Software Testing Technique
Chapter 6

Logic Coverage

Zan Wang（王赞）

Office: 55-A413

Email: wangzan@tju.edu.cn

2022

Schedule

- **Session 9, Graph coverage. March 21**
- **Session 10, Logic coverage. March 23**
- **Session 11, Blackbox testing. March 28**
- **Session 12, Test Automation and Selenium. March 30**
- **Session 13, *Lab 3, Selenium I*. April 4**
- **Session 14, *Lab 4, Selenium II*. April 6**
- **Session 15, Load Testing and *Lab 5, Jmeter I*. April 11**
- **Session 16, *Lab 6, Jmeter2*. April 13.**

MORE DETAIL: INTRODUCTION TO SOFTWARE TESTING(J.OFFUTT) CHAPTER 8

Outline

- Introduction to Logic
- Decision Coverage and Condition Coverage
- MC/DC Coverage

INTRODUCTION TO LOGIC

Covering Logic Expressions

- Logic expressions show up in many situations
- Covering logic expressions is required for safety critical software
- Logical expressions can come from many sources
  - Decisions in programs
  - FSMs and statecharts
  - Requirements
- Tests are intended to choose some subset of the total number of truth assignments to the expressions

Logic Predicates and Clauses

- A *predicate* is an expression that evaluates to a boolean value
- Predicates can contain
  - boolean variables
  - non-boolean variables that contain >, <, ==, >=, <=, !=
  - boolean function calls
- Internal structure is created by logical operators
  - ¬ – the *negation* operator
  - ∧ – the *and* operator
  - ∨ – the *or* operator
  - → – the *implication* operator
  - ⊕ – the *exclusive or* operator
  - ↔ – the *equivalence* operator
- A *clause* is a predicate with no logical operators

Examples

- $(a < b) \lor f(z) \land D \land (m >= n*o)$
- Four clauses:
  - $(a < b)$ – relational expression
  - $f(z)$ – boolean-valued function
  - D – boolean variable
  - $(m >= n*o)$ – relational expression
- Most predicates have few clauses
- Sources of predicates
  - Decisions in programs
  - Guards in finite state machines
  - Decisions in UML activity graphs
  - Requirements, both formal and informal
  - SQL queries

CONDITION COVERAGE AND DECISION COVERAGE

Types of logic coverage

- **Decision coverage (i.e. branch coverage) 判定覆盖（也就是分支覆盖）**
- **Condition coverage 条件覆盖**
- **Condition/decision coverage (C/D) 条件判定覆盖**
- **Multiple-condition coverage 条件组合覆盖**
- **Modified condition/decision coverage (MC/DC)改进的条件判定覆盖**

Definitions: Condition and Decision

- **Decision**
  - Branching expression of the if/while/for statements
- **Condition**
  - A Boolean expression containing no Boolean operators (||, &&, !).
    - E.g., a>b
  - If the same expression appears more than once in a decision, each occurrence is considered a distinct condition.

```
if (((a>b) || G)) && (a>b))
{
     y = 0;
     x = x + 1;
}
```

Decision Coverage

- **Decision coverage concerns the coverage of *all feasible edges coming out of the decision control*.**
  - **Cover all edges in CFG**
    - **True**
    - **False**
  - **Insensitive to the logical operators (|| and &&) in the decision node.**

if(A && B) {…}
else {…}

Test suite that satisfy condition coverage:
A=true, B=false
A=false, B=true
Problem: branch coverage not achieved

```
int foo(int x, int y) {
        int z = y*2; \\ z=y;
        if ((x>5) && (y>0)) {
                z = x; }
        return x*z;
}
```

Condition Coverage

- **Condition coverage concerns the coverage of each condition taking both *true* and *false*.**
  - **Does not consider constant condition, such as (true) and (x==x).**
  - **Condition coverage does not subsume decision coverage.**
  - **What is the number of test cases to achieve 100% condition coverage?**
    - **2**

```
if ( (A || B) && C ) { …}
else {…}
```

Test suite that satisfy condition coverage:
A=true, B=false, C=false
A=false, B=true, C=true

```
if(A && B) {…}
else {…}
```

Test suite that satisfy condition coverage:
A=true, B=false
A=false, B=true
Problem: branch coverage not achieved

```
int foo(int x, int y) {
    int z = y*2; \\ z=y;
    if ((x>5) && (y>0)) {
            z = x; }
    return x*z;
}
```

Condition/Decision Coverage

- **Simply condition coverage + decision coverage**
  - **For each condition, the test suite covers both *true* and *false*.**
  - **For the whole decision, the test suite covers both *true* and *false*.**
  - **C/DC(P)=CC(P) ∩ DC(P)**

```
if ( (A || B) && C ) { …}
else {…}
```

Test suite that satisfy ***both*** condition coverage ***and*** condition/decision coverage.

A=true, B=false, C=false
A=false, B=true, C=true

```
if(A && B) {…}
else {…}
```

Test suite that satisfy condition coverage, but not condition/decision coverage.

A=true, B=false
A=false, B=true

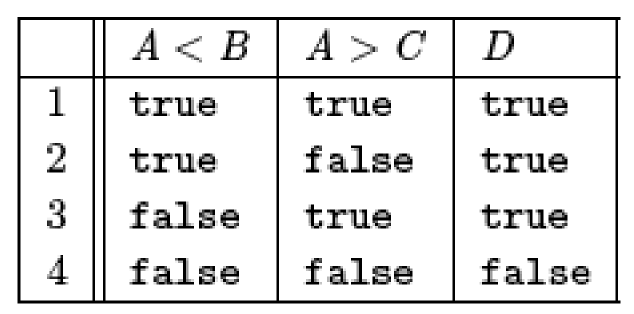
```
int foo(int x, int y) {
    int z = y*2; \\ z=y;
    if ((x>5) && (y>0)) {
            z = x; }
    return x*z;
}
```

Multiple Condition Coverage

- Cover all possible combinations of conditions.
  - Some combinations are not possible because coupled conditions. e.g. (x>0 && x>0)
  - If a decision D has k uncoupled conditions, the total number of combinations 2^k
  - Like all path coverage, it is not practical.

Multiple Condition Coverage: Example

- Consider D=(A<B) OR (A>C) composed of two simple conditions A< B and A> C --- there are four possible combinations of the outcomes of these two simple.



	$A < B$	$A > C$	D
1	true	true	true
2	true	false	true
3	false	true	true
4	false	false	false

$$T = \left\{ \begin{array}{lllll} t_1: & < A = 2 & B = 3 & C = 1 > \\ t_2: & < A = 2 & B = 1 & C = 3 > \end{array} \right\}$$

T or T

F or T

Does T cover all four combinations?

No

Does T' cover all four combinations?

Yes

$$T' = \left\{ \begin{array}{lllll} t_1: & < A = 2 & B = 3 & C = 1 > \\ t_2: & < A = 2 & B = 1 & C = 3 > \\ t_3: & < A = 2 & B = 3 & C = 5 > \\ t_4: & < A = 2 & B = 1 & C = 5 > \end{array} \right\}$$

T or T

F or T

T or F

F or F

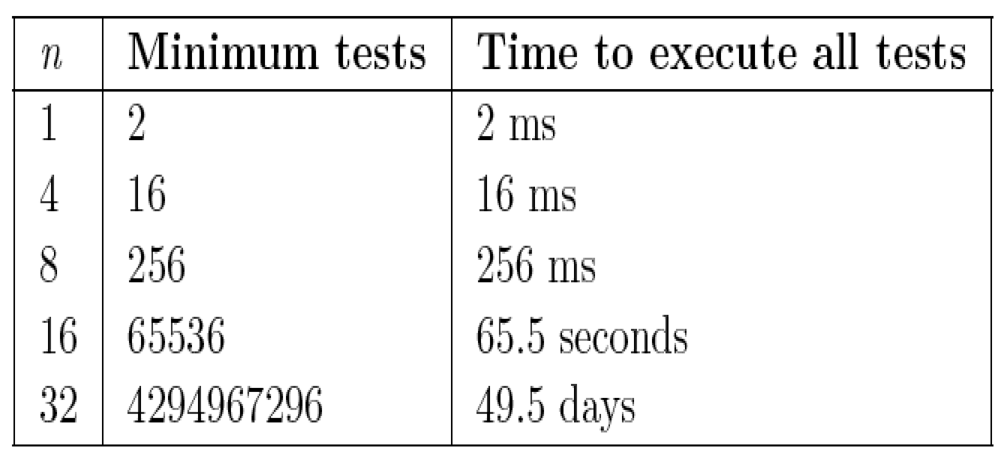
Multiple Condition Coverage: Definition

- Suppose that the program under test contains a total of n decisions and that each decision contains k1, k2, …, kn simple conditions.
- Decision *i* will have a total of 2^{ki} combinations.
- The total number of combinations to be covered is $\sum_{i=1}^{n} 2^{k_i}$

MC/DC COVERAGE

Modified Condition/Decision (MC/DC) Coverage

- Obtaining multiple condition coverage might become expensive when there are many embedded simple conditions.
- If a compound condition C contains n simple conditions, the maximum number of tests required to cover C is 2^n .

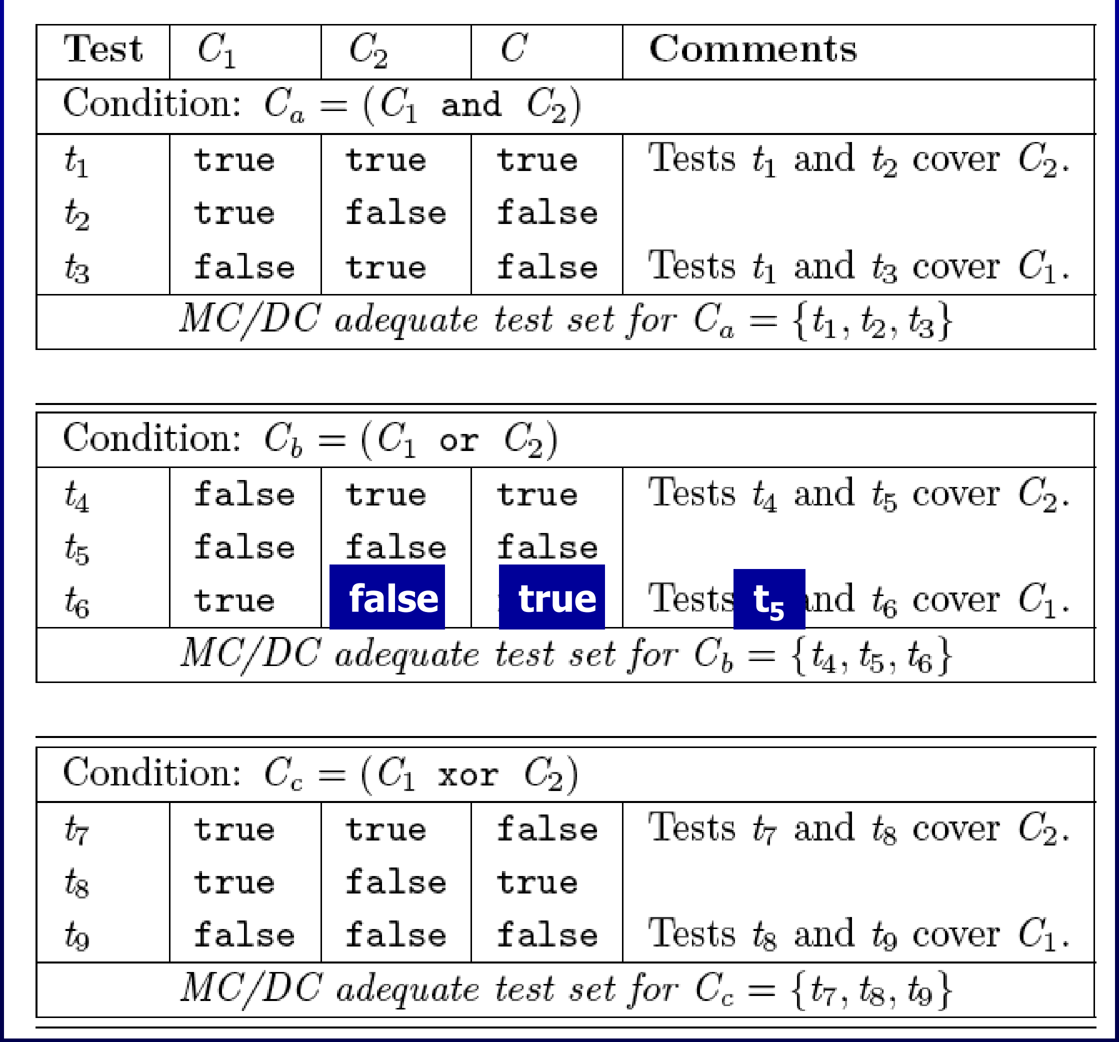


n	Minimum tests	Time to execute all tests
1	2	2 ms
4	16	16 ms
8	256	256 ms
16	65536	65.5 seconds
32	4294967296	49.5 days

Compound conditions and MC/DC

- **MC/DC coverage requires that every compound condition in a program must be tested by demonstrating that each simple condition within the compound condition has an independent effect on its outcome.**
- **Thus, MC/DC coverage is a weaker criterion than the multiple condition coverage criterion.**

MC/DC coverage: Simple conditions

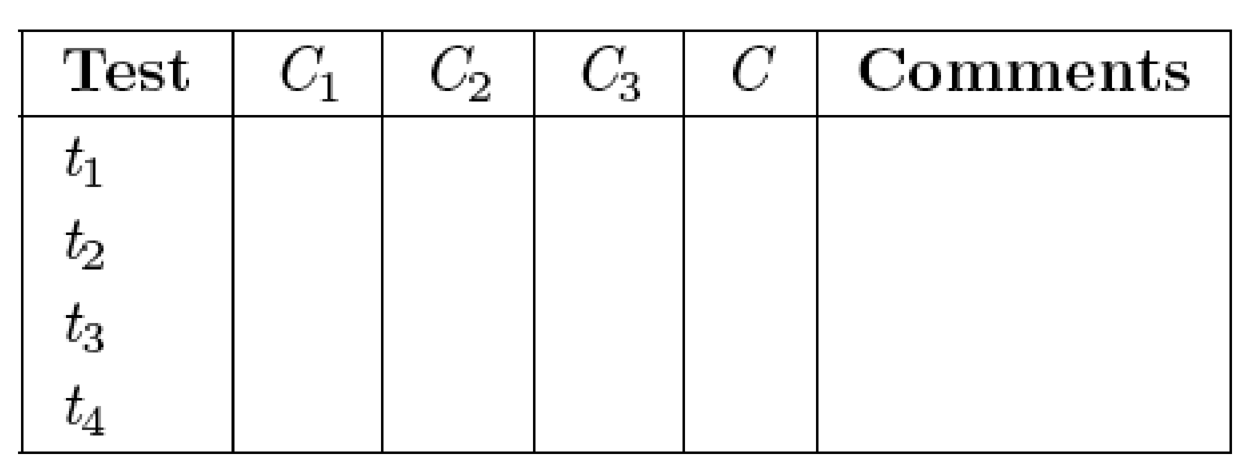


Test	C_1	C_2	C	Comments
Condition: $C_a = (C_1$ and $C_2)$				
t_1	true	true	true	Tests t_1 and t_2 cover C_2.
t_2	true	false	false	
t_3	false	true	false	Tests t_1 and t_3 cover C_1.
MC/DC adequate test set for $C_a = \{t_1, t_2, t_3\}$				
Condition: $C_b = (C_1$ or $C_2)$				
t_4	false	true	true	Tests t_4 and t_5 cover C_2.
t_5	false	false	false	
t_6	true	false	true	Tests t_5 and t_6 cover C_1.
MC/DC adequate test set for $C_b = \{t_4, t_5, t_6\}$				
Condition: $C_c = (C_1$ xor $C_2)$				
t_7	true	true	false	Tests t_7 and t_8 cover C_2.
t_8	true	false	true	
t_9	false	false	false	Tests t_8 and t_9 cover C_1.
MC/DC adequate test set for $C_c = \{t_7, t_8, t_9\}$				

MC/DC coverage: Generating tests for compound conditions

- If C=C1 AND C2 AND C3, create a table with five columns and four rows. Label the columns as Test, C1, C2, C3, and C, from left to right. An optional column “Comments” may be added.

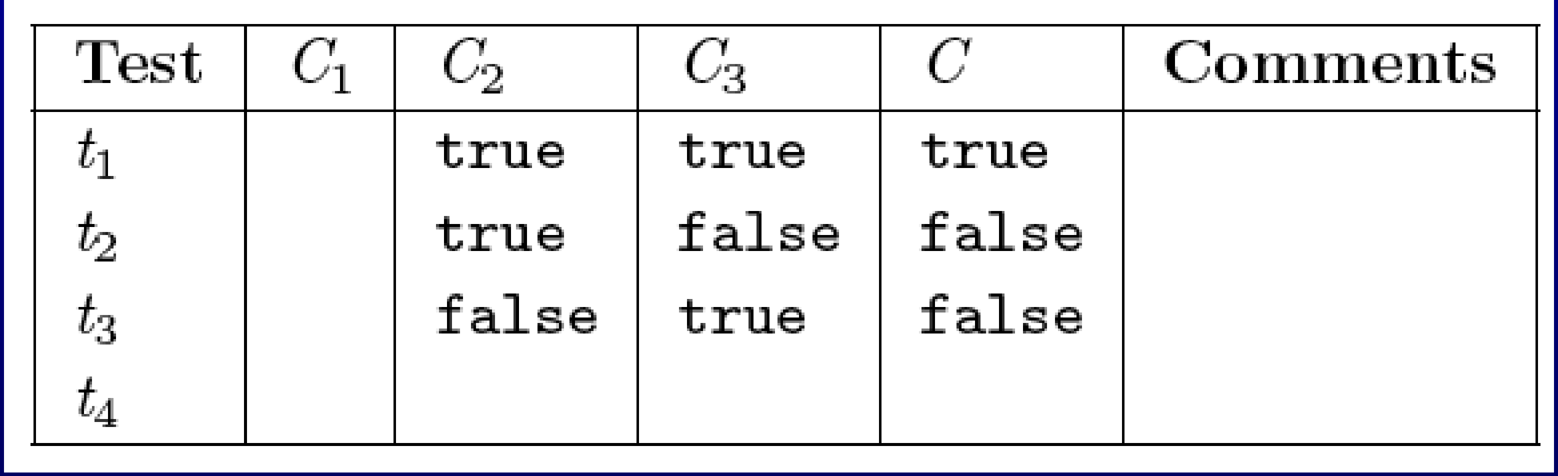
- The column labeled Test contains rows labeled by test case numbers t1 through t4 . The remaining entries are empty.



Test	C_1	C_2	C_3	C	Comments
t_1					
t_2					
t_3					
t_4					

MC/DC coverage: Generating tests for compound conditions (contd.)

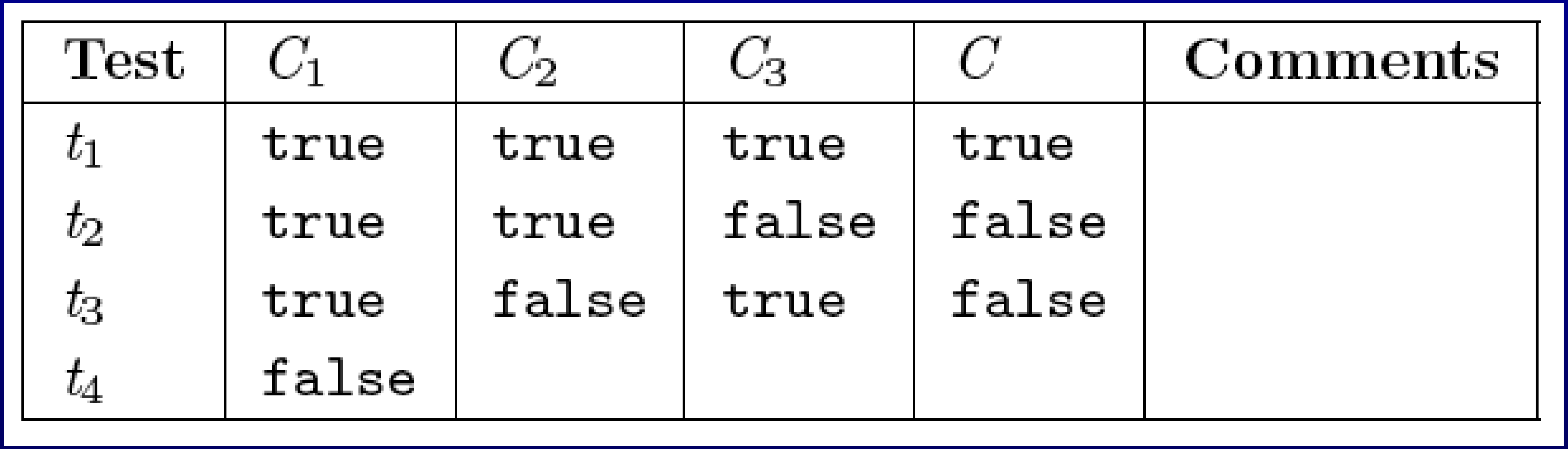
- Copy all entries in columns C1, C2, and C from the table for simple conditions into columns C2, C3, and C of the empty table.



Test	C_1	C_2	C_3	C	Comments
t_1		true	true	true	
t_2		true	false	false	
t_3		false	true	false	
t_4					

MC/DC coverage: Generating tests for compound conditions (contd.)

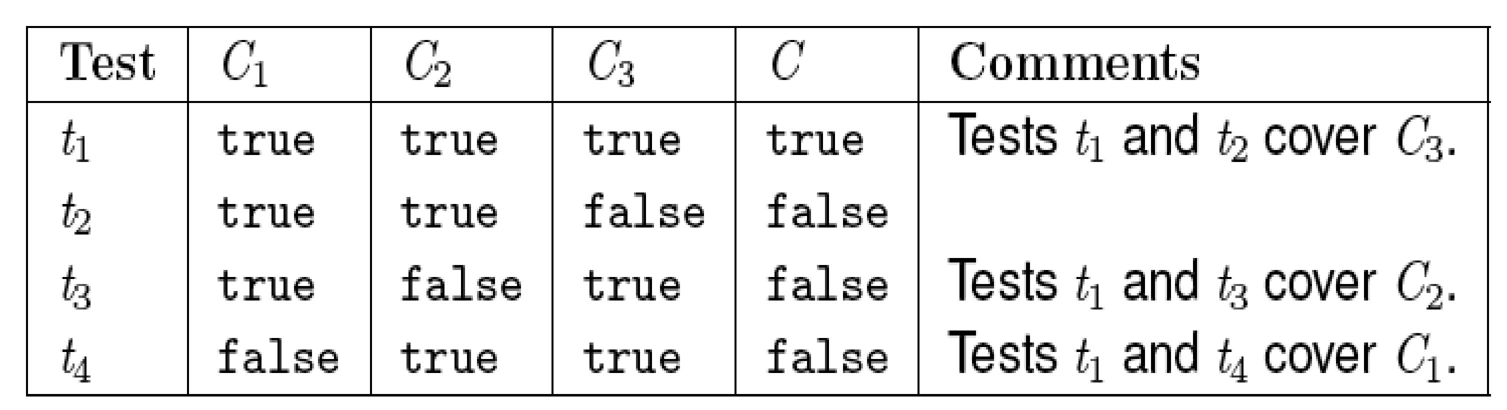
- Fill the first three rows in the column marked C1 with true and the last row with false.



Test	C_1	C_2	C_3	C	Comments
t_1	true	true	true	true	
t_2	true	true	false	false	
t_3	true	false	true	false	
t_4	false				

MC/DC coverage: Generating tests for compound conditions (contd.)

- Fill the last row under columns labeled C2, C3, and C with true, true, and false, respectively.



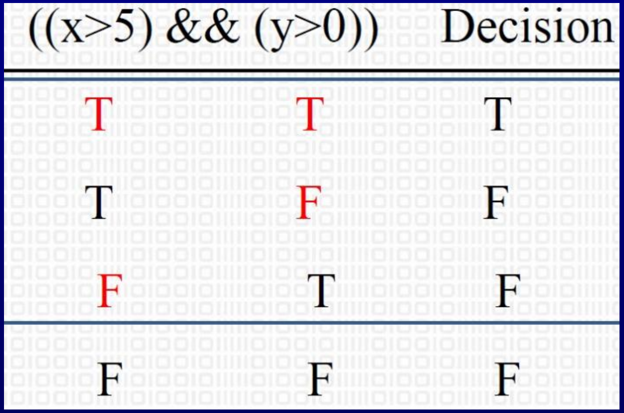
Test	C_1	C_2	C_3	C	Comments
t_1	`true`	`true`	`true`	`true`	Tests t_1 and t_2 cover C_3.
t_2	`true`	`true`	`false`	`false`	
t_3	`true`	`false`	`true`	`false`	Tests t_1 and t_3 cover C_2.
t_4	`false`	`true`	`true`	`false`	Tests t_1 and t_4 cover C_1.

We now have a table containing MC/DC adequate tests for C=(C1 AND C2 AND C3) derived from tests for C=(C1 AND C2) .

MC/DC coverage: Generating tests for compound conditions (contd.)

- The procedure illustrated above can be extended to derive tests for any compound condition using tests for a simpler compound condition.

```
int foo(int x, int y) {
     int z = y*2; \\ z=y;
     if ((x>5) && (y>0)) {
               z = x; }
     return x*z;
}
```



((x>5) && (y>0))		Decision
T	T	T
T	F	F
F	T	F
F	F	F

MC/DC coverage: Summary

- A test set T for program P written to meet requirements R is considered adequate with respect to the MC/DC coverage criterion if, upon the execution of P on each test in T, the following requirements are met:
  - Each block in P has been covered.
  - Each simple condition in P has taken both true and false values.
  - Each decision in P has taken all possible outcomes.
  - Each simple condition within a compound condition C in P has been shown ***to independently affect*** the outcome of C.
  - This is the MC part of the coverage we discussed.

Homework 4

Given the program listed below, please design three test sets according to the following coverage criterion :

1. Condition Coverage
2. Decision Coverage
3. Modified C/D Coverage

```
public double Calc(int a, int b,double c) {
                    double d = 0;
                    if (a>0 && b>0) {c = c/a;}
                    if (a>1 || c>1) {c = c +1;}

                    d = b + c;
                    return d;
          }
```