Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software

Testare şi verificare
Proiect individual

- Să se implementeze un program care primeşte de la tastatură un număr n, 1<=n<=10, urmat de n numere naturale naturale şi doi indecşi low şi high.
- Să se întoarcă suma numerelor **Armstrong** dintre cei doi indecşi.
- Se defineşte un număr ca fiind **Armstrong** dacă este format din suma tuturor cifrelor sale ridicate la puterea numărului total de cifre. De exemplu: numerele **4** = 4^1, **371** = 3^3 + 7^3 + 1^3 sunt Armstrong.

Input:

- n - numărul de elemente ale vectorului.
- n - numere naturale mai mari decât 0.
- low - primul indice.
- high - cel de-al doilea indice.

Output:

- un număr s, reprezentând suma numerelor **Armstrong** din vector.

1. **a) Partiţionare de echivalenţă**

Domeniul de intrări:

- 1 <= n <= 10 - lungimea vectorului
- un vector de numere naturale mai mari decât 0 - nenule
- un număr întreg (indice) low
- un număr întreg (indice) high

Distingem următoarele clase de echivalenţă:

Pentru n:
- N_1 = 1...10
- N_2 = {n/ n<1}
- N_3 = {n/ n>10 }

Pentru a:
- A_1 = {a / a conţine valori pozitive}
- A_2 = {a / a conţine cel puţin o valoare negativă}

Pentru low:
- L_1 = {low / 0 <= low < n}
- L_2 = {low / low < 0}
- L_3 = {low / low >= n }

Pentru high:
- H_1 = {high / 0 <= high < n}
- H_2 = {high / high < 0}
- H_3 = {high / high >= n}

Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software

Domeniul de ieşiri:
- s - suma numerelor Armstrong dintre cei doi indici din vector.
- s = 0, dacă vectorul nu conţine niciun număr Armstrong între low şi high.

Clase de echivalenţă globale:

- C_1111 = {(n, a, high, low) / n în N_1, a în A_1, low în L_1, high în H_1}
- C_1112 = {(n, a, high, low) / n în N_1, a în A_1, low în L_1, high în H_2}
- C_1113 = {(n, a, high, low) / n în N_1, a în A_1, low în L_1, high în H_3}
- C_112 = {(n, a, high, low) / n în N_1, a în A_1, low în L_2 }
- C_113 = {(n, a, high, low) / n în N_1, a în A_1, low în L_3 }
- C_12 = {(n, a, high, low) / n în N_1, a în A_2 }
- C_2 = {(n, a, high, low) / n în N_2 }
- C_3 = {(n, a, high, low) / n în N_3}

Date de intrare	Raspuns
T_1111 = {5, {10, 4, 5, 7, 150}, 1, 4}	16
T_1112 = {5, {10, 4, 5, 7, 150}, 1, -1}	Conditions not met
T_1113 = {5, {10, 4, 5, 7, 150}, 1, 6}	Conditions not met
T_112 = {5, {10, 4, 5, 7, 150}, -1, _}	Conditions not met
T_113 = {5, {10, 4, 5, 7, 150}, 6, _}	Conditions not met
T_12 = {5, {10, -1, 5, 7, 150}, _, _}	Conditions not met
T_2 = {0, _, _ , _}	Conditions not met
T_3 = {11, _ , _ , _}	Conditions not met

1. b) Analiza valorilor de frontieră

Dimensiunea vectorului:

- n = 0, 1, 10, 11
- low ∈ {1, 0, n -1, n}
- high ∈ {1, 0, n -1, n}

Distingem următoarele clase din punctul de vedere al valorilor de frontieră:

N_1 = {(n, a, high, low) / n = 1 sau n = 10}
N_2 = {(n, a, high, low) / n = 0}
N_3 = {(n, a, high, low) / n = 11}

A_1 = {a / a conţine valori pozitive}
A_2 = {a / a conţine cel puţin o valoare negativă}

L_1 = {(n, a, high, low) / low = -1}
L_2 = {(n, a, high, low) / low = 0, n-1}
L_3 = {(n, a, high, low) / low = n}

H_1 = {(n, a, high, low) / high = -1}
H_2 = {(n, a, high, low) / high = 0, n-1}
H_3 = {(n, a, high, low) / high = n}

Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software

- C_1111 = {(n, a, high, low) / n în N_1, a în A_1, low în L_1, high în H_1}
- C_1112 = {(n, a, high, low) / n în N_1, a în A_1, low în L_1, high în H_2}
- C_1113 = {(n, a, high, low) / n în N_1, a în A_1, low în L_1, high în H_3}
- C_1122 = {(n, a, high, low) / n în N_1, a în A_1, low în L_2, high în H_2}
- C_1123 = {(n, a, high, low) / n în N_1, a în A_1, low în L_2, high în H_3}
- C_1133 = {(n, a, high, low) / n în N_1, a în A_1, low în L_3, high în H_3}
- C_12 = {(n, a, high, low) / n în N_1, a în A_2 }
- C_2 = {(n, a, high, low) / n în N_2 }
- C_3 = {(n, a, high, low / n în N_3)}

Date de intrare	Raspuns
T_1111 = {1, {153}, -1, -1}	Conditions not met
T_1112 = {1, {153}, -1, 0}	Conditions not met
T_1113 = {10, {1, 1, 1, 1....}, -1, 9}	Conditions not met
T_1122 = {1, {153}, 0, 0 }	153
T_1123 = {10, {1, 1, 1, 1....}, 0, 9}	10
T_1133 = {10, {1, 1, 1, 1....}, 9, 9}	1
T_12 = {1, {-1}, _, _}	Conditions not met
T_2 = {0, _, _, _}	Conditions not met
T_3 = {11, _, _, _}	Conditions not met

1. * Partiţionarea în categorii

Avem următoarea partiționare în unități a problemei:

- public static boolean isArmstrongNumber (int x)
- public static int solve(int n, int [] a, int low, int high)

isArmstrongNumber()

- Categorii: număr Armstrong sau nu
- Întoarce : **True**, dacă numărul dat este număr Armstrong şi **False**, altfel

Date de intrare:
T_1 = 153 => true
T_2 = 22 => false

Categorii:
n: dacă se află în intervalul valid 1.. 10 sau nu
array: dacă conţine elemente negative sau nu
low: {0...n-1}   high: {0...n-1}

Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software

Alternative:

n < 0, n = 0, n = 1, n = 2 .. 10, n = 10, n >11

array: conţine numai numere naturale sau conţine cel puţin o valoare negativă

low:

- low >= 0 && low < n
- low < 0
- low > n
- low = n 1

high:

- analog low

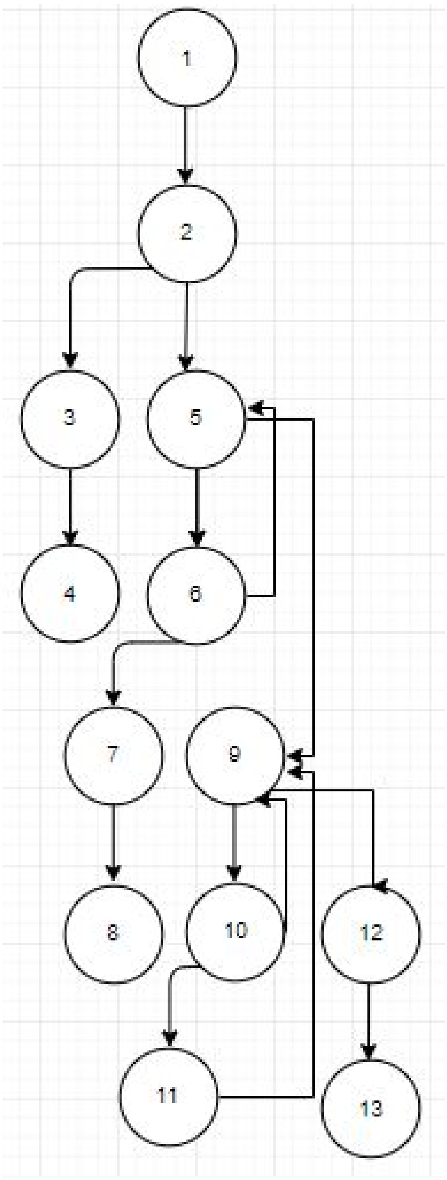
Date de intrare	Raspuns
T_1 = (1, _, _, _)	Conditions not met
T_2 = (0, {}, _, _)	Conditions not met
T_311 = (1, {153}, 0, 0)	153
T_312 = (1, {153}, -1, _)	Conditions not met
T_313 = (1, {153}, 3, _)	Conditions not met
T_3142 = (1, {153}, 0, 1)	Conditions not met
T_4111 = (5, {30, 40, 51, 153, 150}, 3, 4)	153
T_4122 = (5, {30, 40, 51, 153, 150}, 3, [1 | 5 | 6])	Conditions not met
T_5111 = (10, {1, ..., 1}, 0, 9)	10
T_61 = (11, _, _, _)	Conditions not met
T_4113 = (10, {1, …, 1}, 0, 11)	Conditions not met
T_5144 = (10, {1, .., 1}, 9, 9)	1
T_32 = (1, {1}, _, _)	Conditions not met
T_42 = (5, {3, 4, 153, 8, 9}, _, _)	Conditions not met
T_52 = (100, {1, …, 1}, _, _)	Conditions not met

2. Testarea structurală pe baza grafului orientat

```
#   Program
    public class Armstrong{

      public static int solve( int n, int[] a, int low, int high) {
1             int s = 0;
2             if(n < 1 || n > 10 || a == null || low < 0 ||
                 low >= n || high < 0 || high >= n) {
3                System.out.println("Conditions not met.");
4                return -1;
              }
5             for(int i = 0; i < n; ++i) {
6                 if(a[i] < 0) {
7                   System.out.println("Conditions not met.");
8                   return -1;
                  }
```

```
            }

9           for(int i = low; i <= high; ++i) {
10              if(isArmstrongNumber(a[i])) {
11                  s += a[i];
                }
            }
12         System.out.println(s);
13         return s;
        }
    }
#   End of program
```

Graful programului este:



Ceaușescu Ciprian - Mihai
Grupa 506 - Inginerie Software

a) Statement Coverage

Pentru a realiza acoperirea la nivel de instrucțiune, ne concentrăm asupra nodurilor care conțin instrucțiuni. Scopul este acela de a ca fiecare instrucțiune să fie **True**.
Realizăm următoarele teste:

Date de intrare	Raspuns
T_1 = (0, {}, 0, 0)	Conditions not met
T_2 = (6, {3, 153, -5, -153, 370, 407}, 0, 4)	Conditions not met
T_3 = (5, {153, 51, 50, 82, 370 }, 0, 4)	523

b) Branch Coverage

Instrucțiuni care duc la ramuri în program:

if(n < 1 || n > 10 || a == null || low < 0 || low >= n || high < 0 || high >= n)
if(a[i] < 0)
if(*isArmstrongNumber*(a[i]))

Scopul este de a realiza teste care să evalueze fiecare ramură **True**, dar și **False**.
Pentru a testa acoperirea la nivel de ramuri, avem următoarele teste:

Date de intrare	Raspuns
T_1 = (0, {}, 0, 0)	Conditions not met
T_2 = (5, {153, 91,78, 71, 1}, 6, 3)	Conditions not met
T_3 = (5, {153, 91, 78, 71, 1}, 2, -2)	Conditions not met
T_4 = (5, {153, 91, 78, 71, 1}, 2, 10)	Conditions not met
T_5 = (5, {153, -51, -153, 82, 370 }, 2, 4)	Conditions not met
T_6 = (1, {153} 0, 0)	153
T_7 = {1, {91}, 0, 0}	0
T_8 = {5, {153, 31, 3, 4, 5},0, 1}	153

c) Condition coverage

Deciziile din programul Java:

Decizii	**Conditii individuale**
if (n < 1 || n > 10 || a == null || low < 0 || low >= n || high < 0 || high >= n)	**n < 1** **n > 10** **low < 0** **low >= n** **high < 0** **high >= n**
for (int i = 0 ; i < n ; ++i)	**i < n**
if(a[i] < 0)	**a[i] < 0**
for(int i = low; i <= high; ++i)	**i >= low**

	i <= high
if(*isArmstrongNumber*(a[i]))	***isArmstrongNumber*(a[i]) == true**

Pentru a acoperi toate condiţiile din setul de mai sus, folosim următoarea suită de teste:

Date de intrare	**Raspuns**
T_1 = (0, {}, 0, 0)	Conditions not met
T_2 = (11, {}, 0 , 1)	Conditions not met
T_3 = (3, {153, 91, 78}, -5, 0)	Conditions not met
T_4 = (3, {153, 91, 78}, 10, 2)	Conditions not met
T_5 = (3, {153, 91, 78 }, 2, -5)	Conditions not met
T_6 = (1, {153, 91, 78}, 2, 10)	Conditions not met
T_7 = {1, {-1}, 0, 0}	Conditions not met
T_8 = {1, {31}, 0, 0}	0
T_9 = {1, {153}, 0, 0}	153
T_10 = {3, {21, 153, 370}}	523

d) Modified condition/decision

Pentru condiţia: **if (n < 1 || n > 10 || a == null || low < 0 || low >= n || high < 0 || high >= n),** avem următorul tabel:

n < 1	**n > 10**	**a == null**	**low < 0**	**low >= n**	**high < 0**	**high >= n**	**Rezultat**
false	false	false	false	false	false	false	false
true	false	false	false	false	false	false	true
false	true	false	false	false	false	false	true
false	false	true	false	false	false	false	true
false	false	false	true	false	false	false	true
false	false	false	false	true	false	false	true
false	false	false	false	false	true	false	true
false	false	false	false	false	false	true	true

Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software

Date de intrare	Raspuns
T_1 = {1, {153}, 0, 0}	153
T_2 = {-1, {153}, 0, 0}	Conditions not met
T_3 = {11, {153}, 0, 0}	Conditions not met
T_4 = {1, null, 0, 0}	Conditions not met
T_5 = {1, {153}, -1, 0}	Conditions not met
T_6 = {1, {153}, 1, 0}	Conditions not met
T_7 = {1, {153}, 0, -1}	Conditions not met
T_8 = {1, {31}, 0, 1}	Conditions not met

3. Complexitatea programului

Formula lui McCabe pentru complexitatea ciclomatică este:
Dat fiind un graf complet conectat G cu e arce şi n noduri, atunci numărul de circuite linear independente este dat de:
V(G) = e - n + 1, unde:

- **G este graful complet conectat (exista o cale intre oricare doua noduri)**
- **Circuitul este calea care începe şi se termină în acelaşi nod**
- **Circuite linear independente, fiecare dintre acestea nu poate fi obţinut ca o combinaţie a celorlalte**

Adaugăm următoarele muchii în graful de mai sus pentru a deveni complet conectat:
(4,1) , (8,1) şi (13, 1)
Atunci: V(G) = 18 - 13 + 1 = 6
Circuite independente:

- 1 -> 2 -> 3-> 4 -> 1
- 1 -> 2 -> 5 -> 6 -> 7 -> 8 -> 1
- 1 -> 2 -> 5 -> 9 -> 12 -> 13 -> 1
- 9 -> 10 -> 9
- 9 -> 10 -> 11 -> 9
- 5 -> 6 -> 5

4. Acoperirea la nivel de cale

Căile posibile sunt:

1.2.3.4

1.2.5.6.7.8

1.2.5.9.10.11

1.2.5.9.12.13

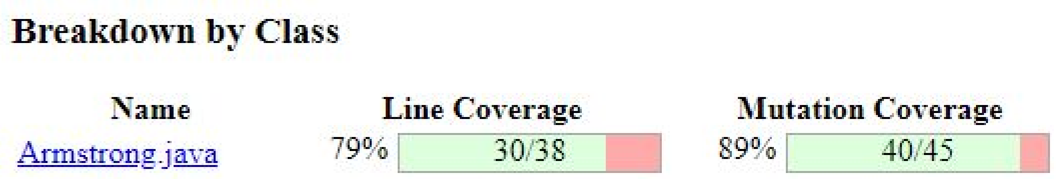
Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software

Date de test:

Date de intrare	Raspuns
T_1 = (0, {}, 0, 0)	Conditions not met
T_2 = (3, {-1, 1,0}, 0, 1)	Conditions not met
T_3 = (3, {155, 91, 78}, 0, 1)	0
T_4 = (3, {155, 91, 153}, 10, 2)	153

5. Generatorul de mutanti

Pentru generarea mutanţilor s-a folosit pluginul PIT pentru IntelliJ:

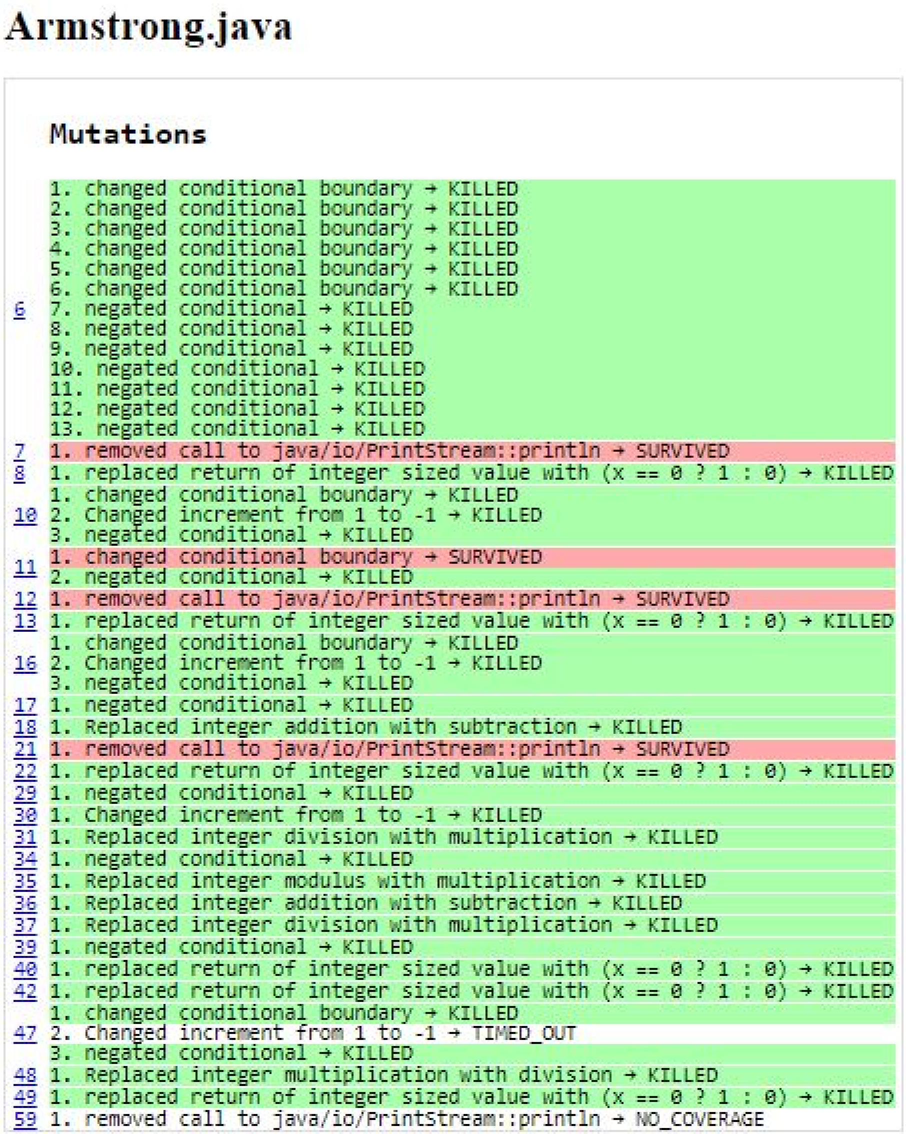


Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Armstrong.java	79%	30/38	89%	40/45

Observăm faptul că testele omoară 40 din cei 45 mutanţi generaţi.

Ceauşescu Ciprian - Mihai
Grupa 506 - Inginerie Software



Armstrong.java

```
   Mutations

   1. changed conditional boundary → KILLED
   2. changed conditional boundary → KILLED
   3. changed conditional boundary → KILLED
   4. changed conditional boundary → KILLED
   5. changed conditional boundary → KILLED
   6. changed conditional boundary → KILLED
6  7. negated conditional → KILLED
   8. negated conditional → KILLED
   9. negated conditional → KILLED
   10. negated conditional → KILLED
   11. negated conditional → KILLED
   12. negated conditional → KILLED
   13. negated conditional → KILLED
7  1. removed call to java/io/PrintStream::println → SURVIVED
8  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
   1. changed conditional boundary → KILLED
10 2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
   1. changed conditional boundary → SURVIVED
11 2. negated conditional → KILLED
12 1. removed call to java/io/PrintStream::println → SURVIVED
13 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
   1. changed conditional boundary → KILLED
16 2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
17 1. negated conditional → KILLED
18 1. Replaced integer addition with subtraction → KILLED
21 1. removed call to java/io/PrintStream::println → SURVIVED
22 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
29 1. negated conditional → KILLED
30 1. Changed increment from 1 to -1 → KILLED
31 1. Replaced integer division with multiplication → KILLED
34 1. negated conditional → KILLED
35 1. Replaced integer modulus with multiplication → KILLED
36 1. Replaced integer addition with subtraction → KILLED
37 1. Replaced integer division with multiplication → KILLED
39 1. negated conditional → KILLED
40 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
42 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
   1. changed conditional boundary → KILLED
47 2. Changed increment from 1 to -1 → TIMED_OUT
   3. negated conditional → KILLED
48 1. Replaced integer multiplication with division → KILLED
49 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
59 1. removed call to java/io/PrintStream::println → NO_COVERAGE
```

6. Omorârea mutanţilor

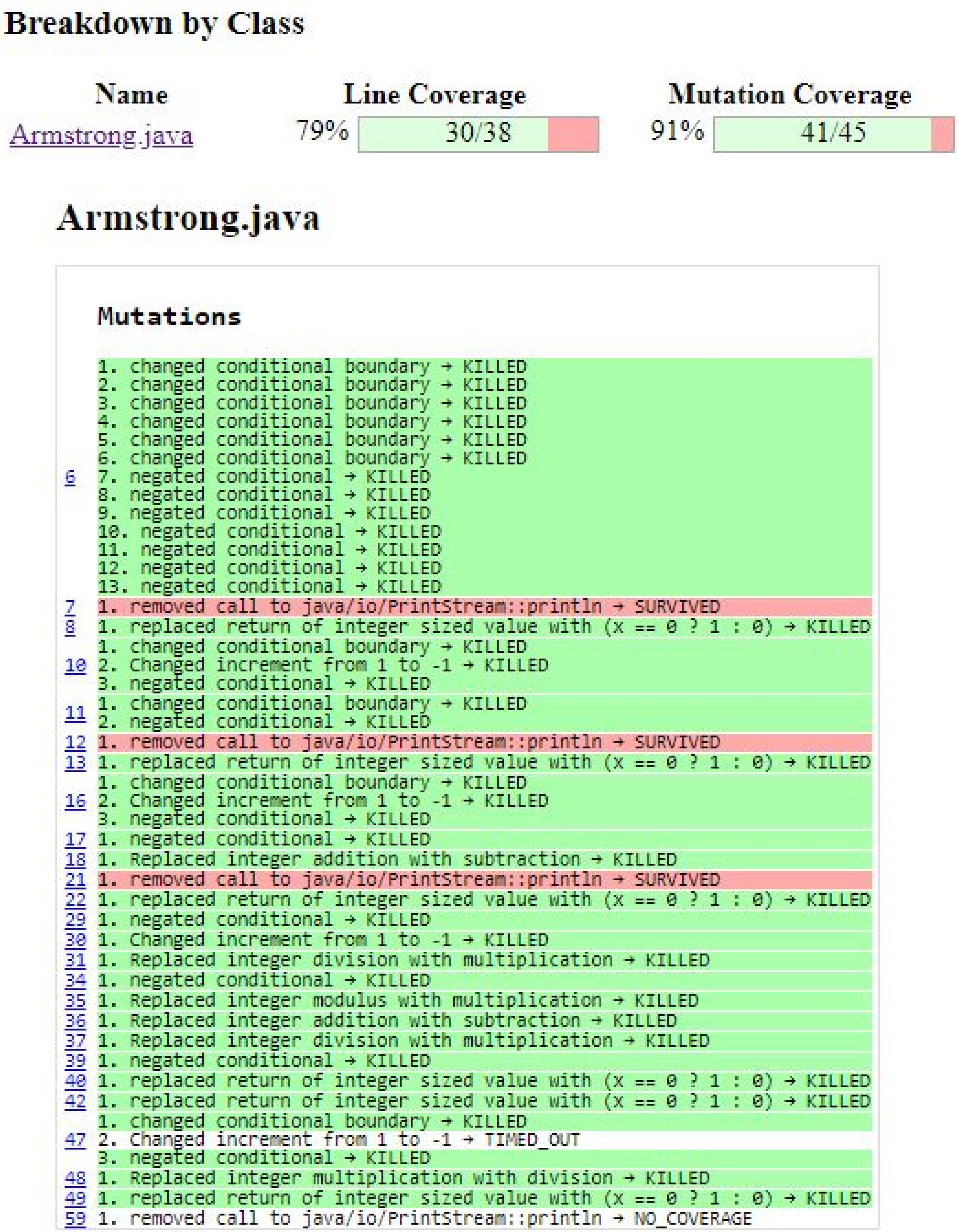
Unul dintre mutanţii neomorâţi este un ConditionalsBoundaryMutator generat la linia (11) din cod:

Original conditional	**Mutated conditional**
If (a[i] < 0)	**If (a[i] <= 0)**

Pentru a omorî acest mutant, adăugăm un test care sa conţină elemente nule în vector:

int []d = {0,1,2};
assertEquals(0,Arms.solve(1,d, 0, 0));

În urma adăugării testelor, observăm faptul că am omorât mutantul:



Breakdown by Class

Name	Line Coverage	Mutation Coverage
Armstrong.java	79% 30/38	91% 41/45

Armstrong.java

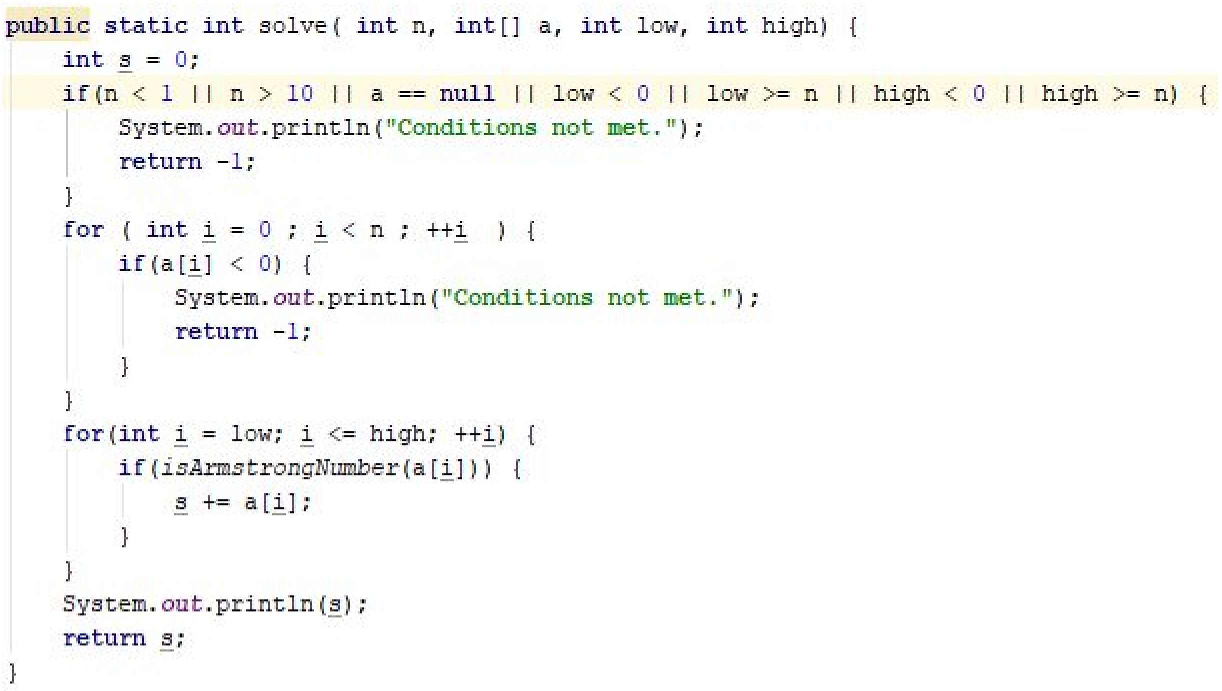
Mutations

```
   1. changed conditional boundary → KILLED
   2. changed conditional boundary → KILLED
   3. changed conditional boundary → KILLED
   4. changed conditional boundary → KILLED
   5. changed conditional boundary → KILLED
   6. changed conditional boundary → KILLED
6  7. negated conditional → KILLED
   8. negated conditional → KILLED
   9. negated conditional → KILLED
   10. negated conditional → KILLED
   11. negated conditional → KILLED
   12. negated conditional → KILLED
   13. negated conditional → KILLED
7  1. removed call to java/io/PrintStream::println → SURVIVED
8  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
   1. changed conditional boundary → KILLED
10 2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
11 1. changed conditional boundary → KILLED
   2. negated conditional → KILLED
12 1. removed call to java/io/PrintStream::println → SURVIVED
13 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
   1. changed conditional boundary → KILLED
16 2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
17 1. negated conditional → KILLED
18 1. Replaced integer addition with subtraction → KILLED
21 1. removed call to java/io/PrintStream::println → SURVIVED
22 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
29 1. negated conditional → KILLED
30 1. Changed increment from 1 to -1 → KILLED
31 1. Replaced integer division with multiplication → KILLED
34 1. negated conditional → KILLED
35 1. Replaced integer modulus with multiplication → KILLED
36 1. Replaced integer addition with subtraction → KILLED
37 1. Replaced integer division with multiplication → KILLED
39 1. negated conditional → KILLED
40 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
42 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
   1. changed conditional boundary → KILLED
47 2. Changed increment from 1 to -1 → TIMED_OUT
   3. negated conditional → KILLED
48 1. Replaced integer multiplication with division → KILLED
49 1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
59 1. removed call to java/io/PrintStream::println → NO_COVERAGE
```

Ceaușescu Ciprian - Mihai
Grupa 506 - Inginerie Software

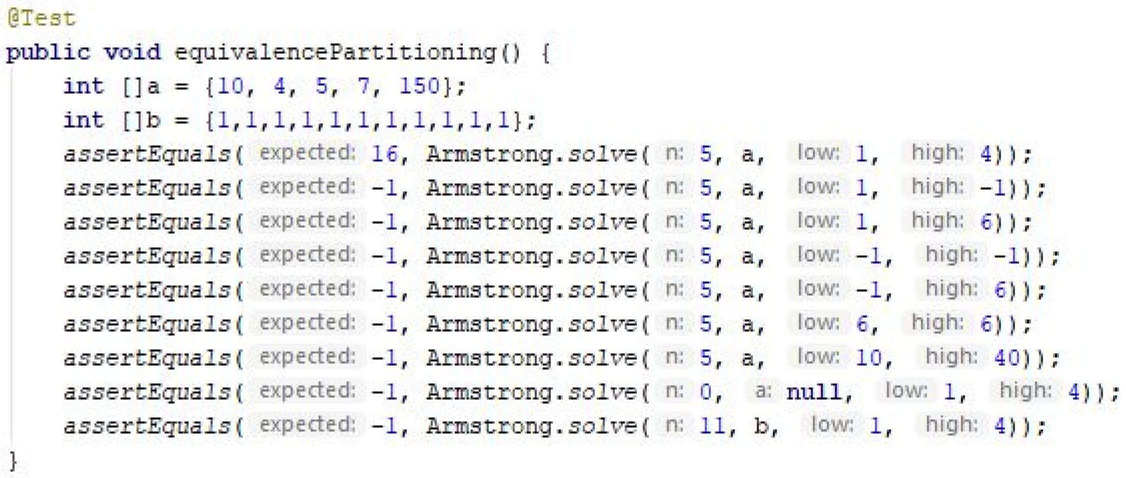
Anexe

Metoda solve - principala metodă a clasei Armstrong

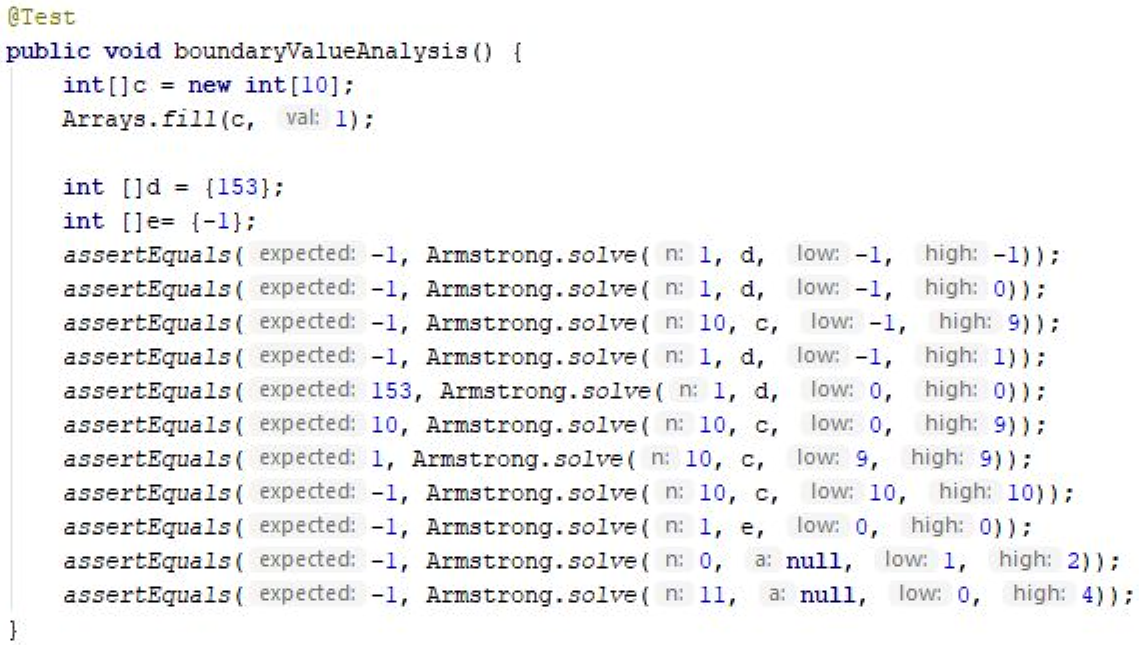


```
public static int solve( int n, int[] a, int low, int high) {
    int s = 0;
    if(n < 1 || n > 10 || a == null || low < 0 || low >= n || high < 0 || high >= n) {
        System.out.println("Conditions not met.");
        return -1;
    }
    for ( int i = 0 ; i < n ; ++i  ) {
        if(a[i] < 0) {
            System.out.println("Conditions not met.");
            return -1;
        }
    }
    for(int i = low; i <= high; ++i) {
        if(isArmstrongNumber(a[i])) {
            s += a[i];
        }
    }
    System.out.println(s);
    return s;
}
```

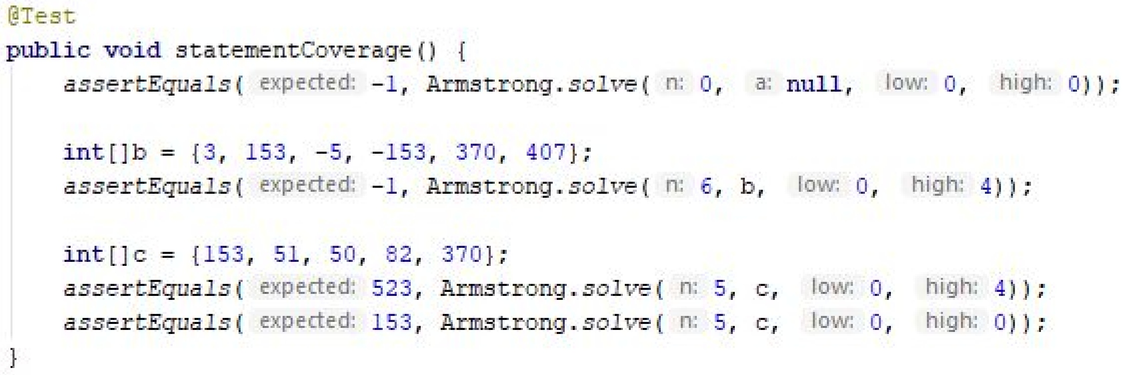
Teste



```
@Test
public void equivalencePartitioning() {
    int []a = {10, 4, 5, 7, 150};
    int []b = {1,1,1,1,1,1,1,1,1,1,1};
    assertEquals( expected: 16, Armstrong.solve( n: 5, a,  low: 1,  high: 4));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 1,  high: -1));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 1,  high: 6));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: -1,  high: -1));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: -1,  high: 6));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 6,  high: 6));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 10,  high: 40));
    assertEquals( expected: -1, Armstrong.solve( n: 0,  a: null,  low: 1,  high: 4));
    assertEquals( expected: -1, Armstrong.solve( n: 11, b,  low: 1,  high: 4));
}
```

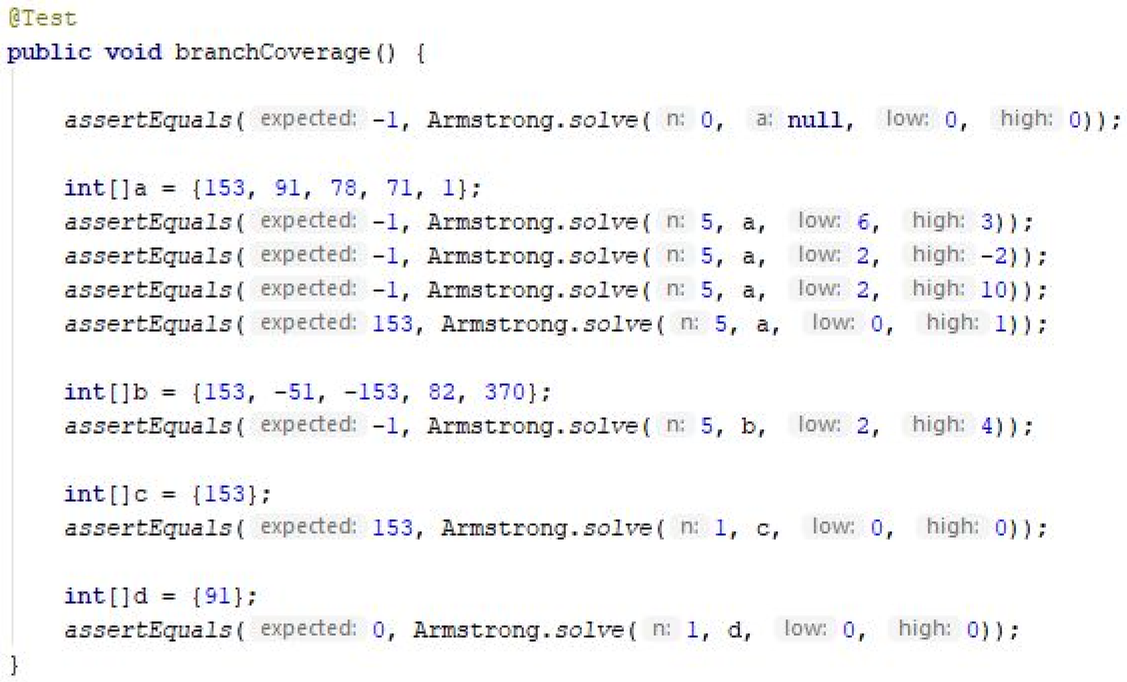


```
@Test
public void boundaryValueAnalysis() {
    int[]c = new int[10];
    Arrays.fill(c,  val: 1);

    int []d = {153};
    int []e= {-1};
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: -1,  high: -1));
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: -1,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 10, c,  low: -1,  high: 9));
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: -1,  high: 1));
    assertEquals( expected: 153, Armstrong.solve( n: 1, d,  low: 0,  high: 0));
    assertEquals( expected: 10, Armstrong.solve( n: 10, c,  low: 0,  high: 9));
    assertEquals( expected: 1, Armstrong.solve( n: 10, c,  low: 9,  high: 9));
    assertEquals( expected: -1, Armstrong.solve( n: 10, c,  low: 10,  high: 10));
    assertEquals( expected: -1, Armstrong.solve( n: 1, e,  low: 0,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 0,  a: null,  low: 1,  high: 2));
    assertEquals( expected: -1, Armstrong.solve( n: 11,  a: null,  low: 0,  high: 4));
}
```

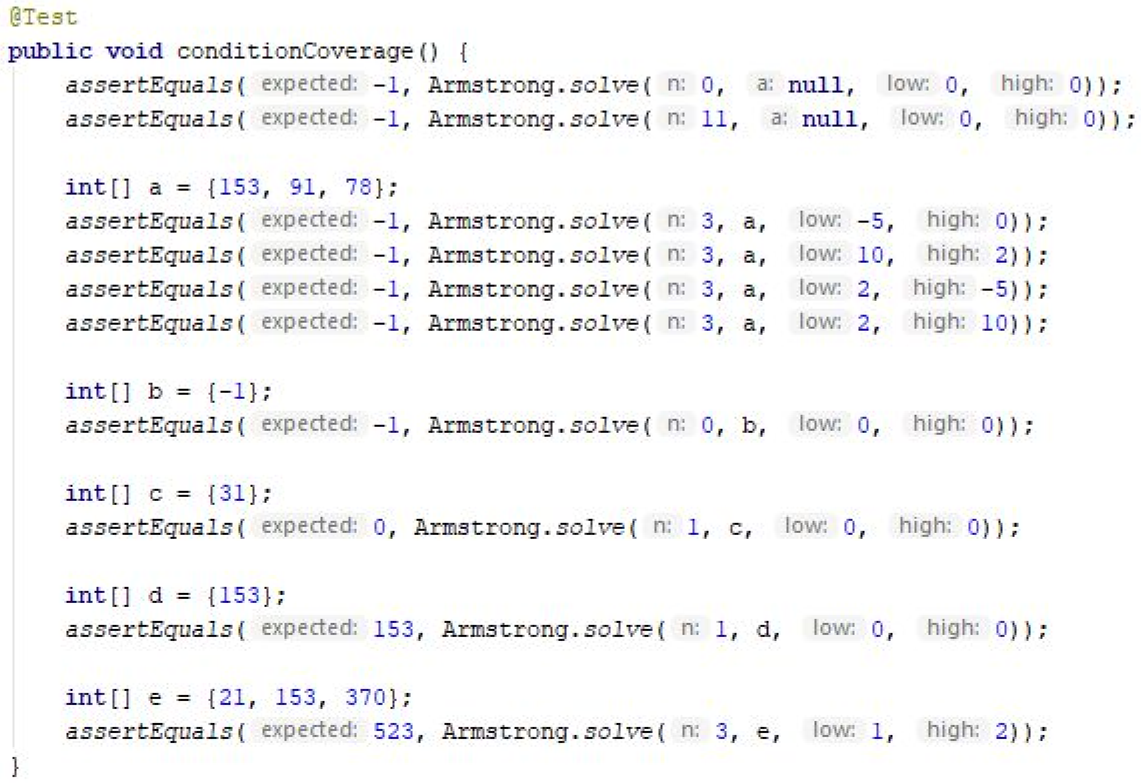


```
@Test
public void statementCoverage() {
    assertEquals( expected: -1, Armstrong.solve( n: 0,  a: null,  low: 0,  high: 0));

    int[]b = {3, 153, -5, -153, 370, 407};
    assertEquals( expected: -1, Armstrong.solve( n: 6, b,  low: 0,  high: 4));

    int[]c = {153, 51, 50, 82, 370};
    assertEquals( expected: 523, Armstrong.solve( n: 5, c,  low: 0,  high: 4));
    assertEquals( expected: 153, Armstrong.solve( n: 5, c,  low: 0,  high: 0));
}
```

Ceaușescu Ciprian - Mihai
Grupa 506 - Inginerie Software

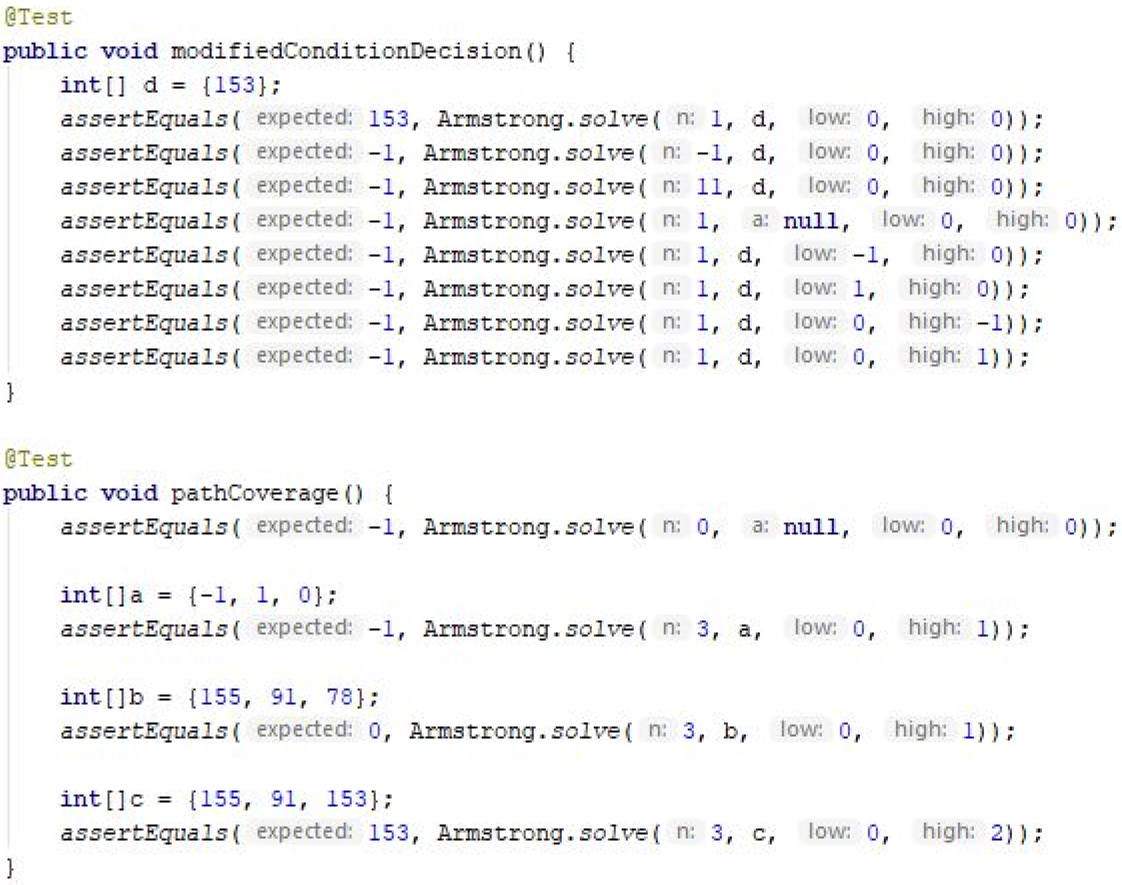


```
@Test
public void branchCoverage() {

    assertEquals( expected: -1, Armstrong.solve( n: 0,  a: null,  low: 0,  high: 0));

    int[]a = {153, 91, 78, 71, 1};
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 6,  high: 3));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 2,  high: -2));
    assertEquals( expected: -1, Armstrong.solve( n: 5, a,  low: 2,  high: 10));
    assertEquals( expected: 153, Armstrong.solve( n: 5, a,  low: 0,  high: 1));

    int[]b = {153, -51, -153, 82, 370};
    assertEquals( expected: -1, Armstrong.solve( n: 5, b,  low: 2,  high: 4));

    int[]c = {153};
    assertEquals( expected: 153, Armstrong.solve( n: 1, c,  low: 0,  high: 0));

    int[]d = {91};
    assertEquals( expected: 0, Armstrong.solve( n: 1, d,  low: 0,  high: 0));
}
```



```
@Test
public void conditionCoverage() {
    assertEquals( expected: -1, Armstrong.solve( n: 0,  a: null,  low: 0,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 11,  a: null,  low: 0,  high: 0));

    int[] a = {153, 91, 78};
    assertEquals( expected: -1, Armstrong.solve( n: 3, a,  low: -5,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 3, a,  low: 10,  high: 2));
    assertEquals( expected: -1, Armstrong.solve( n: 3, a,  low: 2,  high: -5));
    assertEquals( expected: -1, Armstrong.solve( n: 3, a,  low: 2,  high: 10));

    int[] b = {-1};
    assertEquals( expected: -1, Armstrong.solve( n: 0, b,  low: 0,  high: 0));

    int[] c = {31};
    assertEquals( expected: 0, Armstrong.solve( n: 1, c,  low: 0,  high: 0));

    int[] d = {153};
    assertEquals( expected: 153, Armstrong.solve( n: 1, d,  low: 0,  high: 0));

    int[] e = {21, 153, 370};
    assertEquals( expected: 523, Armstrong.solve( n: 3, e,  low: 1,  high: 2));
}
```

Ceaușescu Ciprian - Mihai
Grupa 506 - Inginerie Software



```
@Test
public void modifiedConditionDecision() {
    int[] d = {153};
    assertEquals( expected: 153, Armstrong.solve( n: 1, d,  low: 0,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: -1, d,  low: 0,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 11, d,  low: 0,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 1,  a: null,  low: 0,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: -1,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: 1,  high: 0));
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: 0,  high: -1));
    assertEquals( expected: -1, Armstrong.solve( n: 1, d,  low: 0,  high: 1));
}

@Test
public void pathCoverage() {
    assertEquals( expected: -1, Armstrong.solve( n: 0,  a: null,  low: 0,  high: 0));

    int[]a = {-1, 1, 0};
    assertEquals( expected: -1, Armstrong.solve( n: 3, a,  low: 0,  high: 1));

    int[]b = {155, 91, 78};
    assertEquals( expected: 0, Armstrong.solve( n: 3, b,  low: 0,  high: 1));

    int[]c = {155, 91, 153};
    assertEquals( expected: 153, Armstrong.solve( n: 3, c,  low: 0,  high: 2));
}
```