

# **System Test Plan Logic Based Testing**

By: Cail Keeling

## Table of Contents

<b>1.</b>	<b>DETERMINATION FOR FINDEVEN PREDICATES .....</b>	<b>3</b>
1.1.	PREDICATE 1 .....	3
<b>2.</b>	<b>CACC TEST REQUIREMENTS .....</b>	<b>6</b>
<b>3.</b>	<b>TEST RESULTS WITH TRACEABILITY .....</b>	<b>7</b>
<b>4.</b>	<b>TEST RESULTS WITH TRACEABILITY .....</b>	<b>8</b>
4.1.	COVERAGE OF LOGIC-BASED TEST SUITE.....	8
4.2.	FAULTS, ERRORS AND FAILURES.....	8

## 1. Determination for findEven Predicates

### 1.1. Predicate 1

**Predicate 1:**  $i < \text{digits.length} - 2$

**Clauses:**

- $C1 = i < \text{digits.length} - 2$

**Reachability:**

$r(p1) = \text{true}$  (always reached)

**P1C1 determination = P1=C1**

C1	P1
T	T
F	F

Predicate Table (PT) 1

Row ID	C1	P1	P1C1
1	T	T	T
2	F	F	T

### 1.2. Predicate 2

**Predicate 2:**  $j < \text{digits.length} - 1$

**Clauses:**

- $C2 = j < \text{digits.length} - 1$

**Reachability:**

$r(p2) = r(p1) \ \&\& \ p1$   
 $= i < \text{digits.length} - 2$

**P2C2 determination = P2=C2**

C2	P2
T	T
F	F

Predicate Table (PT) 2

Row ID	C2	P2	P2C2
1	T	T	T
2	F	F	T

### 1.3. Predicate 3

**Predicate 1:**  $k < \text{digits.length}$

**Clauses:**

- $C3 = k < \text{digits.length}$

**Reachability:**

$$\begin{aligned} r(p3) &= r(p2) \ \&\& \ p2 \\ &= (i < \text{digits.length} - 2) \ \&\& \ (j < \text{digits.length} - 1) \end{aligned}$$

**P3<sub>C3</sub> determination = P3=C3**

C3	P3
T	T
F	F

Predicate Table (PT) 3

Row ID	C3	P3	P3 <sub>C3</sub>
1	T	T	T
2	F	F	T

### 1.4. Predicate 4

**Predicate 1:**  $i==j \ \parallel \ j==k \ \parallel \ i==k$

**Clauses:**

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

**Reachability:**

$$\begin{aligned} r(p4) &= r(p3) \ \&\& \ p3 \\ &= (i < \text{digits.length} - 2) \ \&\& \ (j < \text{digits.length} - 1) \ \&\& \ (k < \text{digits.length}) \end{aligned}$$

**P4<sub>C4</sub> determination =**

$$P4_{C4} = (T \vee C5 \vee C6) \oplus (F \vee C5 \vee C6)$$

$$P4_{C4} = (T \vee (C5 \vee C6)) \oplus (F \vee C5 \vee C6) \quad \text{Associative Law}$$

$$P4_{C4} = T \oplus (C5 \vee C6) \quad \text{Domination and Identify Laws}$$

$$P4_{C4} = \neg (C5 \vee C6) \quad \text{Identify Law}$$

$$P4_{C4} = \neg C5 \wedge \neg C6 \quad \text{De Morgan's Laws}$$

**P4<sub>C5</sub> determination =**

$$P4_{C5} = (C4 \vee T \vee C6) \oplus (C4 \vee F \vee C6)$$

$$P4_{C5} = (T \vee (C4 \vee C6)) \oplus (F \vee (C4 \vee C6)) \quad \text{Associative Law}$$

$$P4_{C5} = T \oplus (C4 \vee C6) \quad \text{Domination and Identify Laws}$$

$$P4_{C5} = \neg (C4 \vee C6) \quad \text{Identify Law}$$

$$P4_{C5} = \neg C4 \wedge \neg C6 \quad \text{De Morgan's Laws}$$

**P4<sub>C6</sub> determination =**

$$P4_{C6} = (C4 \vee C5 \vee T) \oplus (C4 \vee C5 \vee F)$$

$$P4_{C6} = (T \vee (C4 \vee C5)) \oplus (F \vee (C4 \vee C5)) \quad \text{Associative Law}$$

$P4_{C6} = T \oplus (C4 \vee C5)$   
 $P4_{C6} = \neg (C4 \vee C5)$   
 $P4_{C6} = \neg C4 \wedge \neg C5$

Domination and Identify Laws  
 Identify Law  
 De Morgan's Law

Predicate Table (PT) 4

Row ID	C4	C5	C6	P4	P4C4	P4C5	P4C6
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:**  $\text{num} \% 2 == 0$

**Clauses:**

- $C7 = \text{num} \% 2 == 0$

**Reachability:**

$r(p5) = r(p4) \ \&\& \ !p4$   
 $= (i < \text{digits.length} - 2) \ \&\& \ (j < \text{digits.length} - 1) \ \&\& \ (k < \text{digits.length}) \ \&\& \ !(i == j \ \parallel \ j == k \ \parallel \ i == k)$

**P<sub>C7</sub> determination = P5=C7**

C7	P5
T	T
F	F

Predicate Table (PT) 5

Row ID	C7	P5	P5C7
1	T	T	T
2	F	F	T

## 2. CACC Test Requirements

TR	PT ID	Row	Coverage	Feasible?
1	1	1	P <sub>C1</sub> where P <sub>1</sub> = True	Y
2	1	2	P <sub>C1</sub> where P <sub>1</sub> = False	Y
3	2	1	P <sub>C2</sub> where P <sub>2</sub> = True	Y
4	2	2	P <sub>C2</sub> where P <sub>2</sub> = 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 <sub>C3</sub> where P <sub>3</sub> = True	Y
6	3	2	P <sub>C3</sub> where P <sub>3</sub> = 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 <sub>C4</sub> where P <sub>4</sub> = 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 <sub>C5</sub> where P <sub>4</sub> = 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 <sub>C6</sub> where P <sub>4</sub> = 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 <sub>C4,5,6</sub> where P <sub>4</sub> = False	Y
11	5	1	P <sub>C7</sub> where P <sub>5</sub> = True	Y
12	5	2	P <sub>C7</sub> where P <sub>5</sub> = 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 ( $i == j \parallel \dots$ ) 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 iteration of  $i$ ,  $j$ , and  $k$  and therefore  $P4$  never ends up being true.

The screenshot displays the Eclipse IDE interface. The top toolbar includes standard development tools. The Project Explorer on the left shows the project structure with files like Solution.java, solTest.java, Lab1Test.java, and glTest1.java. The JUnit test runner shows a successful run of solTest (0.099 s) with 1/1 runs, 0 errors, and 1 failure. The Failure Trace pane lists two failures related to the test suite. The main editor shows the code for Solution.java, which implements the findEvenNumbers method. The bottom pane shows the Coverage view, which includes a table of coverage data.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Lab_3	92.5 %	184	15	199
sol	92.5 %	184	15	199
solTest.java	88.2 %	105	14	119
solTest	88.2 %	105	14	119
Solution.java	98.8 %	79	1	80
Solution	98.8 %	79	1	80
findEvenNumbers(int[])	98.7 %	76	1	77

### 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.