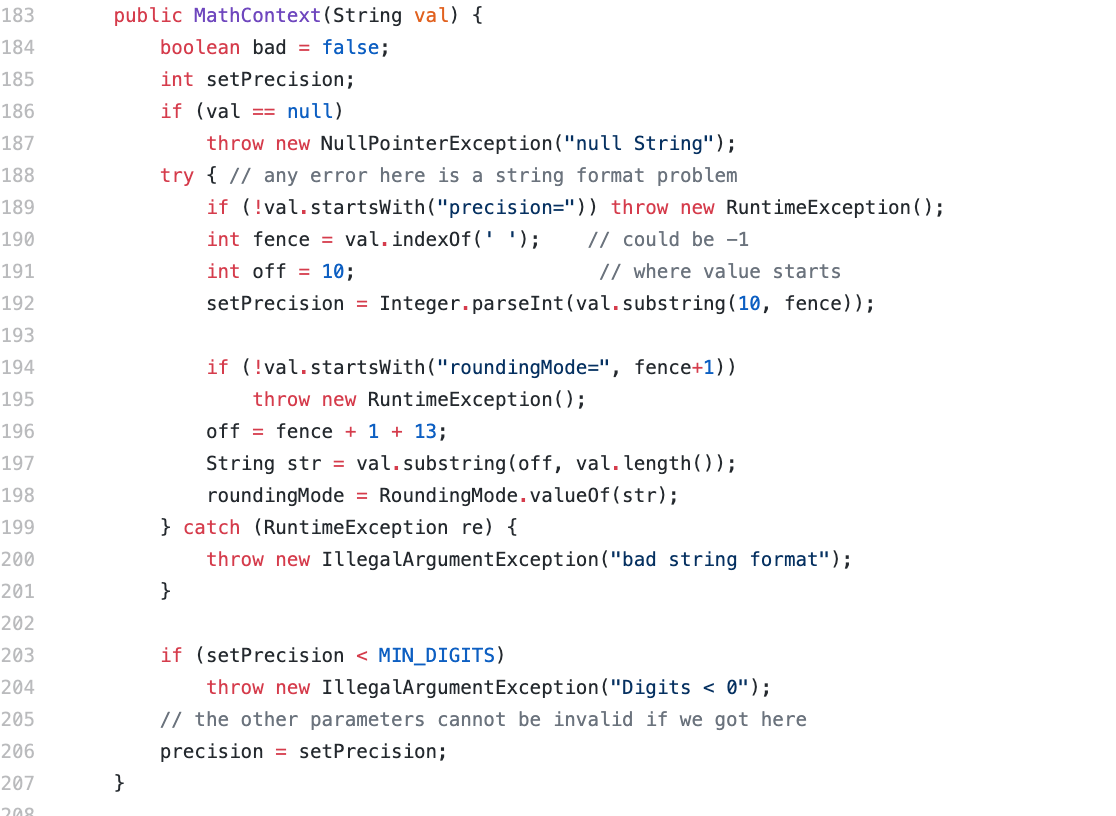
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | 15 | Pass | Covers basis path AI |
| **2** | a = 0  b =15 | 15 | 15 | Pass | Covers basis path ABJ |
| **3** | a = 12  b = 12 | 12 | 12 | Pass | Covers basis path ABCDH |
| **4** | a = 2  b = 4 | 2 | 2 | Pass | Covers basis path ABCDEFH |
| **5** | a = 4  b = 2 | 2 | 2 | Pass | Covers basis path ABCDEFH |

**Test Result**

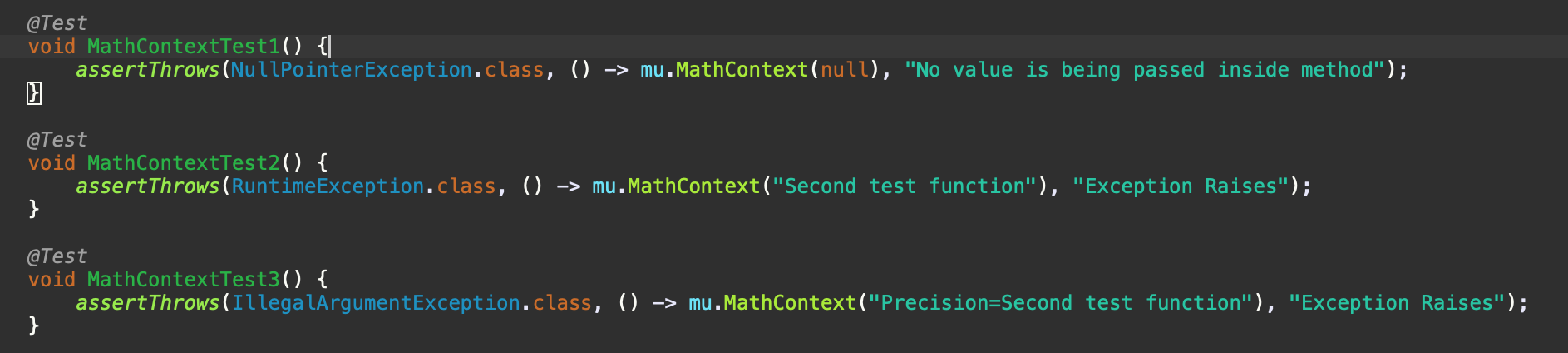


### **Function 8:**

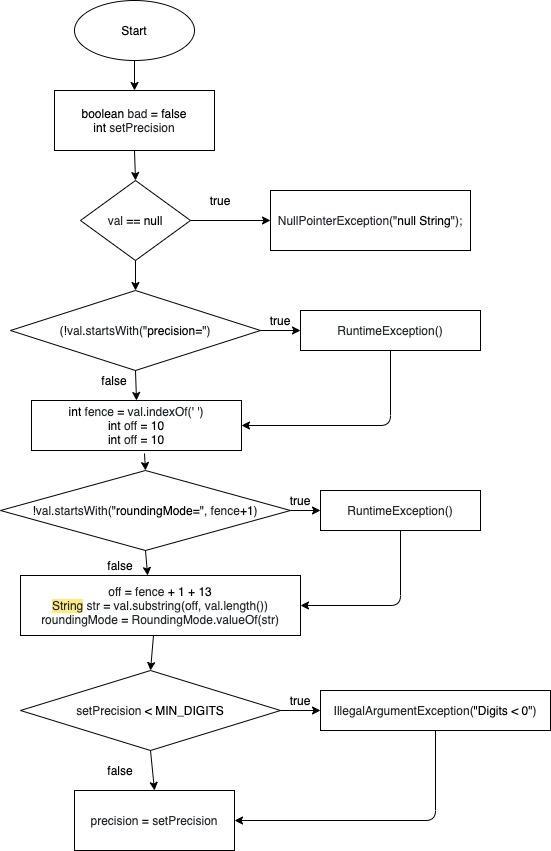
**Source Code:**

****

**Test Code:**

****

**CFG:**

****

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | null | exception | Covered 184, 185, 186, 187 |
| **2** | Second test function | exception | Covered 184, 185, 186, 188, 189 |
| **3** | ‘precision=Second test function’ | exception | Covered 184, 185, 186, 188, 190, 191, 192, 194, 195 |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (null) | exception | Covered B186(True) |
| **2** | Second test function | exception | Covered B186(False), B189(True) |
| **3** | ‘precision=Second test function’ | exception | Covered B186(False), B189(False), B194(True) |

**Test Result:**

****

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (null) | exception | Covered C186(True) |

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | Second test function | exception | Covered C186(False), C189(True) |
| **3** | ‘precision=Second test function’ | exception | Covered C186(False), C189(False),  C194(True) |

**Boundary Interior:**

No Loop in the program.

**Loop Boundary:**

No Loop in the program.

**Basis Path:**

No of decision points = 4

No. of basis path = No of decision points +1 = 4+1 = 5

**Path 1:**

183, 184, 185, 186, 203, 206

**Path 2:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 196, 197, 198, 203, 206

**Path 3:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 203, 206

**Path 4:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 203, 204, 206

**Path 5:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 203, 204, 206

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | null | Exception | Covers Path 1 |
| **2** | ‘precision=Second test function’ | Exception | Covers Path 3 |
| **3** | Second test function | Exception | Covers Path 5 |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | Val | 183 | 186,189,190,192,197 |
| 2 | setPrecision | 185,192 | 203,206 |
| 3 | Fence | 190 | 192,194 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | Val | <183,186>,<183,189>,<183,190>,<183,192>,<183,197> |
| 2 | setPrecision | <192,203>,<192,206> |
| 3 | Fence | <190,192>,<190,194> |

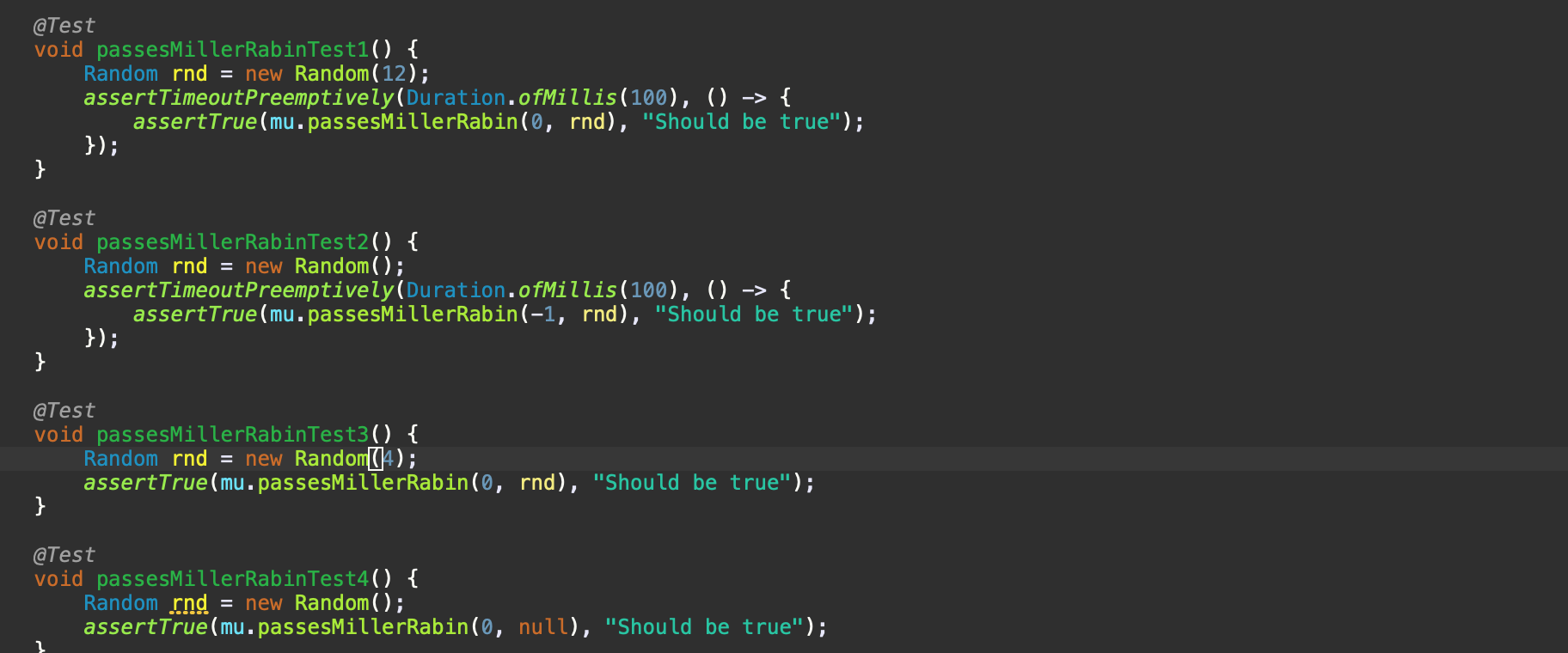
|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | Second test function | Exception | For Val:  <183,186>,<183,189>  For setPrecision:  Not used  For Fence:  Not used  because it does not contains ‘precision=’ at start |
| **2** | ‘precision=Second test function’ | Exception | For Val:  <183,186>,<183,189>,<183,190>,<183,192>  For setPrecision:  Not used  For Fence:  <190,192>,<190,194>  It returns exception because when next if executes it’ll not find ‘roundingMode=’ at start |

### **Function 9**

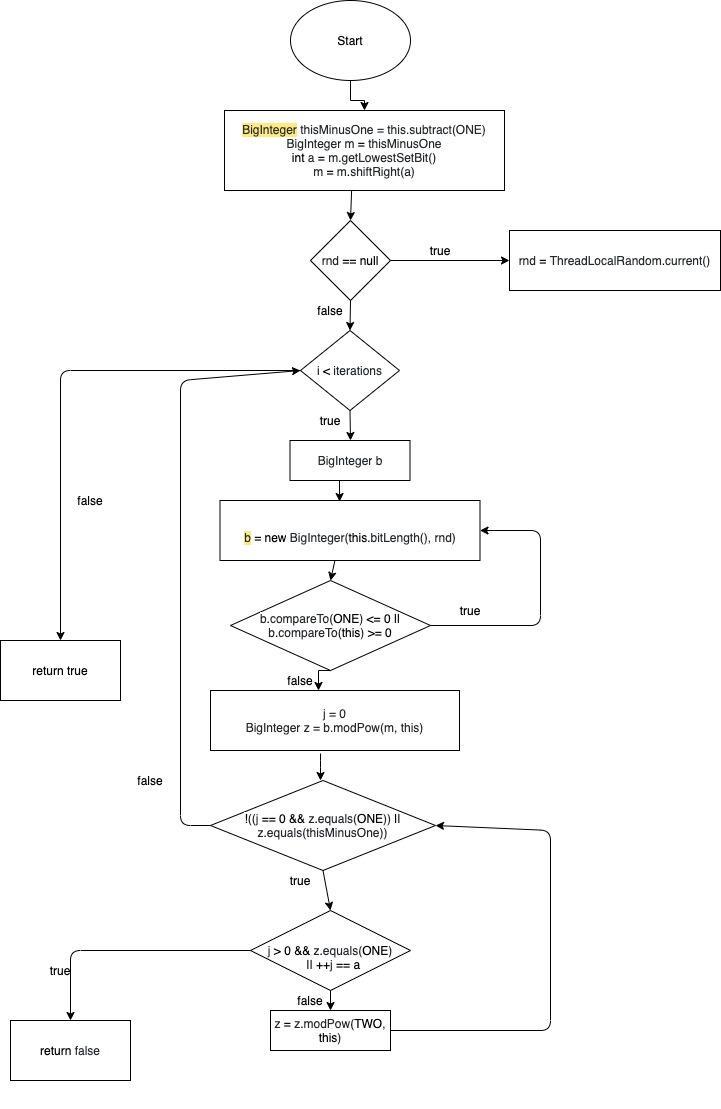
**Source Code:**

****

**Test Case Code:**

****

**CFG:**

****

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (0, null) | true | covers 1103,1104,1105,,1106,1109,1110,1111,1112,1113,1114-1128 |
| **2** | (0,4) | true | covers 1103,1104,1105,,1106,1109,1112,1127 |
| **3** | (0,12) | true | covers 1103-1111,1112 |

**Test Result:**



**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (0, null) | true | covers B1109(T),  B1112(T), B1117(T),  B1121(T) |
| **2** | (0, 4) | true | covers B1109(F),  B1112(F) |
| **4** | (0,12) | true | covers B1109(T)  B1112(T), B1117(T),  B1121(T), B1122(T) |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (0, null) | true | covers C1109(T),  C1112(T), C1117(T),  C1121(T) |
| **2** | (0, 4) | true | covers C1109(F),  C1112(F) |
| **3** | (0,12) | true | covers C1109(T),  C1112(T), C1117(T),  C1121(T), C1122(T) |

**Boundary Interior:**

Below we are taking line numbers to execute boundary interior.

1112 -> 1114

1112 -> 1114 -> 1115

1112 -> 1114 -> 1116 -> 1117

1112 -> 1114 -> 1116 -> 1117 -> 1116

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1123

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121 -> 1127

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (4, null) | True | Covers 1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121 -> 1126 |
| **2** | (0, 4) | True | Covers 1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121 -> 1127 |

**Loop Boundary:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (0,4) | True | Covers 1109T  When the loop will not execute |
| **2** | (0,12) | True | Covers 1112T once |
| **3** | (4,2) | False | Covers 1112T  more than one passes |

**Basis Path:**

No of decision points = 3

No. of basis path = No of decision points +1 = 3+1 = 4

**Path 1:**

1101, 1103, 1104, 1105, 1106, 1127

**Path 2:**

1101, 1103, 1104, 1105, 1106, 1109, 1110, 1127

**Path 3:**

1101, 1103, 1104, 1105, 1106, 1109, 1110, 1112, 1113, 1114, 1115, 1116, 1117, 1119, 1120, 1127

**Path 4:**

1101, 1103, 1104, 1105, 1106, 1109, 1110, 1112, 1113, 1114, 1115, 1116, 1117, 1119, 1120, 1121, 1122, 1123, 1124, 1127

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (4, null) | True | Covers Path 1 |
| **2** | (0, 4) | True | Covers Path 2 |
| **3** | (null, null) | True | Covers Path 3 |
| **4** | (7,9) | False | Covers Path 4 |

**Data Flow Testing:**

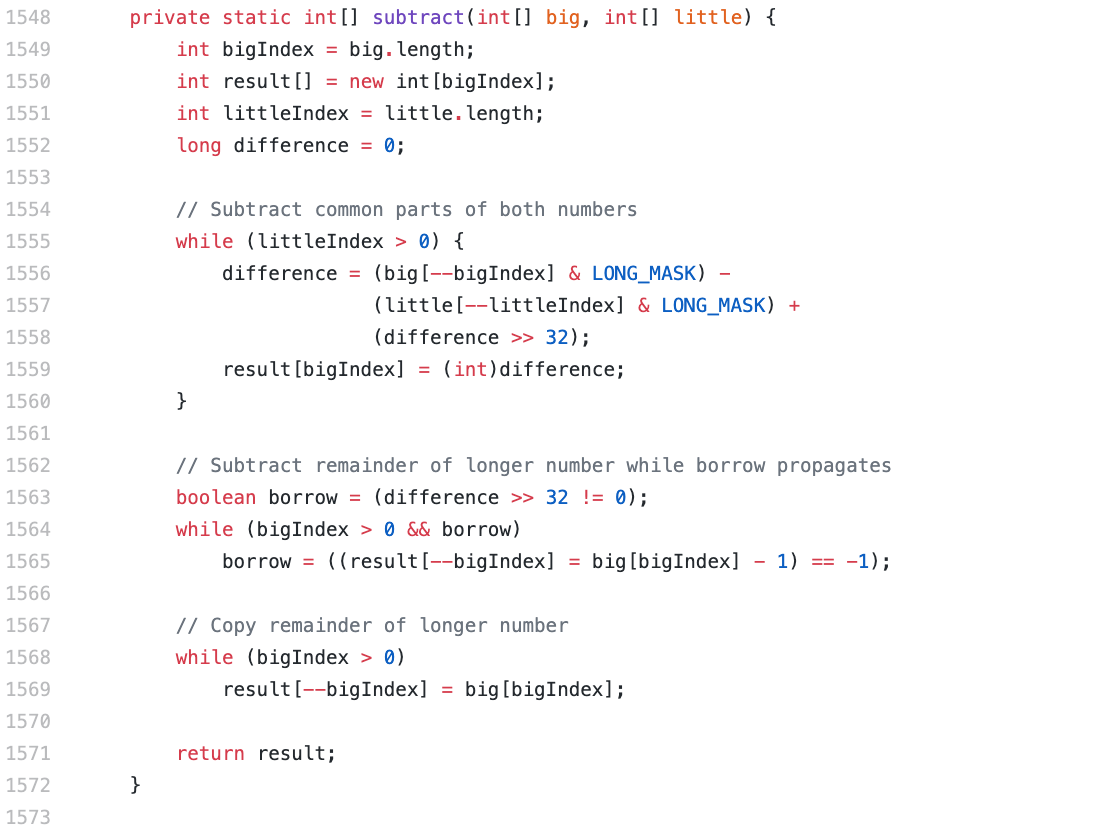
|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | iterations | 1101 | 1112 |
| 2 | Rnd | 1101,1110 | 1109,1116 |
| 3 | A | 1105 | 1106 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | iterations | <1101,1112> |
| 2 | Rnd | <1101,1109>,<1110,1116> |
| 3 | A | <1105,1106> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (4, null) | True | For iterations:  Not defined and used  For Rnd:  <1101,1109>,<1110,1116>  For A:  <1105,1106>  It returns true second null value is handled in function |
| **2** | (7,9) | False | For iterations:  <1101,1112>  For Rnd:  <1101,1109>,<1110,1116>  For A:  <1105,1106>  It returns the result false due to its values |

### **Function 10:**

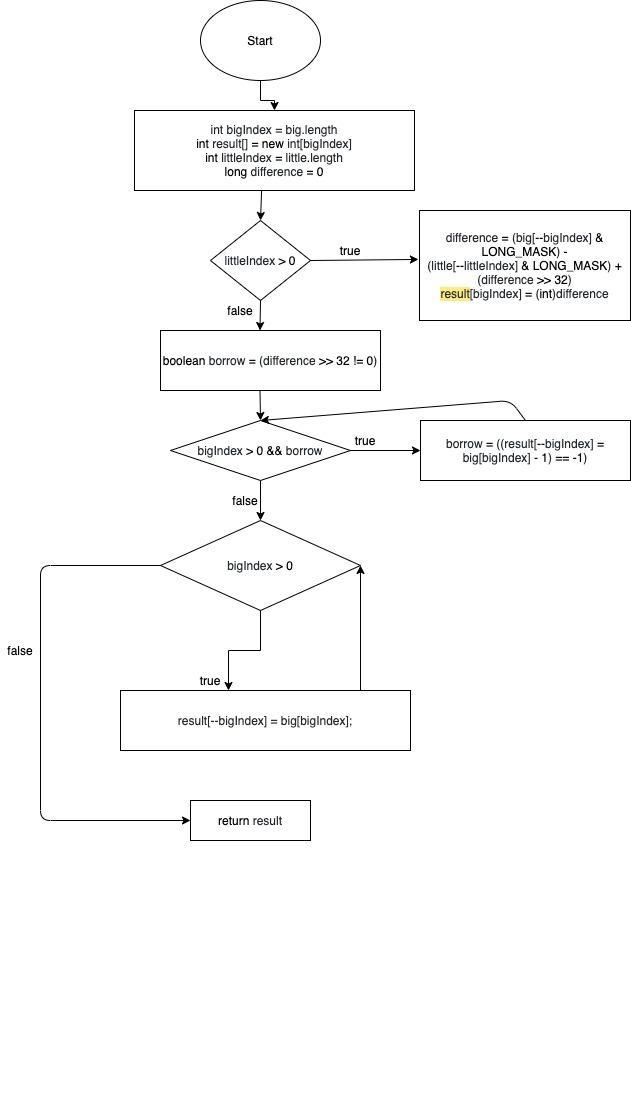
**Source Code:**

****

**Test Case Code:**

****

**CFG:**

****

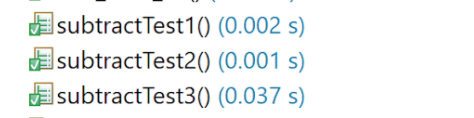
**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | x = {10,20}  y = {30,40} | [30,40] | covers 1549, 1550, 1551, 1552, 1553, 1555,  1563,1564, 1565, 1568 |
| **2** | x={10,20}  y = {} | [10,20] | covers 1549, 1550, 1551, 1552, 1553, 1555,  1563,1564, 1565, 1568,  1569 |
| **3** | x = {10,20,30}  y = {10, 20} | Exception | raises exception |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | x = {10, 20}  y = {30, 40} | [30,40] | covers B1555T,  B1564T, B1568T |
| **2** | x = {10,20}  y = {} | [10,20] | covers B1555F, B1564T, B1568T |
| **3** | x = {10,20,30}  y = {10, 20} | Exception | covers B1555F, B1564F, B1568F |

**Test Result:**



**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | x = {10,20};  y = {30,40} | [30,40] | covers C1555T,  C1564T, C1568T |
| **2** | x={10,20}  y = {} | [10,20] | covers C1555F,  C1564T, C1568T |
| **3** | x = {10,20,30}  y = {10, 20} | Exception | covers C1555F, C1564F, C1568F |

**Boundary Interior:**

**Loop 1:**

1555 -> 1556

1555 -> 1556 -> 1557

1555 -> 1556 -> 1557 -> 1558

1555 -> 1556 -> 1557 -> 1558 -> 1559

1555 -> 1556 -> 1557 -> 1558 -> 1559 -> 1555

**Loop 2:**

1564 -> 1565

1564 -> 1565 - 1564

**Loop 3:**

1568 -> 1569

1568 -> 1569 -> 1568

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | x = {10,20}  y = {30,40} | [30,40] | Covers Loop 2  Covers Loop 1 |
| **2** | x={10,20}  y = {} | [10,20] | Covers Loop 2  Covers Loop 3 |

**Loop Boundary:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | ([0,2], []) | [0,2] | Covers:  Loop 1:  1555T  Loop 2:  1564T  Loop 3:  1568T  When the loop will not execute |
| **2** | ([5],[2]) | [2,4] | loop 1:  1555T  loop 2:  1564T  loop 3:  1568T  Only one iteration |
| **3** | ([10,20], [30,40]) | [30,40] | loop 1:  littleIndex > 0 True  loop 2:  bigIndex > 0 True  loop 3:  bigIndex > 0 True  more than one passes |

**Basis Path:**

No of decision points = 4

No. of basis path = No of decision points +1 = 4+1 = 5

**Path 1:**

1548, 1549, 1550, 1551, 1552, 1555, 1556, 1557, 1558, 1559, 1563, 1571

**Path 2:**

1548, 1549, 1550, 1551, 1552, 1563, 1564, 1565, 1571

**Path 3:**

1548, 1549, 1550, 1551, 1552, 1555, 1556, 1557, 1558, 1559, 1563, 1564, 1565, 1568, 1569, 1571

**Path 4:**

1548, 1549, 1550, 1551, 1552, 1555, 1556, 1557, 1558, 1559, 1563, 1568, 1569, 1571

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | x = {10,20}  y = {30,40} | [30,40] | Covers Path 3 |
| **2** | x={10,20}  y = {} | [10,20] | Covers Path 2 |
| **3** | x={}  y = {10,20} | Exception | Covers Path 1 |
| **4** | x = {10,20,30}  y = {10, 20} | Exception | Covers Path 4 |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | Big | 1548 | 1549, 1556, 1565, 1569 |
| 2 | Little | 1548 | 1551,1556 |
| 3 | Borrow | 1563,1565 | 1564 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | Big | <1548,1549>,<1548,1549><1548,1556><1565,1569> |
| 2 | Little | <1548,1551>,<1548,1556> |
| 3 | Borrow | <1563,1564> |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | x = {10, 20}  y = {30, 40} | [30,40] | [-21,20] | Pass | For big covers  <1548,1549>, <1565,1569>  For little covers  <1548,1551>,  <1548,1556>  For borrow covers  <1563,1564>  It returns true second null value is handled in function |
| **2** | x={10,20}  y = {} | [10,20] | [10,20] | Pass | For big covers  <1548,1549>, <1565,1569>  For little covers  <1548,1551>,  <1548,1556>  For borrow covers  <1563,1564>  It returns the result false due to its values |

**All Test Cases Result:**

