Logic Based Coverage:

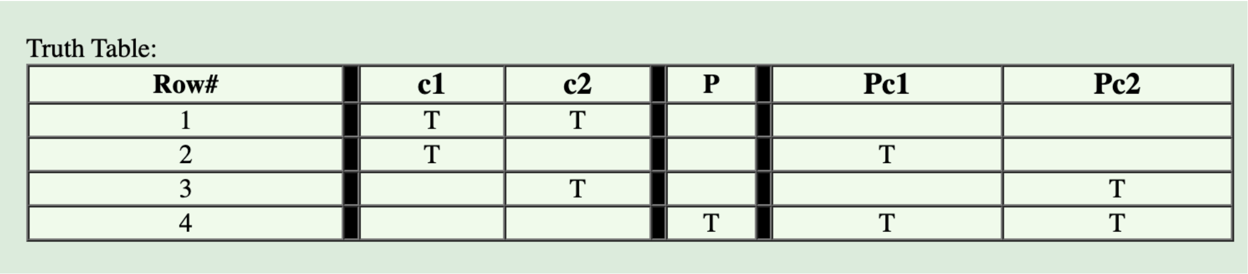
Class 1: CopyOnWriteHashList.java

Method1: Test Logic for : public boolean add(T item)

Testing the decision for code:
if (item == null || (index >= 0 && indexOf(item) == index))

Simplify it to the logic condition:
C1 = item == null
C2 = item is already at the end of the list
Decision: D = !(C1 OR C2)

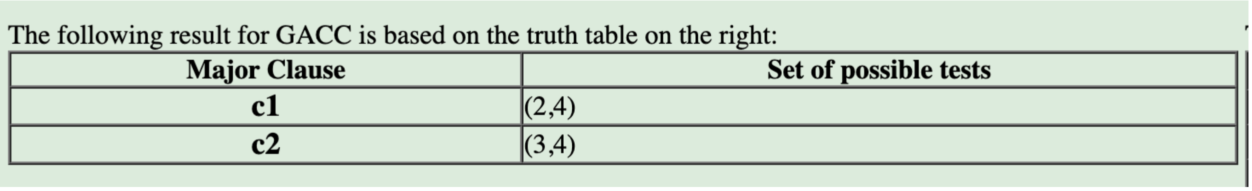
Truth Table:



Truth Table:

Row#	c1	c2	P	Pc1	Pc2
1	T	T			
2	T			T	
3		T			T
4			T	T	T

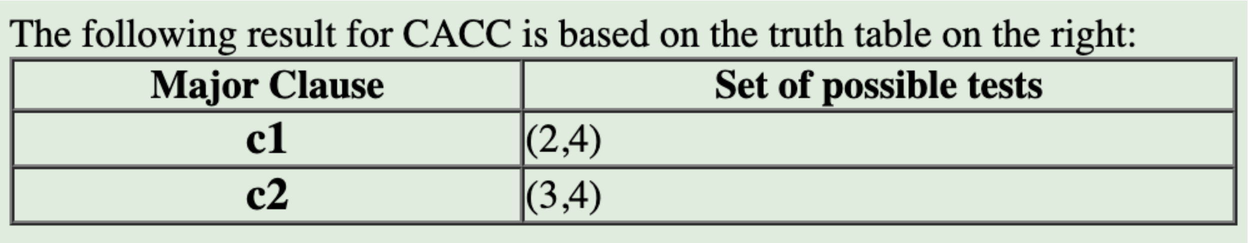
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
c1	(2,4)
c2	(3,4)

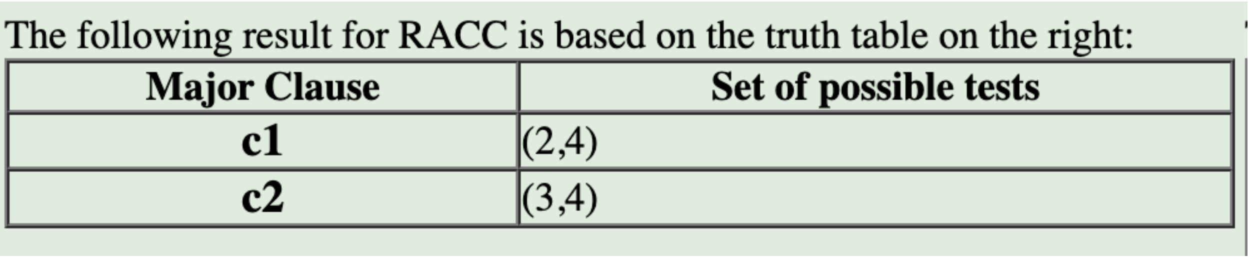
CACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
c1	(2,4)
c2	(3,4)

RACC:



The following result for RACC is based on the truth table on the right:

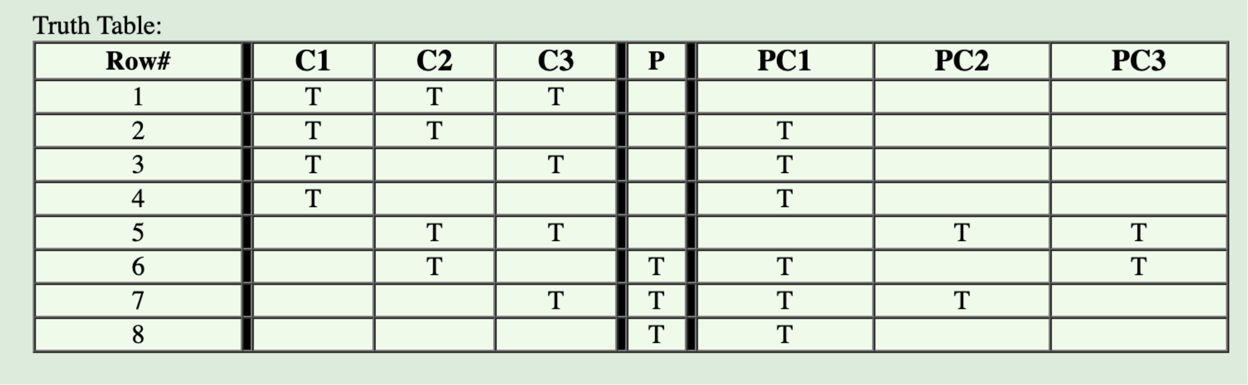
Major Clause	Set of possible tests
c1	(2,4)
c2	(3,4)

Method 2: public void add(int index, T item)

Testing the Decision for Code:
if (item == null || (index >= 0 && indexOf(item) == index))

Simplify it to the Logic Condition:
C1 = item == null
C2 = index >= 0
C3 = indexOf(item) == index
Decision: D = !(C1 OR (C2 AND C3))

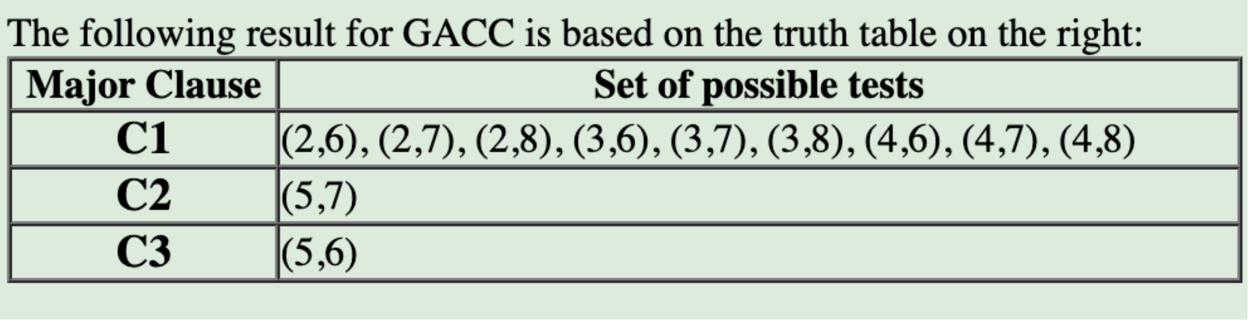
Truth Table:



Truth Table:

Row#	C1	C2	C3	P	PC1	PC2	PC3
1	T	T	T				
2	T	T			T		
3	T		T		T		
4	T				T		
5		T	T			T	T
6		T		T	T		T
7			T	T	T	T	
8				T	T		

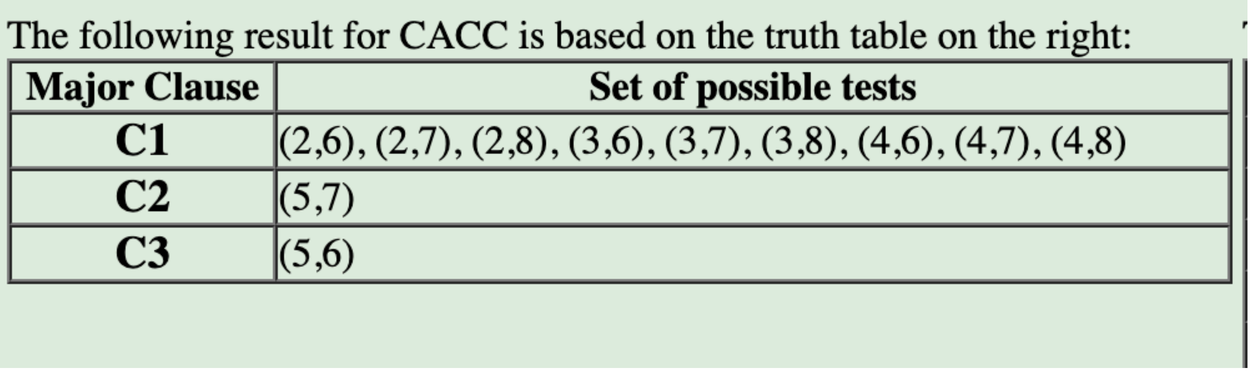
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(2,6), (2,7), (2,8), (3,6), (3,7), (3,8), (4,6), (4,7), (4,8)
C2	(5,7)
C3	(5,6)

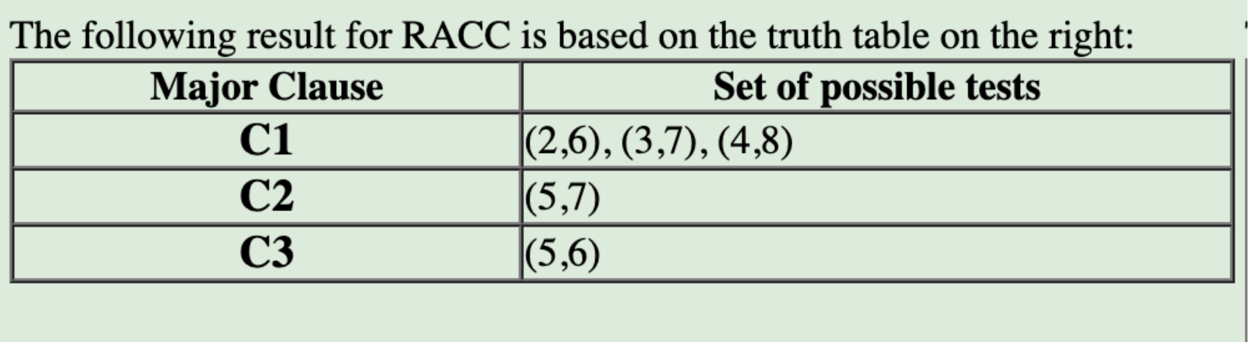
CACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(2,6), (2,7), (2,8), (3,6), (3,7), (3,8), (4,6), (4,7), (4,8)
C2	(5,7)
C3	(5,6)

RACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(2,6), (3,7), (4,8)
C2	(5,7)
C3	(5,6)

Class 2: PasswdURI.java

Method 3: public String getUsername()

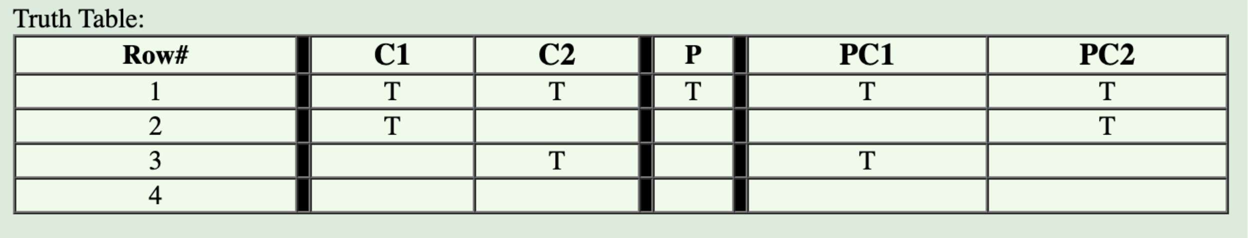
Testing the Decision for Code:

```
if (split.length == 2) {
   String[] split2 = split[0].split(":");

   if (split2.length == 2) {
        result = split2[0]; // Username
    }
}
```

Simplify it to the Logic Condition:
C1 = split.length == 2
C2 = split2.length == 2
Decision: D = C1 AND C2 → only if both are true, return a username

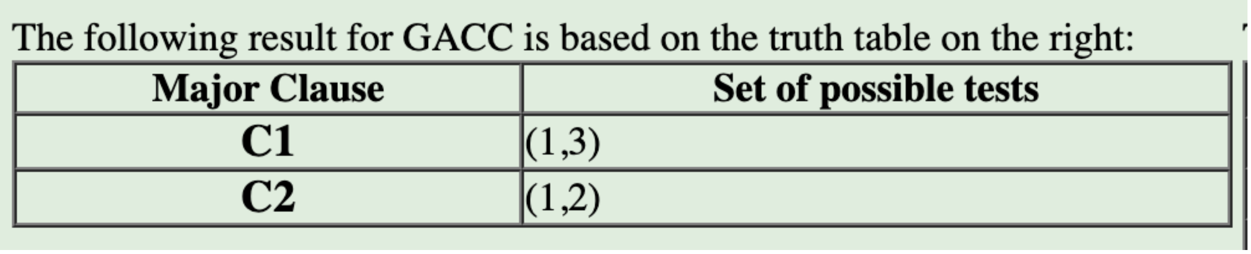
Truth Table:



Truth Table:

Row#	C1	C2	P	PC1	PC2
1	T	T	T	T	T
2	T				T
3		T		T	
4					

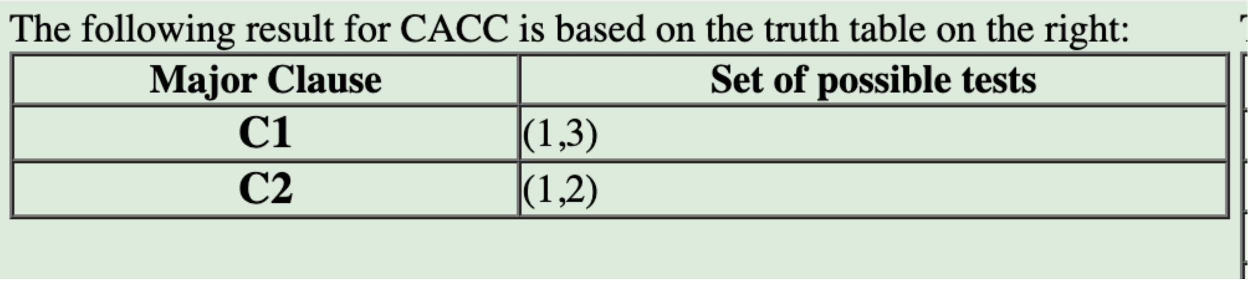
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(1,2)

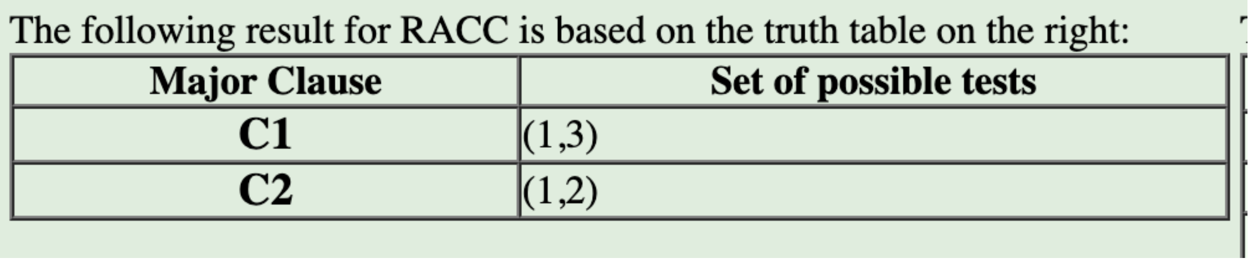
CACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(1,2)

RACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(1,2)

Method 4: public String getPassword()

Testing the Decision for Code:

Same logic as getUsername(), but accessing split2[1] for password.

```
if (split.length == 2) {
    String[] split2 = split[0].split(":");

    if (split2.length == 2) {
        result = split2[1]; // Password
    }
}
```

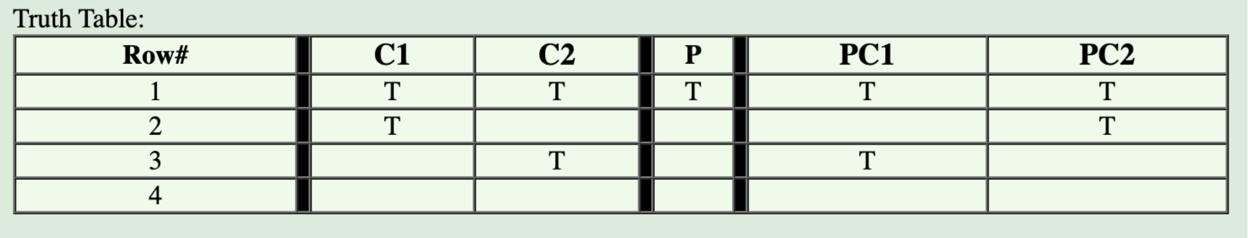
Simplify it to the Logic Condition:

C1 = split.length == 2

C2 = split2.length == 2

Decision: D = C1 AND C2

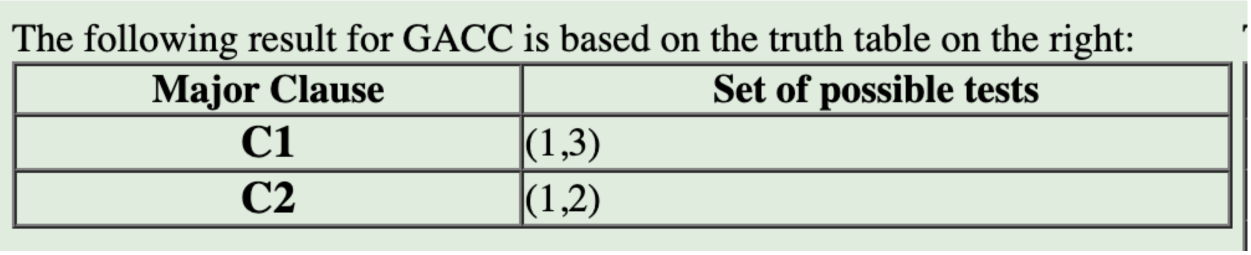
Truth Table:



Truth Table:

Row#	C1	C2	P	PC1	PC2
1	T	T	T	T	T
2	T				T
3		T		T	
4					

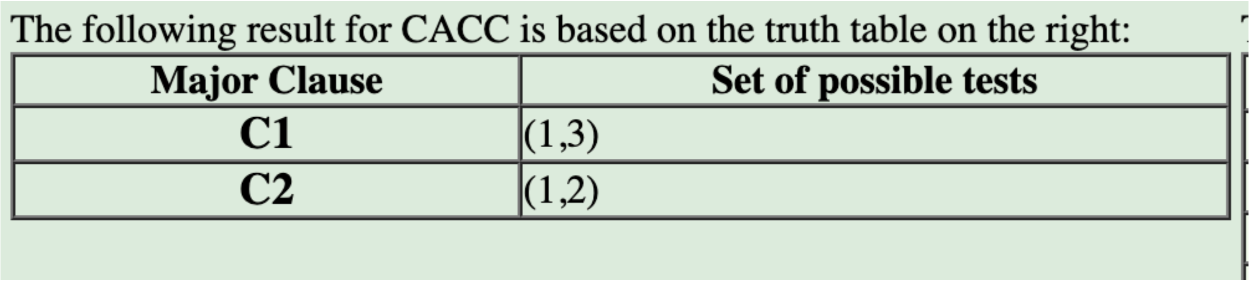
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(1,2)

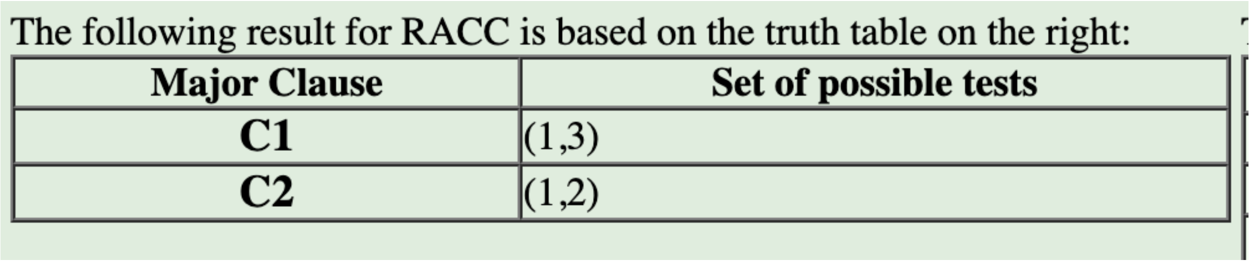
CACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(1,2)

RACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(1,2)

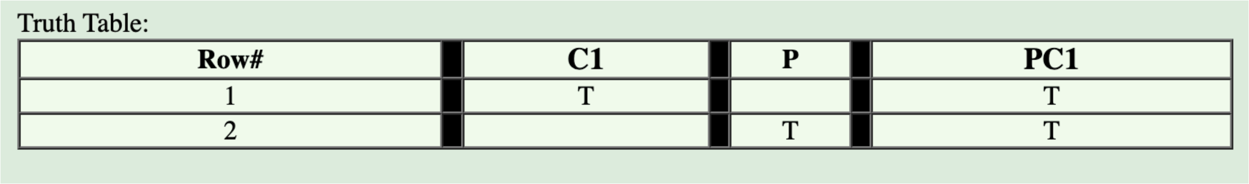
Class 3: DateFormatter.java

Method 5: public static String format(Date date, String format)

Testing the Decision for Code:
if (date == null) return "";
The only logic is checking for null. If the date is not null, the formatting runs normally.

Simplify it to Logic Condition:
C1 = date == null
Decision:
If C1 is true → return ""
If C1 is false → return formatted date string

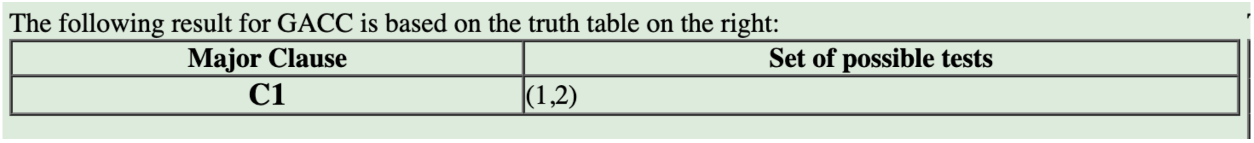
Truth Table:



Truth Table:

Row#	C1	P	PC1
1	T		T
2		T	T

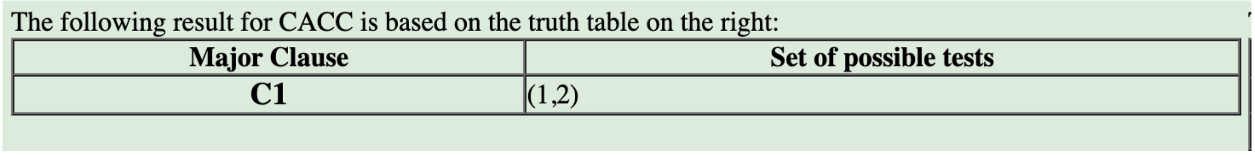
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,2)

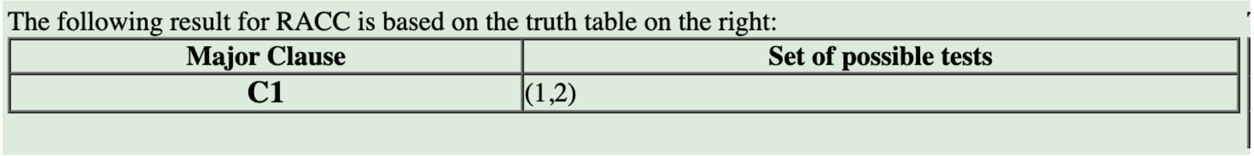
CACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,2)

RACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,2)

Method 6:public static boolean isSameDay(Date date1, Date date2)

This method wraps the overloaded one:
Part1: public static boolean isSameDay(Calendar cal1, Calendar cal2)

Testing the Decision for Code:

```
if (date1 == null || date2 == null) {
   throw new IllegalArgumentException("Dates must not be null");
}
```

Then compare:

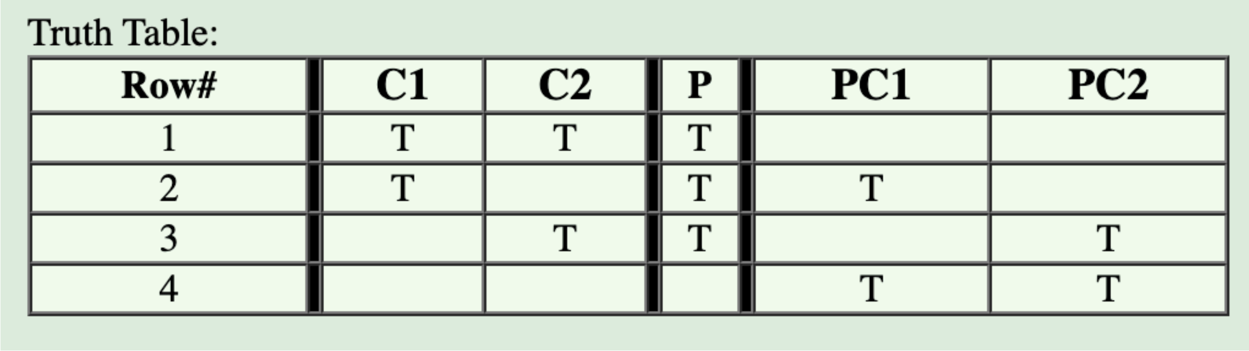
```
Part2: return (cal1.get(ERA) == cal2.get(ERA) &&
       cal1.get(YEAR) == cal2.get(YEAR) &&
       cal1.get(DAY_OF_YEAR) == cal2.get(DAY_OF_YEAR));
```

We will treat this in two layers:

Part 1: Null Check

Simplify to Logic Condition:
C1 = date1 == null
C2 = date2 == null
Decision: C1 OR C2 → if true, throws exception

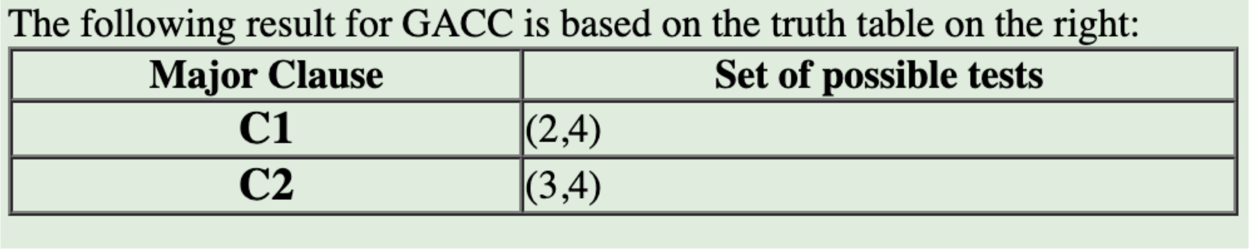
Truth Table:



Truth Table:

Row#	C1	C2	P	PC1	PC2
1	T	T	T		
2	T		T	T	
3		T	T		T
4				T	T

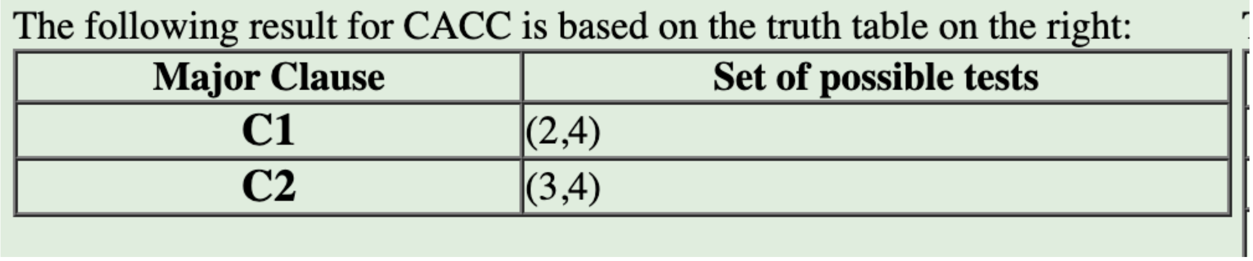
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(2,4)
C2	(3,4)

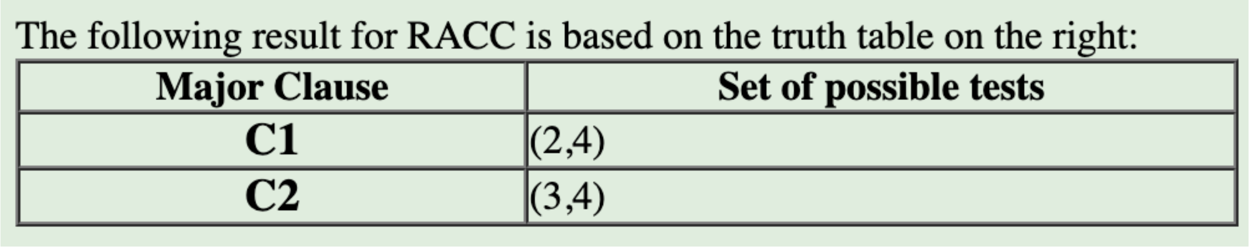
CACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(2,4)
C2	(3,4)

RACC:

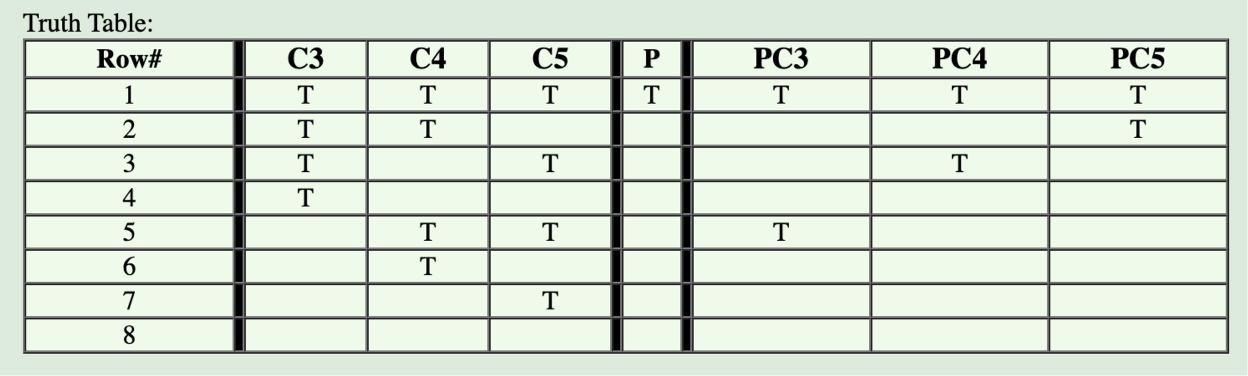


The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(2,4)
C2	(3,4)

Part 2: Same-Day Comparison
Logic Condition:
C3 = same ERA
C4 = same YEAR
C5 = same DAY_OF_YEAR
Decision = C3 & C4 & C5

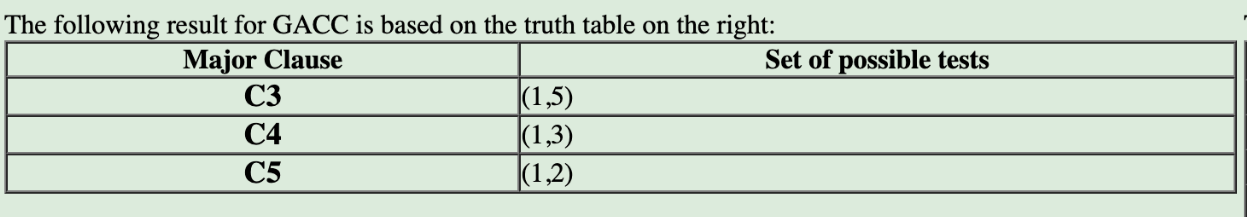
Truth Table:



Truth Table:

Row#	C3	C4	C5	P	PC3	PC4	PC5
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							

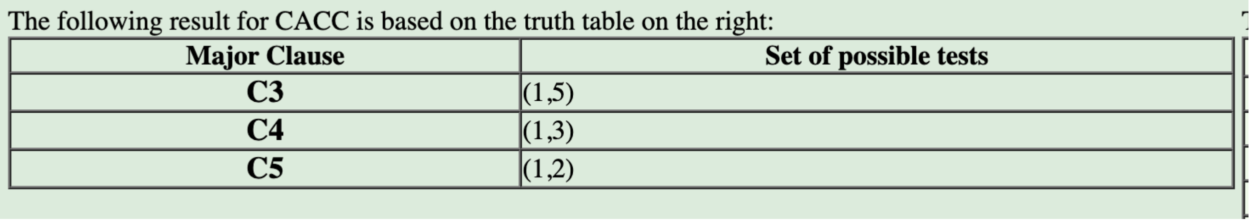
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C3	(1,5)
C4	(1,3)
C5	(1,2)

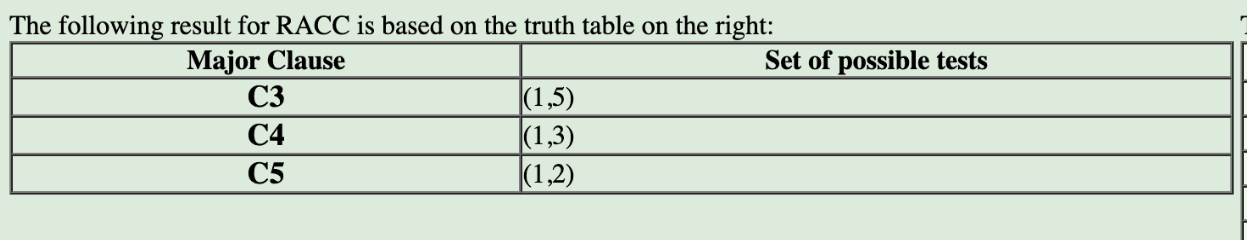
RACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C3	(1,5)
C4	(1,3)
C5	(1,2)

CACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C3	(1,5)
C4	(1,3)
C5	(1,2)

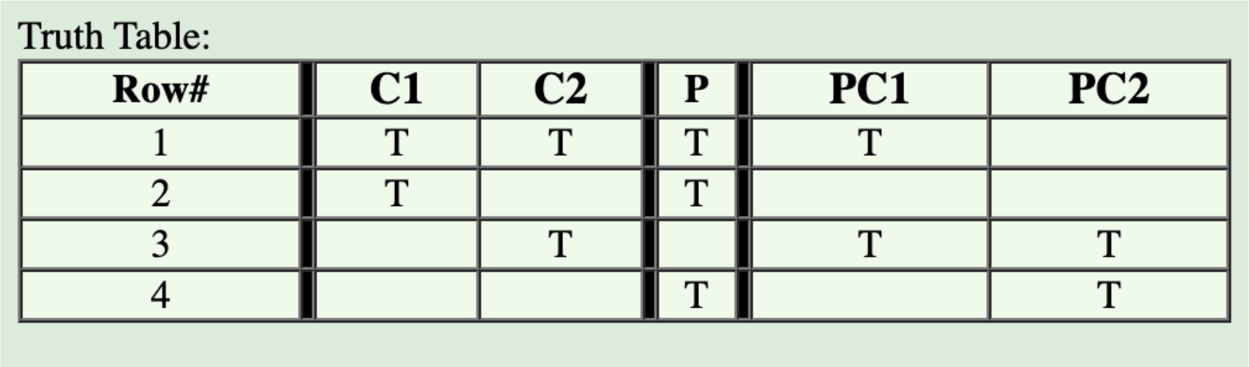
Class 4 : Proxy.java

Method 7: public Proxy(Type type, SocketAddress sa)

Testing the Decision for Code:
if ((type == Type.DIRECT) || !(sa instanceof PasswdInetSocketAddress))
    throw new IllegalArgumentException(...)

Simplify it to Logic Condition:
C1 = (type == Type.DIRECT)
C2 = (sa instanceof PasswdInetSocketAddress)
Decision: D = C1 || !C2 → throw exception if this is true

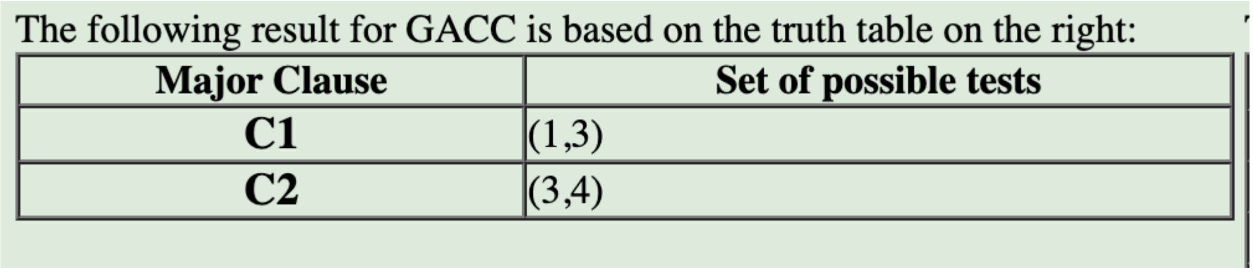
Truth Table:



Truth Table:

Row#	C1	C2	P	PC1	PC2
1	T	T	T	T	
2	T		T		
3		T		T	T
4			T		T

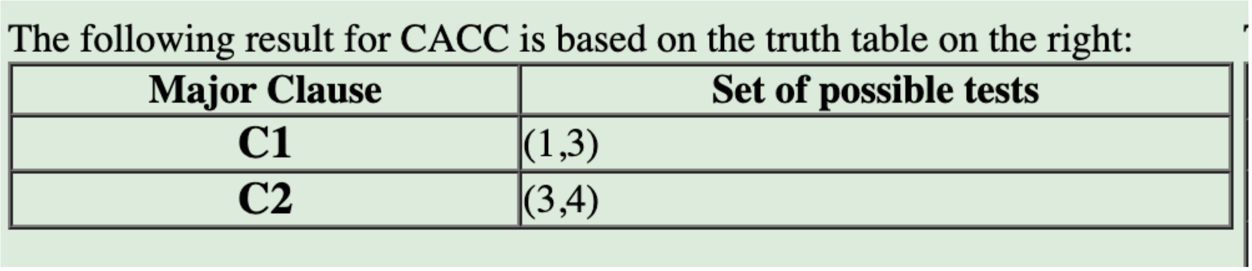
GACC;



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(3,4)

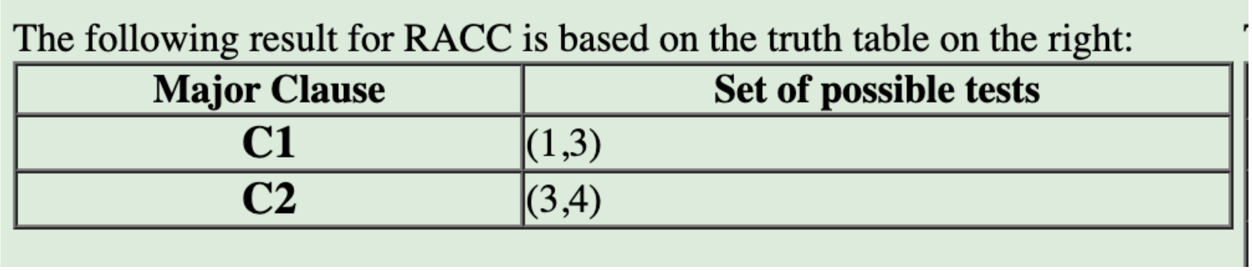
RACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(3,4)

CACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(3,4)

Method 8: public final boolean equals(Object obj)

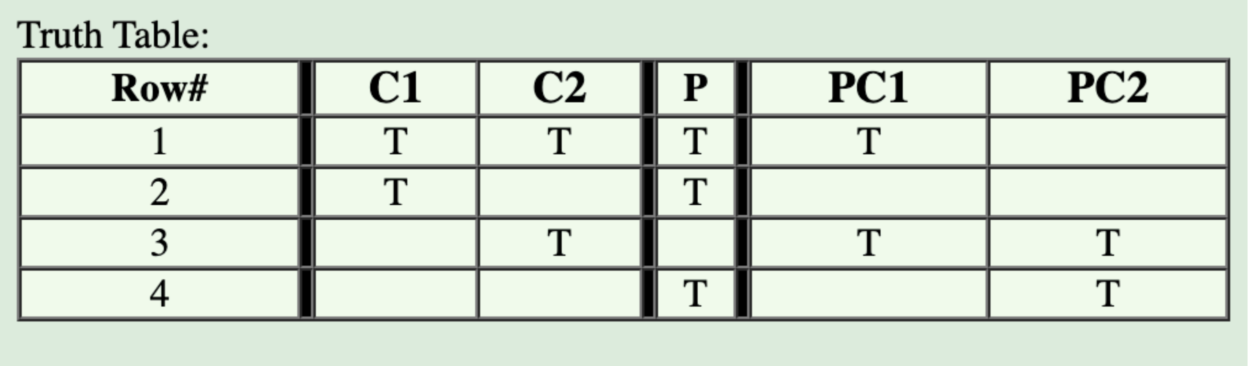
Testing the Decision for Code:

```
if (obj == null || !(obj instanceof Proxy))
    return false;
...
if (p.type() == type()) {
    if (address() == null) {
        return (p.address() == null);
    } else {
        return address().equals(p.address());
    }
}
return false;
```

Part 1: Early Exit Condition
if (obj == null || !(obj instanceof Proxy))

Simplified Logic:
C1 = obj == null
C2 = obj instanceof Proxy
Decision: C1 || !C2

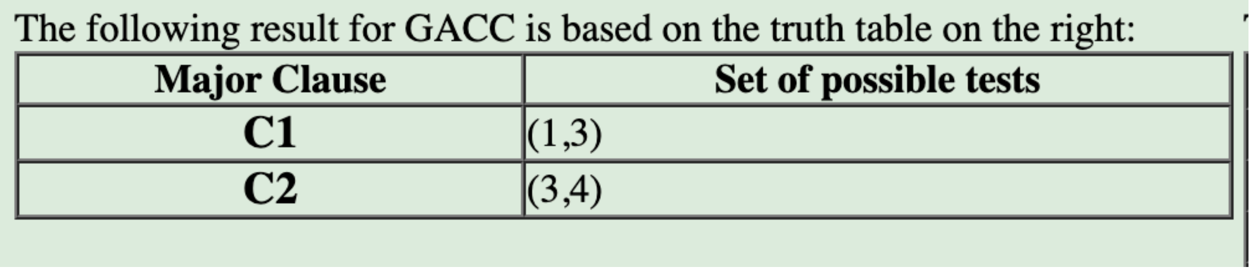
Truth Table:



Truth Table:

Row#	C1	C2	P	PC1	PC2
1	T	T	T	T	
2	T		T		
3		T		T	T
4			T		T

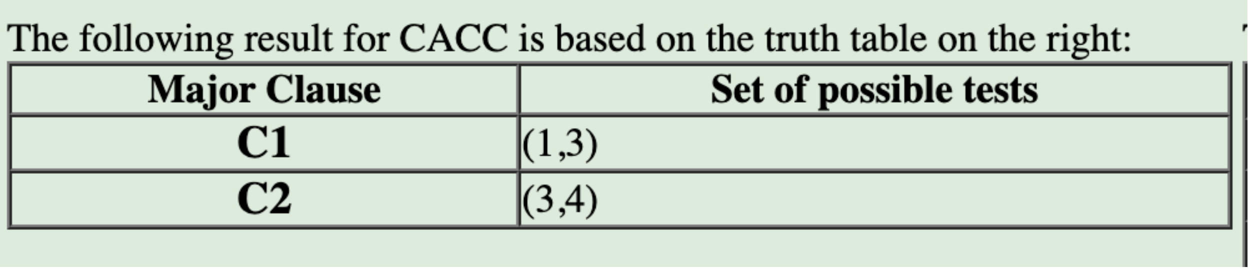
GACC;



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(3,4)

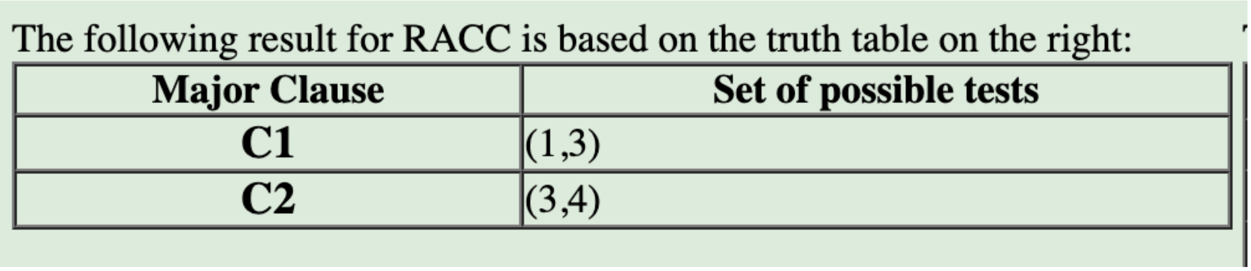
RACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,3)
C2	(3,4)

CACC:



The following result for RACC is based on the truth table on the right:

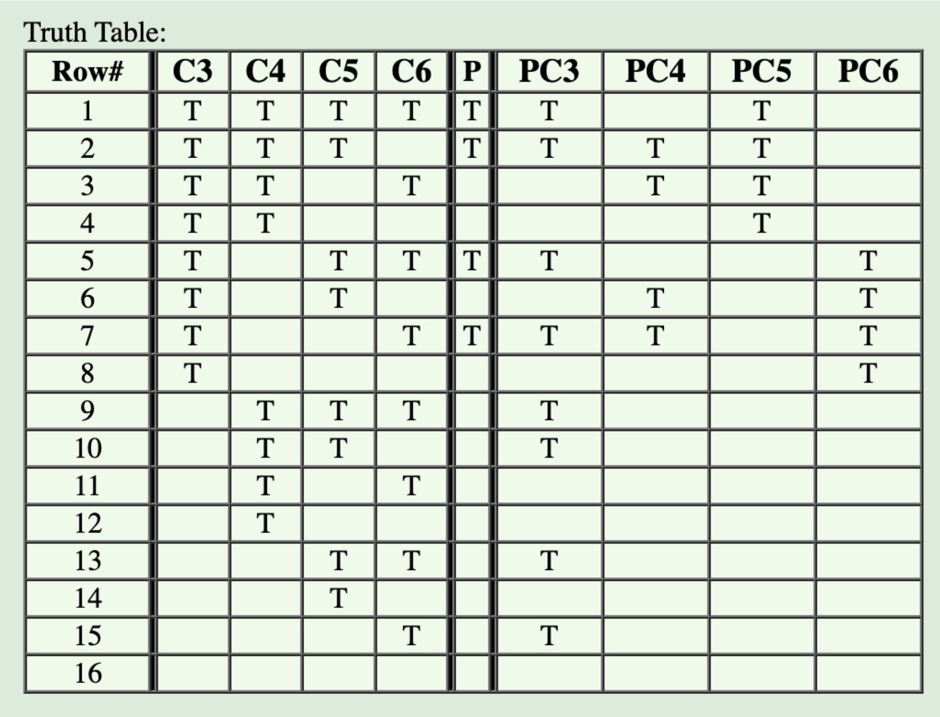
Major Clause	Set of possible tests
C1	(1,3)
C2	(3,4)

Part 2: Deep Comparison

```
if (p.type() == type()) {
   if (address() == null) {
       return (p.address() == null);
   } else {
       return address().equals(p.address());
   }
}
```

Simplified Logic:
C3 = type equals
C4 = address is null
C5 = other.address is null
C6 = address.equals(other.address)

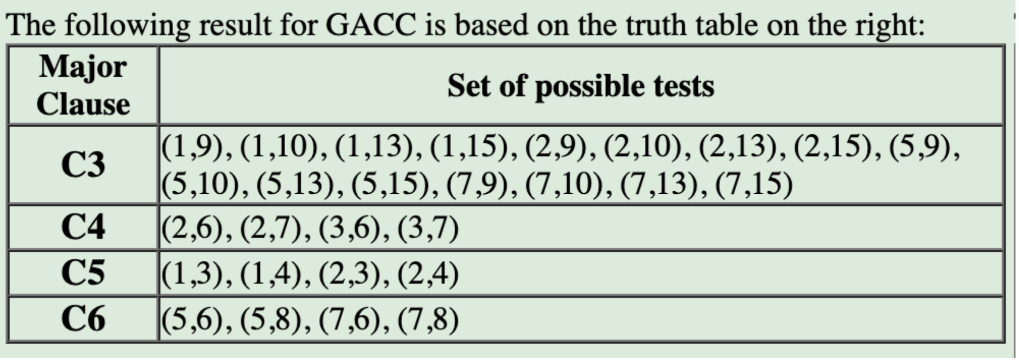
Truth Table:



Truth Table:

Row#	C3	C4	C5	C6	P	PC3	PC4	PC5	PC6
1	T	T	T	T	T	T		T	
2	T	T	T		T	T	T	T	
3	T	T		T			T	T	
4	T	T						T	
5	T		T	T	T	T			T
6	T		T				T		T
7	T			T	T	T	T		T
8	T								T
9		T	T	T		T			
10		T	T			T			
11		T		T					
12		T							
13			T	T		T			
14			T						
15				T		T			
16									

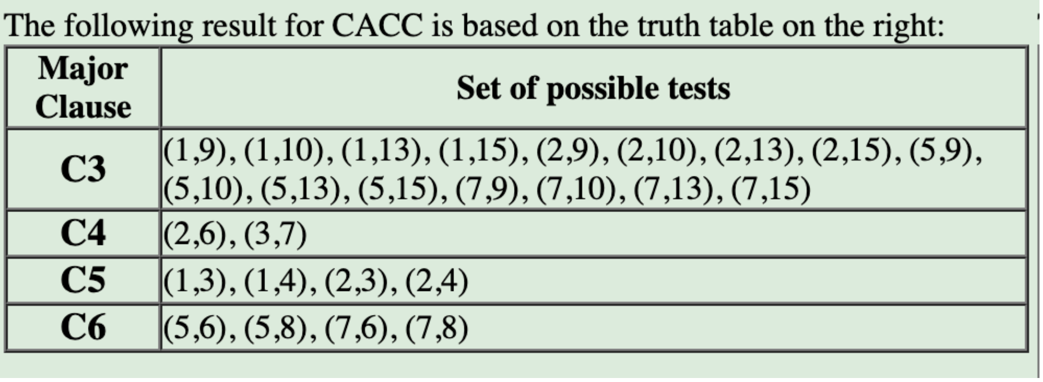
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C3	(1,9), (1,10), (1,13), (1,15), (2,9), (2,10), (2,13), (2,15), (5,9), (5,10), (5,13), (5,15), (7,9), (7,10), (7,13), (7,15)
C4	(2,6), (2,7), (3,6), (3,7)
C5	(1,3), (1,4), (2,3), (2,4)
C6	(5,6), (5,8), (7,6), (7,8)

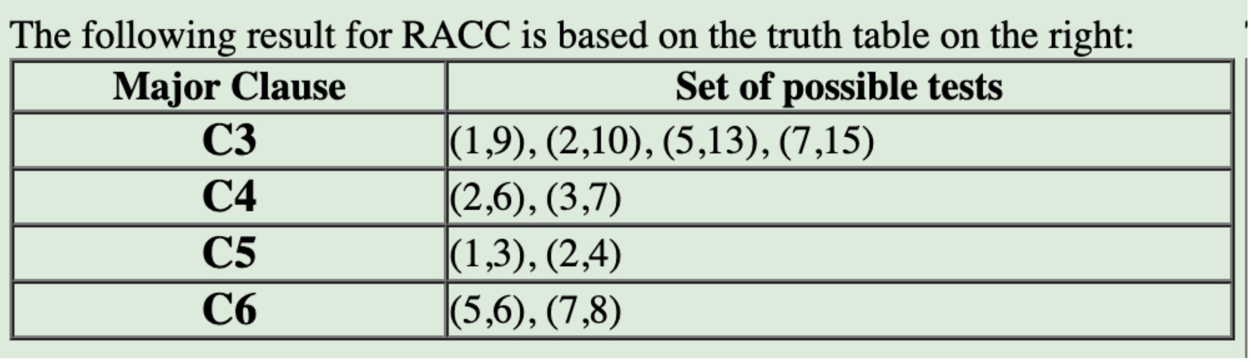
RACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C3	(1,9), (1,10), (1,13), (1,15), (2,9), (2,10), (2,13), (2,15), (5,9), (5,10), (5,13), (5,15), (7,9), (7,10), (7,13), (7,15)
C4	(2,6), (3,7)
C5	(1,3), (1,4), (2,3), (2,4)
C6	(5,6), (5,8), (7,6), (7,8)

CACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C3	(1,9), (2,10), (5,13), (7,15)
C4	(2,6), (3,7)
C5	(1,3), (2,4)
C6	(5,6), (7,8)

Class 5: ActionHelpers.java

Method 9:public static int getIconHighlightColor(Context context)

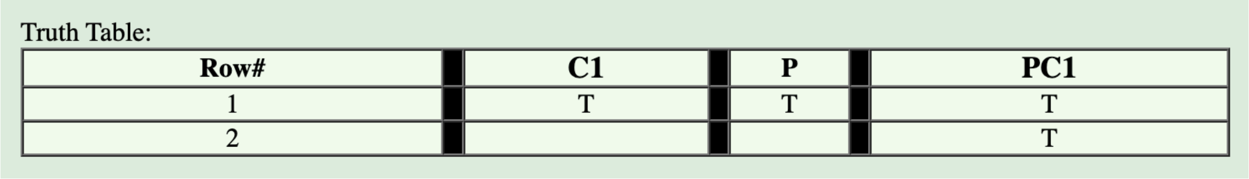
Testing the Decision for Code:

```
if (context.getTheme().resolveAttribute(R.attr.playbackControlsIconHighlightColor, outValue,
true)) {
   return outValue.data;
}
```

return ContextCompat.getColor(context, R.color.lb_playback_icon_highlight_no_theme);

Simplify to Logic Condition:
C1 = context.getTheme().resolveAttribute(...) returns true
Decision:
If C1 == true → return outValue.data
If C1 == false → return fallback color from resources

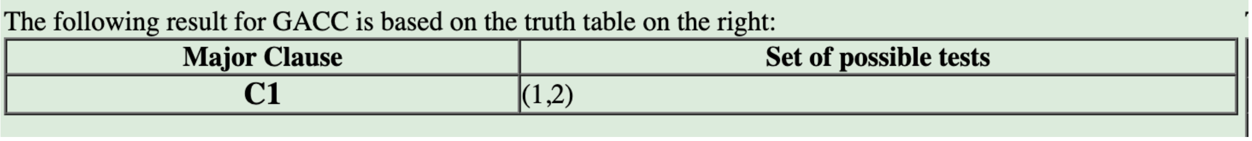
Truth Table:



Truth Table:

Row#	C1	P	PC1
1	T	T	T
2			T

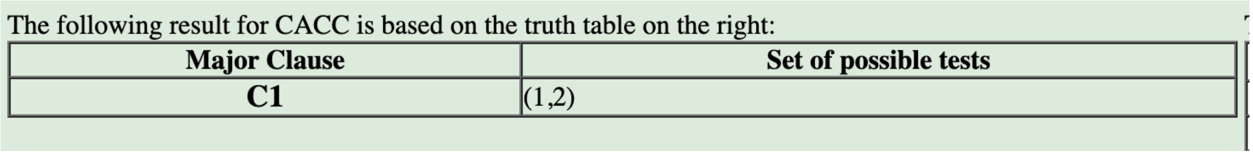
GACC:



The following result for GACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,2)

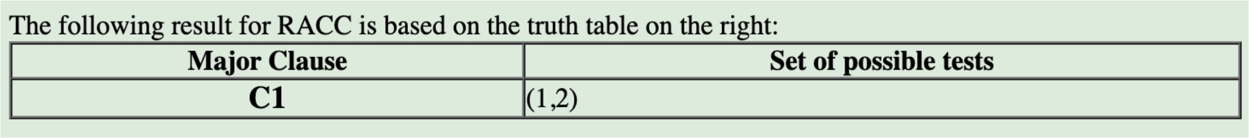
RACC:



The following result for CACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,2)

CACC:



The following result for RACC is based on the truth table on the right:

Major Clause	Set of possible tests
C1	(1,2)

Method 10: public static int getIconGrayedOutColor(Context context)

Testing the Decision for Code:
return ContextCompat.getColor(context, R.color.gray);

The method getIconGrayedOutColor(Context context) contains no conditional logic. It always returns a fixed resource value, and therefore does not require logic-based testing. No decision points can be found.