# SOFTWARE TESTING

## LOGIC BASED TESTING

Criteria : Restricted Active Clause Coverage

November 21, 2017

**Team members**

| | |
|---|---|
| Chavala Sriharipriya | (MT2016040) |
| Nelluri Naveen | (MT2016092) |

## 1 Introduction

Testing is required to find the effective performance of a software. It is very important to ensure that the application should not result in any failures because it can be very expensive in the future.

There are many ways to and procedures to test a software. We have chosen logic based testing to test our program.

## 2 Source code Description

The source code is basically performing all functions of a binary search tree.

### 2.a Binary search tree:

A binary search tree is a tree where each node can contain at most 2 children and for each node all left children should be less than the parent and all right children should be greater than the parent.

### 2.b Functions:

The descriptions of the functions in the code is as below:

- **String add(int d)**
  It adds a node to the tree. The method finds out the correct position of the node to add and inserts it in the tree.
- **void display()**
  It displays the nodes in tree in different orders
- **void preorder(Node n)**
  Displaying the nodes in the tree in such a way that root then left and then right child using recursion.
- **void preorderIterative()**
  Displays nodes same as above but using iteration instead of recursion.
- **void postorder(Node n)**
  Displaying the nodes in the tree in such a way that left then right child and then root using recursion.
- **void postorderIterative()**
  Displays nodes same as above but using iteration instead of recursion.
- **void inorder(Node n)**
  Displaying the nodes in the tree in such a way that left child then root and then right child using recursion.

- **void inorderIterative()**
  Displays nodes same as above but using iteration instead of recursion.
- **String FindAncestors(int d)**
  Finds all ancestors to a given node.
- **void levelOrder()**
  It visits the nodes level wise and displays it.
- **int height()**
  It returns the height of the tree. Height is the no of edges between root node and the deepest leaf node.
- **int FindHeight(Node n)**
  It finds the height of the tree.
- **void levelOrderrec()**
  same as levelOrder but using recursion.
- **void widthOfTree()**
  It finds the width of the tree.
- **int FindDiameter(Node n)**
  It finds the diameter of a tree. Diameter is the largest possible distance between two leaf nodes.
- **void size()**
  It finds the number of nodes in the tree.
- **void printKthLevelNodes(int k)**
  It displays all the nodes that are at level k.
- **public String connectNodesAtLevel()**
  It connects the nodes in the specified level.
- **boolean isSubtree(Node n,Node s)**
  It checks whether s is subtree of n or not.
- **boolean search(int d)**
  This method searches for a particular element in the tree.
- **String remove(int d)**
  This method removes the specified element from the tree if it is present.
- **void findInordersuccessor(int d)**
  This functions finds the inorder sucessor for the node that is being specified
- **void findInorderpredecessor(int d)**
  This functions finds the inorder predecessor for the node that is being specified.

- **Node maximum()**
  Finds the maximum node value in the tree.
- **Node minimum()**
  Finds the minimum node value in the tree.
- **int KthSmallestElement(int k)**
  It finds the kth smallest element in the tree.

## 3 Logic Based Testing:

One of the main blocks of the program are conditional statements. It is very important to test those statements to verify the quality of software. There are many coverage criteria to test these. Among them we have chosen Restricted Active Clause Coverage Criteria(RACC).

### 3.a RACC

For each p âĹĹ P and each major clause ci âĹĹ Cp , choose minor clauses cj , j!=i such that ci determines p.
TR has two requirements for each ci : ci evaluates to true and ci evaluates to false. The values chosen for the minor clauses cj must be the same when ci is true as when ci is false.

### 3.b RACC for our sourcecode:

There are a total of 10 predicates having atleast 3 clauses. We have taken them into consideration and derived test cases for each of the clauses as major excluding the in feasible test requirements.

**For each predicate there are multiple test cases but we have designed the test program in such a way that we can combine the testcases of all predicates into one test case.**

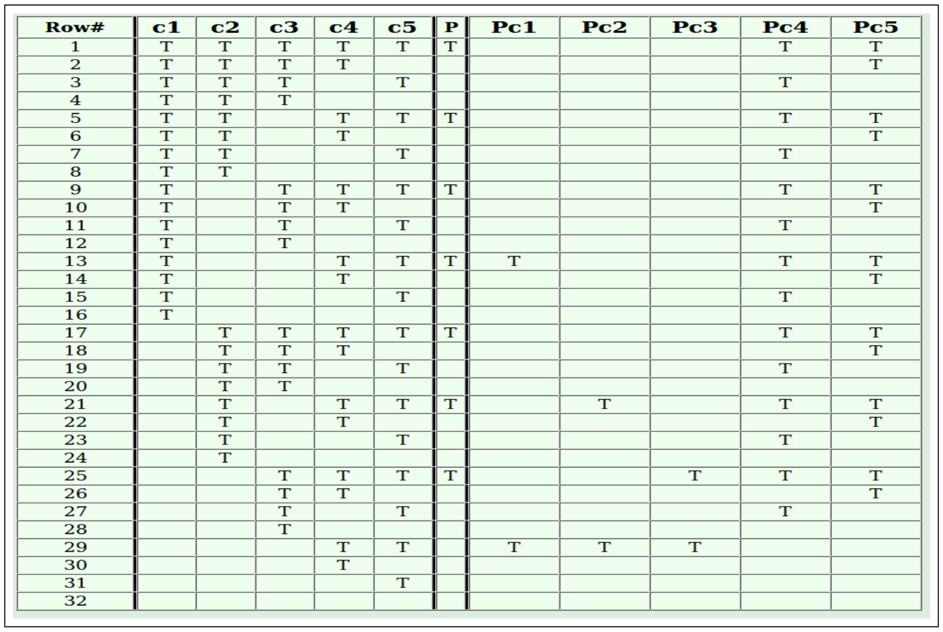
## 4 Predicate1

else if((curr==prev or prev.left==curr or prev.right==curr) and curr.left==null and curr.right==null)

### 4.a Clause Representation

(c1 or c2 or c3) and c4 and c5

### 4.b Truth Table



| Row# | c1 | c2 | c3 | c4 | c5 | P | Pc1 | Pc2 | Pc3 | Pc4 | Pc5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T | T | | | | T | T |
| 2 | T | T | T | T | | | | | | | T |
| 3 | T | T | T | | T | | | | | T | |
| 4 | T | T | T | | | | | | | | |
| 5 | T | T | | T | T | T | | | | T | T |
| 6 | T | T | | T | | | | | | | T |
| 7 | T | T | | | T | | | | | T | |
| 8 | T | T | | | | | | | | | |
| 9 | T | | T | T | T | T | | | | T | T |
| 10 | T | | T | T | | | | | | | T |
| 11 | T | | T | | T | | | | | T | |
| 12 | T | | T | | | | | | | | |
| 13 | T | | | T | T | T | T | | | T | T |
| 14 | T | | | T | | | | | | | T |
| 15 | T | | | | T | | | | | T | |
| 16 | T | | | | | | | | | | |
| 17 | | T | T | T | T | T | | | | T | T |
| 18 | | T | T | T | | | | | | | T |
| 19 | | T | T | | T | | | | | T | |
| 20 | | T | T | | | | | | | | |
| 21 | | T | | T | T | T | | T | | T | T |
| 22 | | T | | T | | | | | | | T |
| 23 | | T | | | T | | | | | T | |
| 24 | | T | | | | | | | | | |
| 25 | | | T | T | T | T | | | T | T | T |
| 26 | | | T | T | | | | | | | T |
| 27 | | | T | | T | | | | | T | |
| 28 | | | T | | | | | | | | |
| 29 | | | | T | T | | T | T | T | | |
| 30 | | | | T | | | | | | | |
| 31 | | | | | T | | | | | | |
| 32 | | | | | | | | | | | |

### 4.c Test Requirements

| Major Clause | Satisfying rows |
|---|---|
| c1 | Not feasible |
| c2 | Not feasible |
| c3 | Not feasible |
| c4 | (13,15) |
| c5 | (21,22) |

## 4.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 0
3 1
3

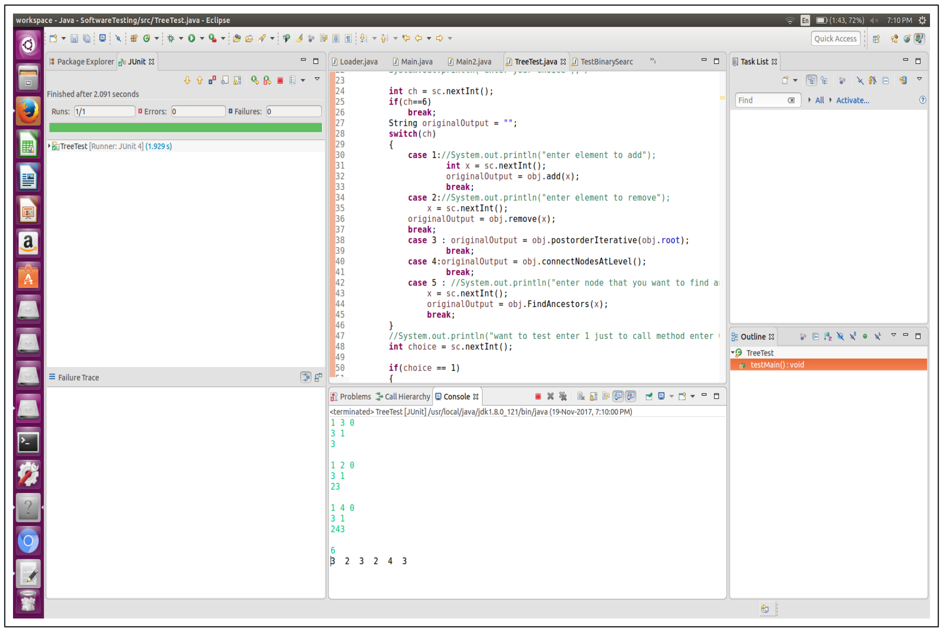
1 2 0
3 1
23

1 4 0
3 1
243

6

## 4.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
            case 4:originalOutput = obj.connectNodesAtLevel();
                    break;
            case 5 : //System.out.println("enter node that you want to find a
                x = sc.nextInt();
                originalOutput = obj.FindAncestors(x);
                break;
        }
        //System.out.println("want to test enter 1 just to call method enter
        int choice = sc.nextInt();

        if(choice == 1)
        {
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:10:00 PM)
1 3 0
3 1
3

1 2 0
3 1
23

1 4 0
3 1
243

6
3  2  3  2  4  3
```

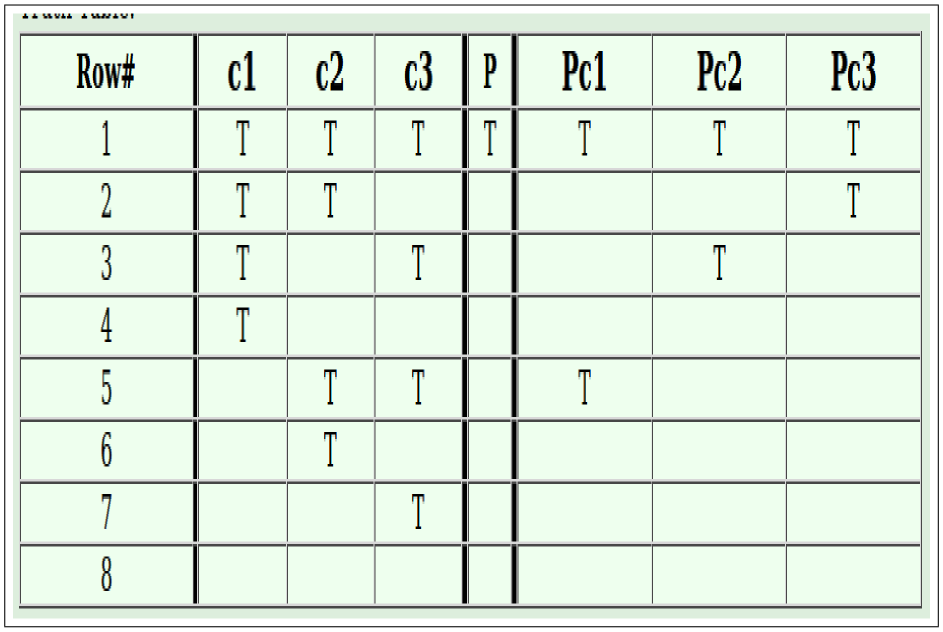
## 5 Predicate2

if(root!=null&d¡current.data&current.left==null)

### 5.a Clause representation

c1&c2&c3

### 5.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T | T | T |
| 2 | T | T |  |  |  |  | T |
| 3 | T |  | T |  |  | T |  |
| 4 | T |  |  |  |  |  |  |
| 5 |  | T | T |  | T |  |  |
| 6 |  | T |  |  |  |  |  |
| 7 |  |  | T |  |  |  |  |
| 8 |  |  |  |  |  |  |  |

### 5.c Test Requirements

| Major Clause | Satisfying rows |
|---|---|
| c1 | (1,5) |
| c2 | (1,3) |
| c3 | (1,2) |

## 5.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 0
1 2 1
23

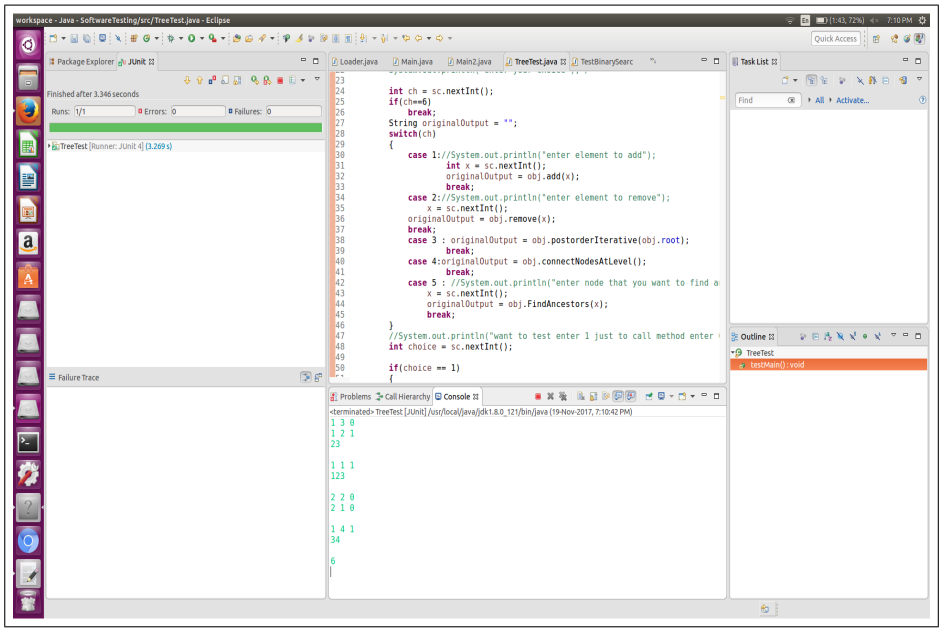
1 1 1
123

2 2 0
2 1 0
1 4 1
34

6

## 5.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
            case 4:originalOutput = obj.connectNodesAtLevel();
                    break;
            case 5 : //System.out.println("enter node that you want to find a
                x = sc.nextInt();
                originalOutput = obj.FindAncestors(x);
                break;
        }
        //System.out.println("want to test enter 1 just to call method enter
        int choice = sc.nextInt();

        if(choice == 1)
        {
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:10:42 PM)
1 3 0
1 2 1
23

1 1 1
123

2 2 0
2 1 0

1 4 1
34

6
```

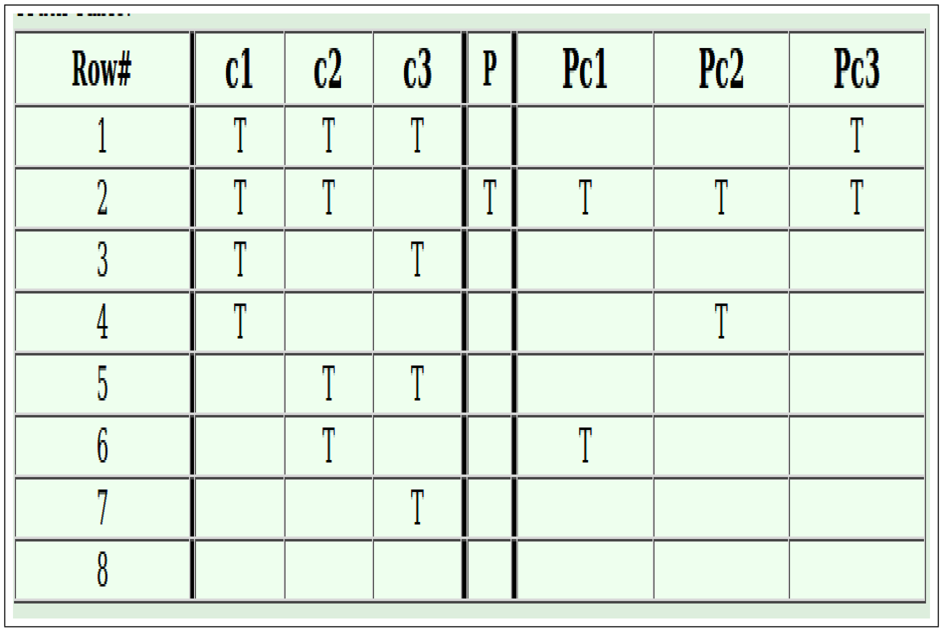
## 6 Predicate3

else if(root!=null&d¡current.data&!(current.left==null))

### 6.a Clause representation

c1&c2!c3

### 6.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | | | | T |
| 2 | T | T | | T | T | T | T |
| 3 | T | | T | | | | |
| 4 | T | | | | | T | |
| 5 | | T | T | | | | |
| 6 | | T | | | T | | |
| 7 | | | T | | | | |
| 8 | | | | | | | |

### 6.c Test Requirements

| Major Clause | Satisfying rows |
|---|---|
| c1 | Not feasible |
| c2 | (2,4) |
| c3 | (1,2) |

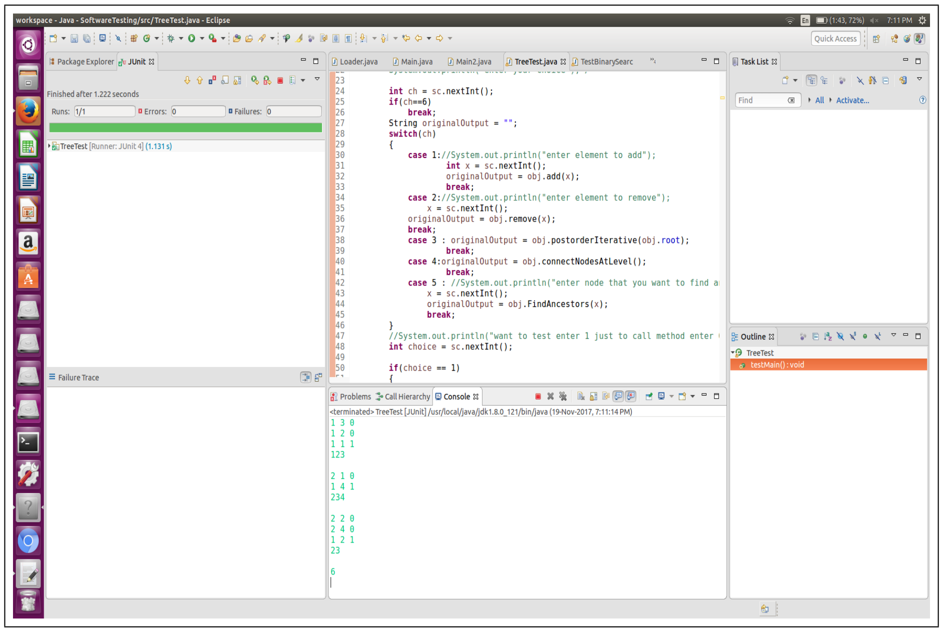
## 6.d Testcases

Testcases for the corresponding RACC criteria are:

```
1 3 0
1 2 0
1 1 1
123

2 1 0
1 4 1
234

2 2 0
2 4 0
1 2 1
23

6
```

## 6.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



workspace - Java - SoftwareTesting/src/TreeTest.java - Eclipse

Finished after 1.222 seconds

Runs: 1/1 Errors: 0 Failures: 0

TreeTest [Runner: JUnit 4] (1.131 s)

```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
            case 4:originalOutput = obj.connectNodesAtLevel();
                    break;
            case 5 : //System.out.println("enter node that you want to find a
                x = sc.nextInt();
                originalOutput = obj.FindAncestors(x);
                break;
        }
        //System.out.println("want to test enter 1 just to call method enter 
        int choice = sc.nextInt();

        if(choice == 1)
        {
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:11:14 PM)
1 3 0
1 2 0
1 1 1
123

2 1 0
1 4 1
234

2 2 0
2 4 0
1 2 1
23

6
```

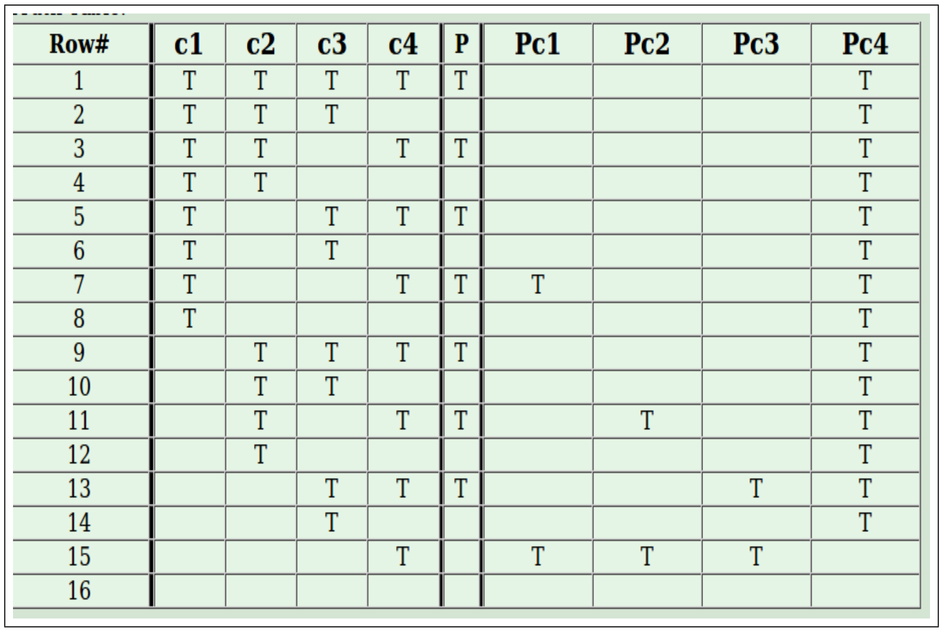
## 7 Predicate4

if((curr==prev $\|prev.left == curr\|prev.right == curr)\&\&curr.left! = null)$

### 7.a Clause representation

(c1$\|c2\|c3)\&c4$

### 7.b Truth Table



| Row# | c1 | c2 | c3 | c4 | P | Pc1 | Pc2 | Pc3 | Pc4 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T | | | | T |
| 2 | T | T | T | | | | | | T |
| 3 | T | T | | T | T | | | | T |
| 4 | T | T | | | | | | | T |
| 5 | T | | T | T | T | | | | T |
| 6 | T | | T | | | | | | T |
| 7 | T | | | T | T | T | | | T |
| 8 | T | | | | | | | | T |
| 9 | | T | T | T | T | | | | T |
| 10 | | T | T | | | | | | T |
| 11 | | T | | T | T | | T | | T |
| 12 | | T | | | | | | | T |
| 13 | | | T | T | T | | | T | T |
| 14 | | | T | | | | | | T |
| 15 | | | | T | | T | T | T | |
| 16 | | | | | | | | | |

### 7.c Test Requirements

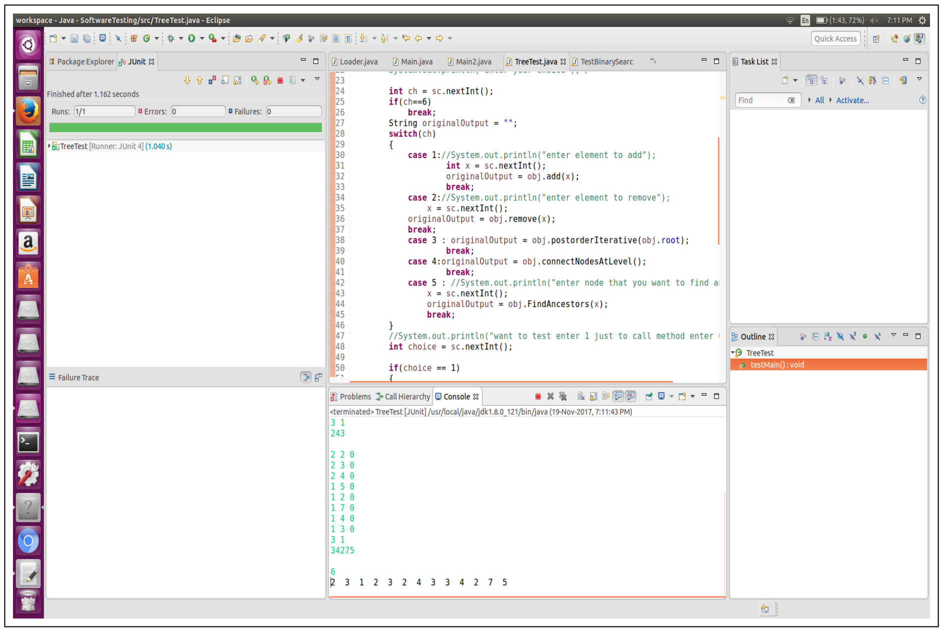
| Major Clause | Satisfying rows |
|---|---|
| c1 | (7,15) |
| c2 | (11,15) |
| c3 | (13,15) |
| c4 | (7,8) |

## 7.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 1
3
1 2 0 3 1
23
1 1 0 3 1
123
2 1 0 1 4 0
3 1
243
2 2 0 2 3 0 2 4 0 1 5 0
1 2 0 1 7 0 1 4 0 1 3 0
3 1
34275
6

## 7.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



Finished after 1.162 seconds
Runs: 1/1 Errors: 0 Failures: 0
TreeTest [Runner: JUnit 4] (1.040 s)

```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
            case 4:originalOutput = obj.connectNodesAtLevel();
                    break;
            case 5 : //System.out.println("enter node that you want to find a
                x = sc.nextInt();
                originalOutput = obj.FindAncestors(x);
                break;
        }
        //System.out.println("want to test enter 1 just to call method enter
        int choice = sc.nextInt();

        if(choice == 1)
        {
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:11:43 PM)
3 1
243

2 2 0
2 3 0
2 4 0
1 5 0
1 2 0
1 7 0
1 4 0
1 3 0
3 1
34275

6
2  3  1  2  3  2  4  3  3  4  2  7  5
```

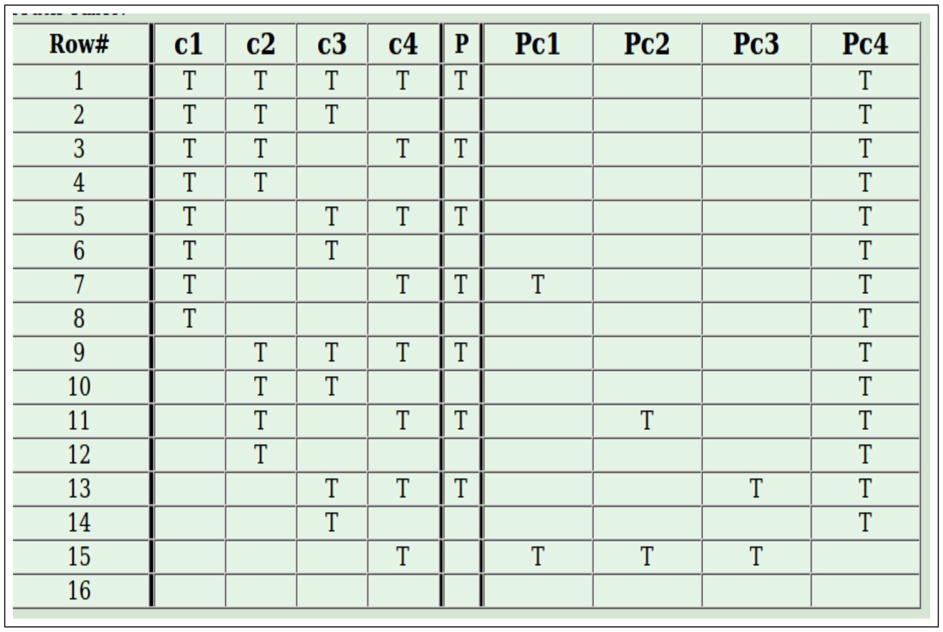
## 8 Predicate5

else if((curr==prev or prev.left==curr or prev.right==curr) and curr.right!=null)

### 8.a Clause representation

(c1 or c2 or c3) and c4

### 8.b Truth Table



| Row# | c1 | c2 | c3 | c4 | P | Pc1 | Pc2 | Pc3 | Pc4 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T |  |  |  | T |
| 2 | T | T | T |  |  |  |  |  | T |
| 3 | T | T |  | T | T |  |  |  | T |
| 4 | T | T |  |  |  |  |  |  | T |
| 5 | T |  | T | T | T |  |  |  | T |
| 6 | T |  | T |  |  |  |  |  | T |
| 7 | T |  |  | T | T | T |  |  | T |
| 8 | T |  |  |  |  |  |  |  | T |
| 9 |  | T | T | T | T |  |  |  | T |
| 10 |  | T | T |  |  |  |  |  | T |
| 11 |  | T |  | T | T |  | T |  | T |
| 12 |  | T |  |  |  |  |  |  | T |
| 13 |  |  | T | T | T |  |  | T | T |
| 14 |  |  | T |  |  |  |  |  | T |
| 15 |  |  |  | T |  | T | T | T |  |
| 16 |  |  |  |  |  |  |  |  |  |

### 8.c Test Requirements

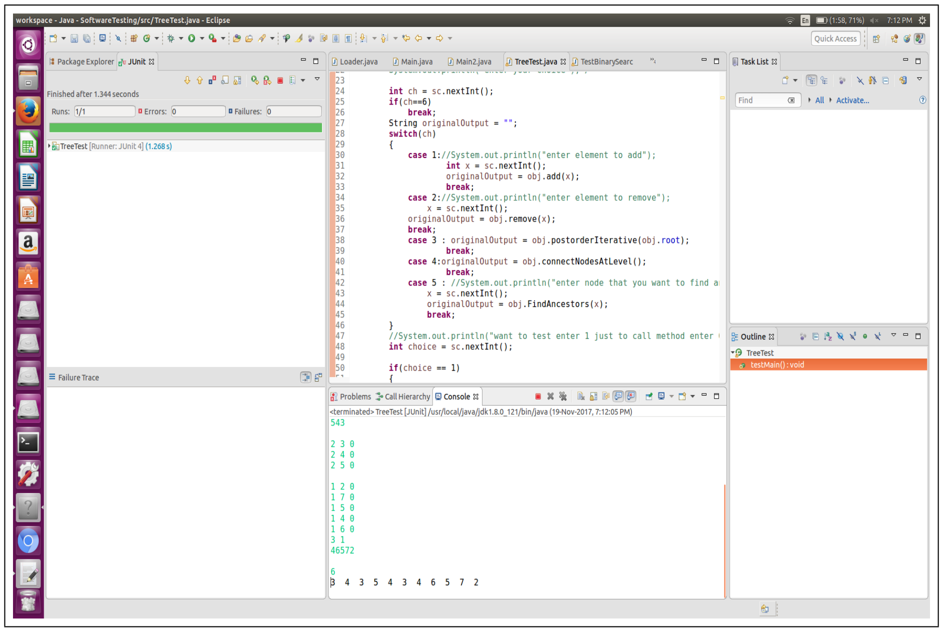
| Major Clause | Satisfying rows |
|---|---|
| c1 | (7,15) |
| c2 | (11,15) |
| c3 | (13,15) |
| c4 | (7,8) |

## 8.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 0 3 1
3
1 4 0 3 1
43
1 5 0
3 1
543
2 3 0 2 4 0
2 5 0 1 2 0
1 7 0 1 5 0
1 4 0 1 6 0
3 1
46572
6

## 8.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



Finished after 1.344 seconds

Runs: 1/1 Errors: 0 Failures: 0

TreeTest [Runner: JUnit 4] (1.268 s)

```
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
            case 4:originalOutput = obj.connectNodesAtLevel();
                    break;
            case 5 : //System.out.println("enter node that you want to find a
                x = sc.nextInt();
                originalOutput = obj.FindAncestors(x);
                break;
        }
        //System.out.println("want to test enter 1 just to call method enter
        int choice = sc.nextInt();

        if(choice == 1)
        {
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:12:05 PM)
543

2 3 0
2 4 0
2 5 0

1 2 0
1 7 0
1 5 0
1 4 0
1 6 0
3 1
46572

6
3  4  3  5  4  3  4  6  5  7  2
```

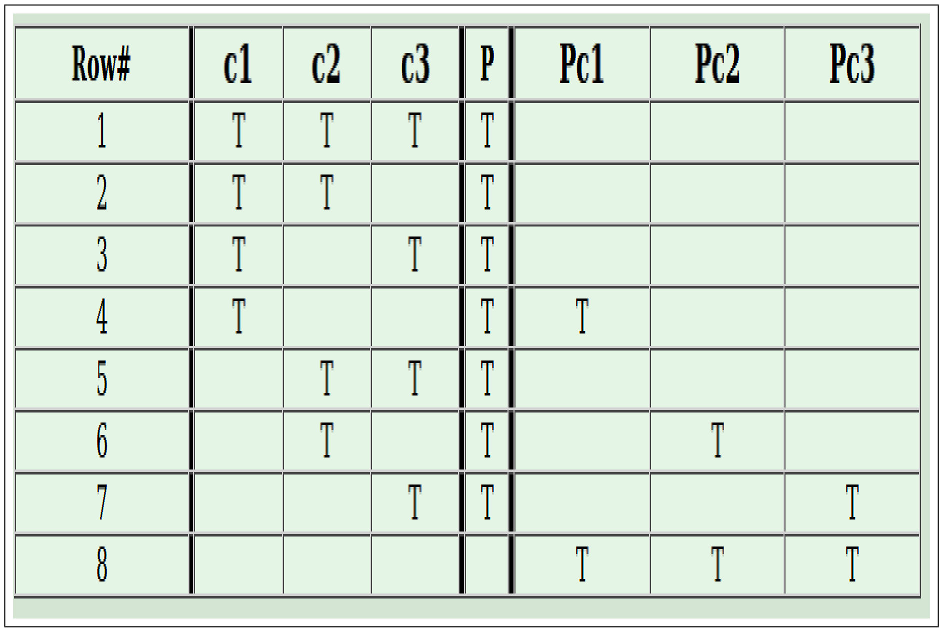
## 9 Predicate6

if(curr==prev or prev.left==curr or prev.right==curr)

### 9.a Clause representation

c1 or c2 or c3

### 9.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | | | |
| 2 | T | T | | T | | | |
| 3 | T | | T | T | | | |
| 4 | T | | | T | T | | |
| 5 | | T | T | T | | | |
| 6 | | T | | T | | T | |
| 7 | | | T | T | | | T |
| 8 | | | | | T | T | T |

### 9.c Test Requirements

| Major Clause | Satisfying rows |
|---|---|
| c1 | (4,8) |
| c2 | (6,8) |
| c3 | (7,8) |

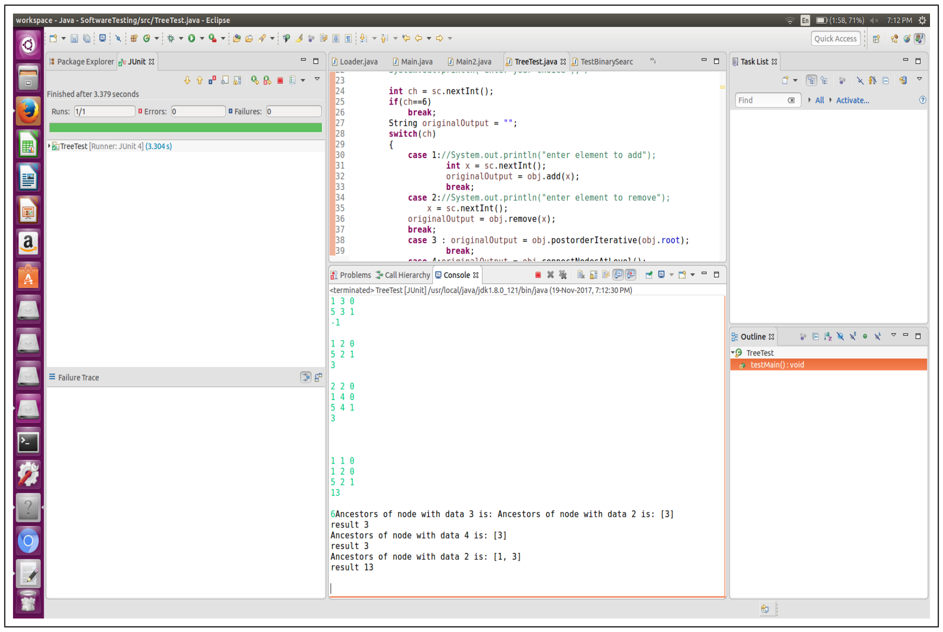
## 9.d Testcases

Testcases for the corresponding RACC criteria are:

1 3 0  
5 3 1  
-1  
1 2 0  
5 2 1  
3  
2 2 0  
1 4 0  
5 4 1  
3  
1 1 0  
1 2 0  
5 2 1  
13  
6

## 9.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



```
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:12:30 PM)
1 3 0
5 3 1
-1

1 2 0
5 2 1
3

2 2 0
1 4 0
5 4 1
3



1 1 0
1 2 0
5 2 1
13

6Ancestors of node with data 3 is: Ancestors of node with data 2 is: [3]
result 3
Ancestors of node with data 4 is: [3]
result 3
Ancestors of node with data 2 is: [1, 3]
result 13
```

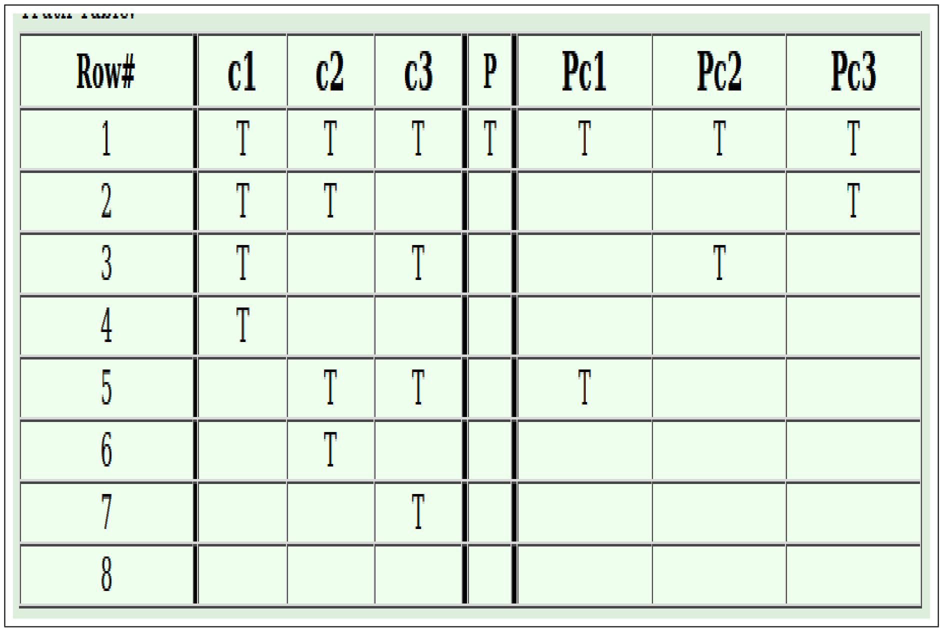
## 10 Predicate7

if((curr.left==prev)&(curr.right==null)&temp.data==d)

### 10.a Clause representation

c1&c2&c3

### 10.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T | T | T |
| 2 | T | T |  |  |  |  | T |
| 3 | T |  | T |  |  | T |  |
| 4 | T |  |  |  |  |  |  |
| 5 |  | T | T |  | T |  |  |
| 6 |  | T |  |  |  |  |  |
| 7 |  |  | T |  |  |  |  |
| 8 |  |  |  |  |  |  |  |

### 10.c Test Requirements

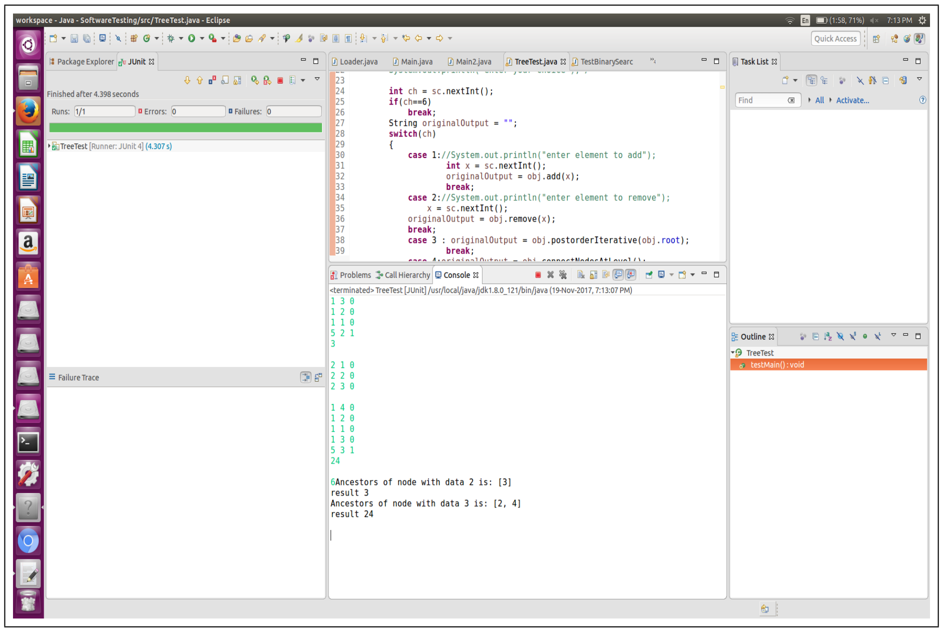
| Major Clause | Satisfying rows |
|---|---|
| c1 | (1,5) |
| c2 | (1,3) |
| c3 | (1,2) |

## 10.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 0
1 2 0
1 1 0
5 2 1
3
2 1 0
2 2 0
2 3 0
1 4 0
1 2 0
1 1 0
1 3 0
5 3 1
24
6

## 10.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



Finished after 4.398 seconds

Runs: 1/1 Errors: 0 Failures: 0

TreeTest [Runner: JUnit 4] (4.307 s)

TreeTest
testMain() : void

```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:13:07 PM)
1 3 0
1 2 0
1 1 0
5 2 1
3

2 1 0
2 2 0
2 3 0

1 4 0
1 2 0
1 1 0
1 3 0
5 3 1
24

6Ancestors of node with data 2 is: [3]
result 3
Ancestors of node with data 3 is: [2, 4]
result 24
```

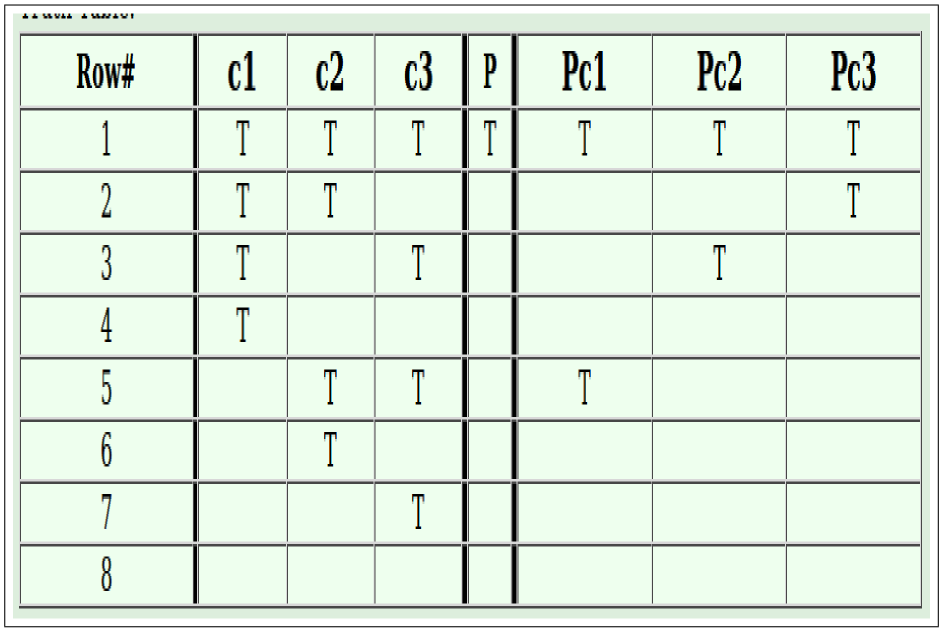
## 11 Predicate8

if(n!=null&i¿=1&n.left!=null)

### 11.a Clause representation

c1&c2&c3

### 11.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T | T | T |
| 2 | T | T |  |  |  |  | T |
| 3 | T |  | T |  |  | T |  |
| 4 | T |  |  |  |  |  |  |
| 5 |  | T | T |  | T |  |  |
| 6 |  | T |  |  |  |  |  |
| 7 |  |  | T |  |  |  |  |
| 8 |  |  |  |  |  |  |  |

### 11.c Test Requirements

| Major Clause | Satisfying rows |
|---|---|
| c1 | (1,5) |
| c2 | (1,3) |
| c3 | (1,2) |

## 11.d Testcases

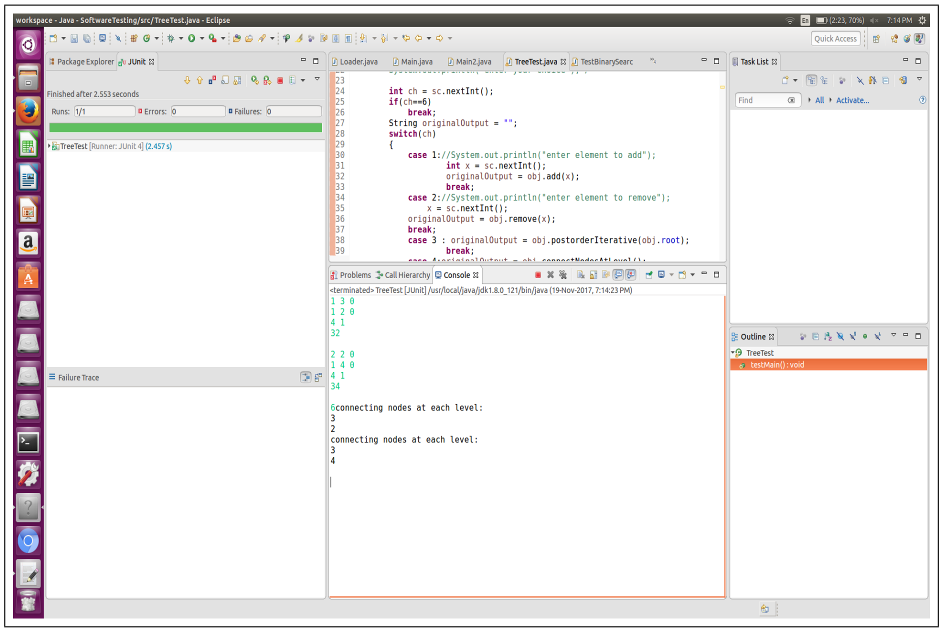
Testcases for the corresponding RACC criteria are:
1 3 0
1 2 0
4 1
32

2 2 0
1 4 0
4 1
34

6

## 11.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:14:23 PM)
1 3 0
1 2 0
4 1
32

2 2 0
1 4 0
4 1
34

6connecting nodes at each level:
3
2
connecting nodes at each level:
3
4
```

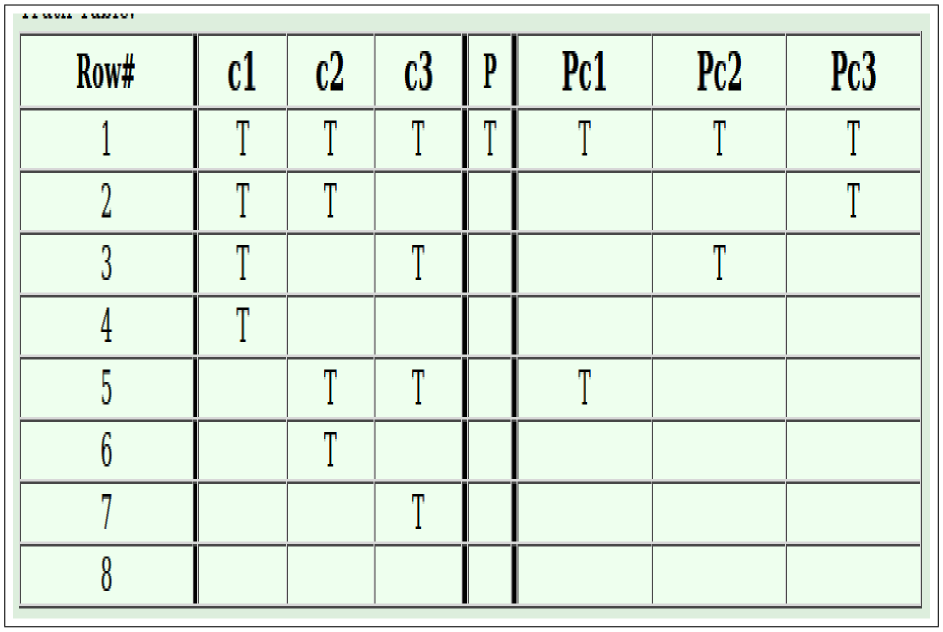
## 12 Predicate9

if(current!=null&current.left==null& current.right==null)

### 12.a Clause representation

c1&c2&c3

### 12.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T | T | T |
| 2 | T | T | | | | | T |
| 3 | T | | T | | | T | |
| 4 | T | | | | | | |
| 5 | | T | T | | T | | |
| 6 | | T | | | | | |
| 7 | | | T | | | | |
| 8 | | | | | | | |

### 12.c Test Requirements

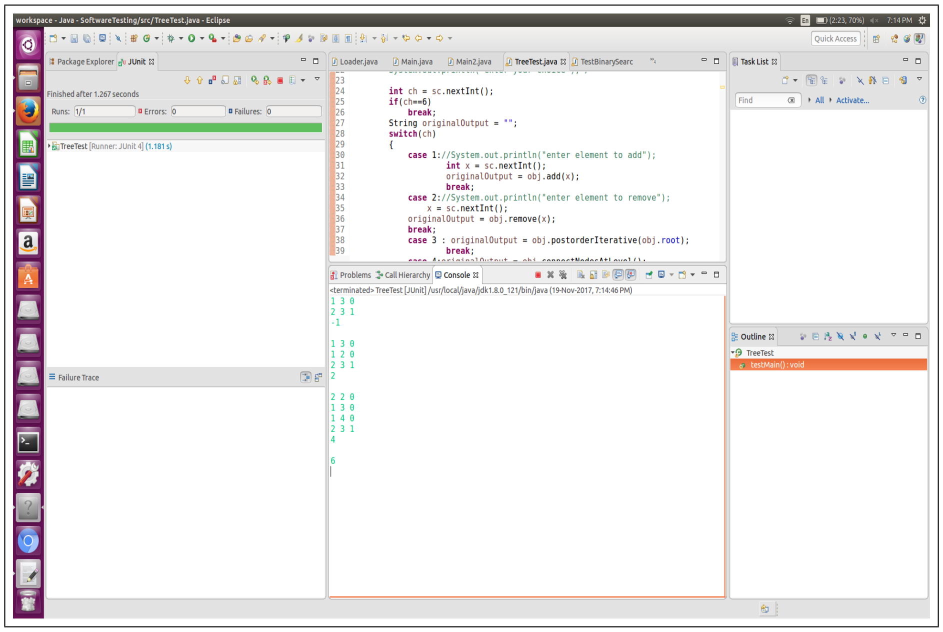
| Major Clause | Satisfying rows |
|---|---|
| c1 | (1,5) |
| c2 | (1,3) |
| c3 | (1,2) |

## 12.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 0
2 3 1
-1
1 3 0
1 2 0
2 3 1
2
2 2 0
1 3 0
1 4 0
2 3 1
4
6

## 12.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



workspace - Java - SoftwareTesting/src/TreeTest.java - Eclipse

Finished after 1.267 seconds

Runs: 1/1 Errors: 0 Failures: 0

TreeTest [Runner: JUnit 4] (1.181 s)

```
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:14:46 PM)
1 3 0
2 3 1
-1

1 3 0
1 2 0
2 3 1
2

2 2 0
1 3 0
1 4 0
2 3 1
4

6
```

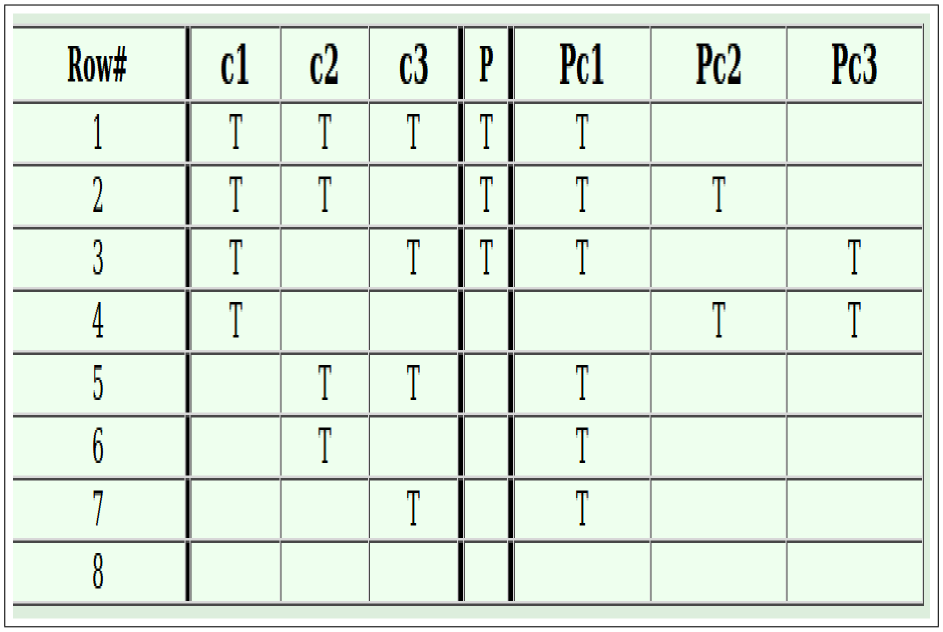
## 13 Predicate10

else if(current!=null&(current.left==null‖$current.right == null$))

### 13.a Clause representation

c1&(c2‖$c3$)

### 13.b Truth Table



| Row# | c1 | c2 | c3 | P | Pc1 | Pc2 | Pc3 |
|---|---|---|---|---|---|---|---|
| 1 | T | T | T | T | T |  |  |
| 2 | T | T |  | T | T | T |  |
| 3 | T |  | T | T | T |  | T |
| 4 | T |  |  |  |  | T | T |
| 5 |  | T | T |  | T |  |  |
| 6 |  | T |  |  | T |  |  |
| 7 |  |  | T |  | T |  |  |
| 8 |  |  |  |  |  |  |  |

### 13.c Test Requirements

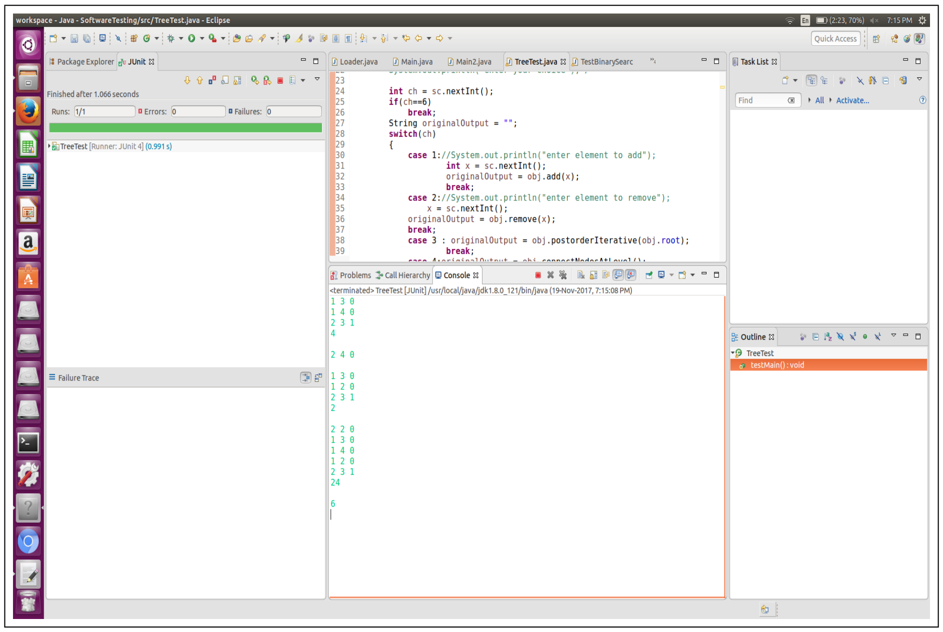
| Major Clause | Satisfying rows |
|---|---|
| c1 | (2,6) |
| c2 | (2,4) |
| c3 | (3,4) |

## 13.d Testcases

Testcases for the corresponding RACC criteria are:
1 3 0 1 4 0
2 3 1
4
2 4 0 1 3 0
1 2 0 2 3 1
2
2 2 0 1 3 0
1 4 0 1 2 0
2 3 1
24
6

## 13.e Screenshot

We have run the same test case and tested using JUNIT. Below the screenshot for the same.



workspace - Java - SoftwareTesting/src/TreeTest.java - Eclipse

Finished after 1.066 seconds

Runs: 1/1 Errors: 0 Failures: 0

TreeTest [Runner: JUnit 4] (0.991 s)

```java
        int ch = sc.nextInt();
        if(ch==6)
            break;
        String originalOutput = "";
        switch(ch)
        {
            case 1://System.out.println("enter element to add");
                    int x = sc.nextInt();
                    originalOutput = obj.add(x);
                    break;
            case 2://System.out.println("enter element to remove");
                x = sc.nextInt();
            originalOutput = obj.remove(x);
            break;
            case 3 : originalOutput = obj.postorderIterative(obj.root);
                    break;
```

```
<terminated> TreeTest [JUnit] /usr/local/java/jdk1.8.0_121/bin/java (19-Nov-2017, 7:15:08 PM)
1 3 0
1 4 0
2 3 1
4

2 4 0

1 3 0
1 2 0
2 3 1
2

2 2 0
1 3 0
1 4 0
1 2 0
2 3 1
24

6
```

As shown in screenshots we have run all testcases on the program and found no faults.