The following program fragment adds an object to the right branch of a tree structure.

```
  if((o != null) && !tree.contains(o) && (tree.isBalanced() || (tree.right().size() <
tree.left().size()))) {

      tree.addRight(o);

   }
```

Derive a set of test cases that satisfy the MC/DC adequacy criterion. The solution should indicate the elementary conditions, specify the combinations of truth values that satisfy the MC/DC criterion, and provide a set of corresponding concrete test cases.

The elementary conditions in the program fragment are:

1- o != null - checks if the object to be added is not null.
2- !tree.contains(o) - checks if the tree does not already contain the object.
3- tree.isBalanced() - checks if the tree is balanced.
4- tree.right().size() < tree.left().size() - checks if the size of the right branch of the tree is less than the size of the left branch.

To satisfy the MC/DC adequacy criterion, we need to have test cases that cover all possible combinations of truth values for the elementary conditions. We can use the truth table method to identify the combinations of truth values that satisfy the MC/DC criterion.

o != null	!tree.contains(o)	tree.isBalanced()	tree.right().size() < tree.left().size()	Add OBJ to right branch
T	T	T	T	YES
T	T	T	F	YES
T	T	F	-	YES
T	F	-	-	NO
F	-	-	-	NO

Based on the truth table, we can derive the following set of test cases that satisfy the MC/DC adequacy criterion:

Test Case 1: o != null, !tree.contains(o), tree.isBalanced(), and tree.right().size() < tree.left().size()

Test Case 2: o != null, !tree.contains(o), tree.isBalanced(), and !(tree.right().size() < tree.left().size())

Test Case 3: o != null, !tree.contains(o), and !tree.isBalanced()

Test Case 4: o != null and tree.contains(o)

Test Case 5: o = null