System Test Plan
Logic Based Testing

By: Cail Keeling

Table of Contents

1. Determination for findEven Predicates

1.1. Predicate 1

Predicate 1: i < digits.length – 2
Clauses:

- C1 = i < digits.length – 2

Reachability:
r(p1) = true (always reached)

$P1_{C1}$ determination = P1=C1

C1	P1
T	T
F	F

Predicate Table (PT) 1

Row ID	**C_1**	**P_1**	**$P1_{C1}$**
1	T	T	T
2	F	F	T

1.2. Predicate 2

Predicate 2: j < digits.length – 1
Clauses:

- C2 = j < digits.length – 1

Reachability:
r(p2) = r(p1) && p1
= i < digits.length – 2

$P2_{C2}$ determination = P2=C2

C2	P2
T	T
F	F

Predicate Table (PT) 2

Row ID	**C2**	**P2**	**$P2_{C2}$**
1	T	T	T
2	F	F	T

1.3. Predicate 3

Predicate 1: k < digits.length
Clauses:

- C3 = k < digits.length

Reachability:
r(p3) = r(p2) && p2
= (i < digits.length – 2) && (j < digits.length – 1)

P3$_{C3}$ determination = P3=C3

C3	P3
T	T
F	F

Predicate Table (PT) 3

Row ID	**C3**	**P3**	**P3$_{C3}$**
1	T	T	T
2	F	F	T

1.4. Predicate 4

Predicate 1: i==j || j==k || i==k
Clauses:

- C4 = i==j
- C5 = j==k
- C6 = i==k

Reachability:
r(p4) = r(p3) && p3
= (i < digits.length – 2) && (j < digits.length – 1) && (k < digits.length)

P4$_{C4}$ determination =
P4 $_{C4}$ = (T v C5 v C6) ⊕ (F v C5 v C6)
P4 $_{C4}$ = (T v (C5 v C6)) ⊕ (F v C5 v C6) Associative Law
P4 $_{C4}$ = T ⊕ (C5 v C6) Domination and Identify Laws
P4 $_{C4}$ = ¬ (C5 v C6) Identify Law
P4 $_{C4}$ = ¬ C5 ^ ¬ C6 De Morgan's Laws

P4$_{C5}$ determination =
P4 $_{C5}$ = (C4 v T v C6) ⊕ (C4 v F v C6)
P4 $_{C5}$ = (T v (C4 v C6)) ⊕ (F v (C4 v C6)) Associative Law
P4 $_{C5}$ = T ⊕ (C4 v C6) Domination and Identify Laws
P4 $_{C5}$ = ¬ (C4 v C6) Identify Law
P4 $_{C5}$ = ¬ C4 ^ ¬ C6 De Morgan's Laws
P4$_{C6}$ determination =
P4 $_{C6}$ = (C4 v C5 v T) ⊕ (C4 v C5 v F)
P4 $_{C6}$ = (T v (C4 v C5)) ⊕ (F v (C4 v C5)) Associative Law

$P4_{C6} = T \oplus (C4 \vee C5)$ Domination and Identify Laws
$P4_{C6} = \neg (C4 \vee C5)$ Identify Law
$P4_{C6} = \neg C4 \wedge \neg C5$ De Morgan's Law

Predicate Table (PT) 4

Row ID	C_4	C_5	C_6	P4	$P4_{C4}$	$P4_{C5}$	$P4_{C6}$
1	T	T	T	T	F	F	F
2	T	T	F	T	F	F	F
3	T	F	T	T	F	F	F
4	T	F	F	T	T	F	F
5	F	T	T	T	F	F	F
6	F	T	F	T	F	T	F
7	F	F	T	T	F	F	T
8	F	F	F	F	T	T	T

1.5. Predicate 5

Predicate 5: num%2==0
Clauses:

- C7 = num%2==0

Reachability:
r(p5) = r(p4) && !p4
= (i < digits.length – 2) && (j < digits.length – 1) && (k < digits.length) && !(i==j || j==k || i==k)

P_{C7} determination = P5=C7

C7	P5
T	T
F	F

Predicate Table (PT) 5

Row ID	C7	P5	$P5_{C7}$
1	T	T	T
2	F	F	T

2. CACC Test Requirements

TR	PT ID	Row	Coverage	Feasible?
1	1	1	P_{C1} where P_1 = True	Y
2	1	2	P_{C1} where P_1 = False	Y
3	2	1	P_{C2} where P_2 = True	Y
4	2	2	P_{C2} where P_2 = False	N – If P1 was false, then the for loop prior to this condition would break, and if this condition was false, then P1 would have to have been false too.
5	3	1	P_{C3} where P_3 = True	Y
6	3	2	P_{C3} where P_3 = False	N – If P1 was false, then the for loop prior to this condition would break, and if this condition was false, then P1 would have to have been false too.
7	4	4	P_{C4} where P_4 = True	N – integers i, j, and k are all designated to be 1+ their predecessor, so they will never equal each other.
8	4	6	P_{C5} where P_4 = True	N – integers i, j, and k are all designated to be 1+ their predecessor, so they will never equal each other.
9	4	7	P_{C6} where P_4 = True	N – integers i, j, and k are all designated to be 1+ their predecessor, so they will never equal each other.
10	4	8	$P_{C4,5,6}$ where P_4 = False	Y
11	5	1	P_{C7} where P_5 = True	Y
12	5	2	P_{C7} where P_5 = False	Y

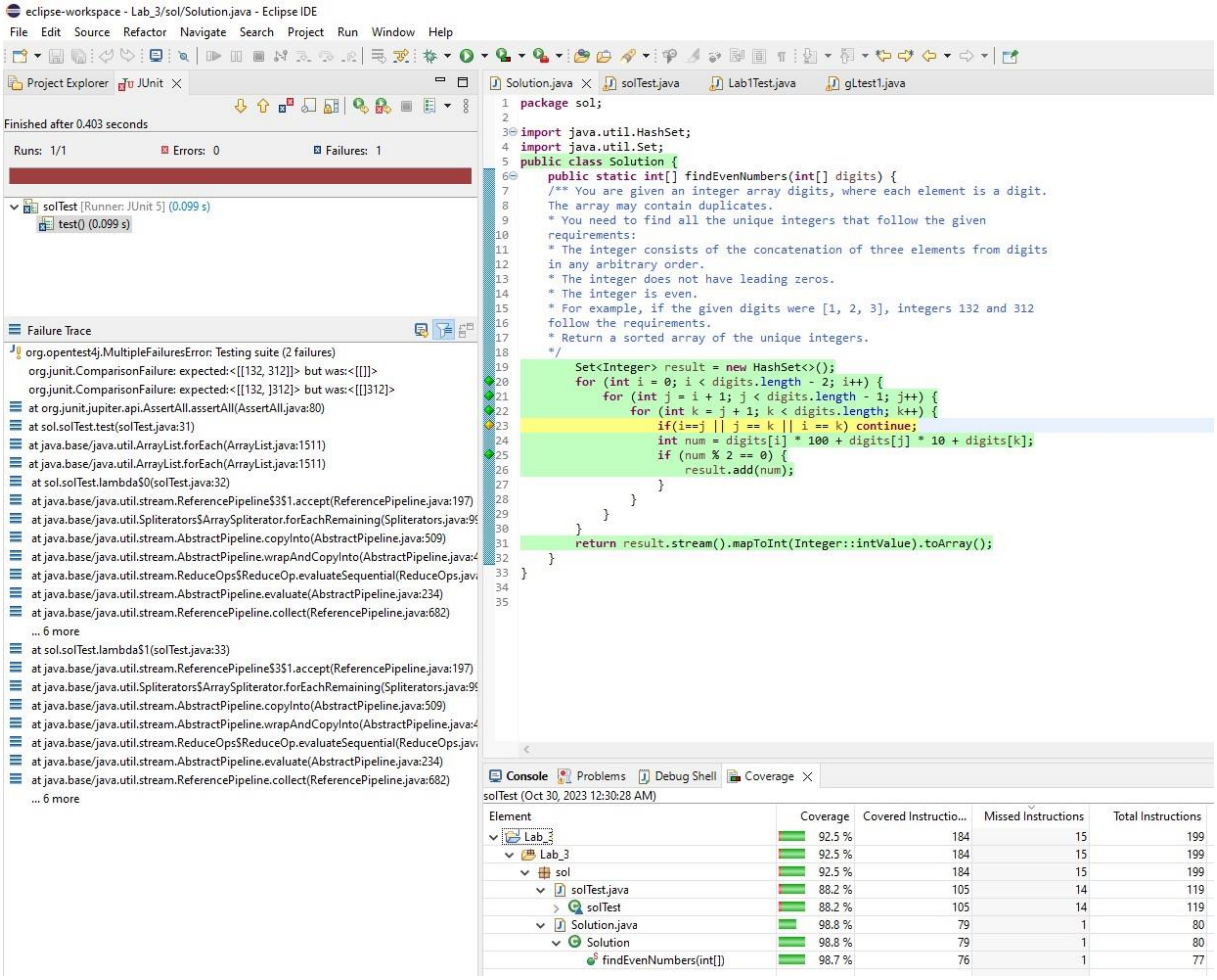
3. Test Results with Traceability

Test ID	Targeted TR	Input	Observed Output	Result
1	1,3,5,10,12	[1,2,3]	[]	Failed
2	1,3,5,10,11	[3,1,2]	[312]	Failed
3	2	[1]	[]	Passed

4. Test Results with Traceability

4.1. Coverage of Logic-Based Test Suite

I do not believe that coverage can be improved, because predicate 4 (i == j || …) will never be triggered. During the initial run, i will be 0, then j will be 1, and k will be 2. The k loop will run until completion, to which it goes back to j, which increments to 2, and then the k loop starts again where it is set to j + 1, which is 2+1=3. This repeats for every iterance of i, j, and k and therefore P4 never ends up being true.



4.2. Faults, Errors and Failures

Fault: Because j and k never get the chance to start at 0, and because they are always 1 value higher than their predecessor, the code will never find all of the integers that meet the requirements.

Failure: Yes, my Junit tests produced 2 failures.