Statement Coverage

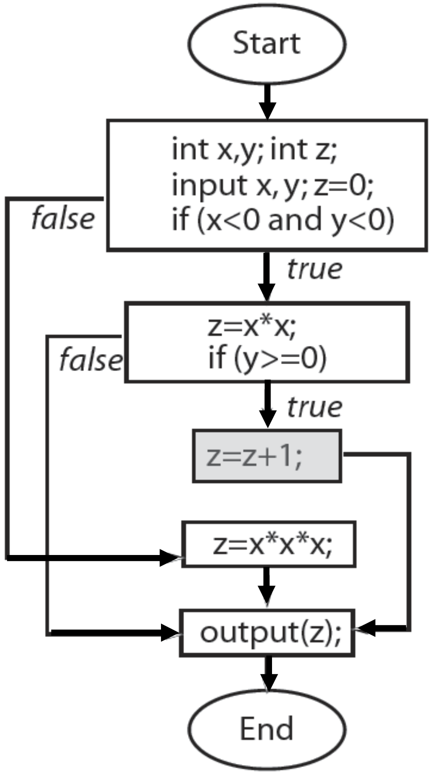
Statement Coverage: Computation

- The statement coverage of a test set *T* with respect to (*P*, *R*) is computed as $S_c/(S_e-S_i)$, where
  - S_c is the number of statements covered,
  - S_i is the number of *infeasible (unreachable) statements*, and
  - S_e is the *total number of executable statements* in the program to be tested.

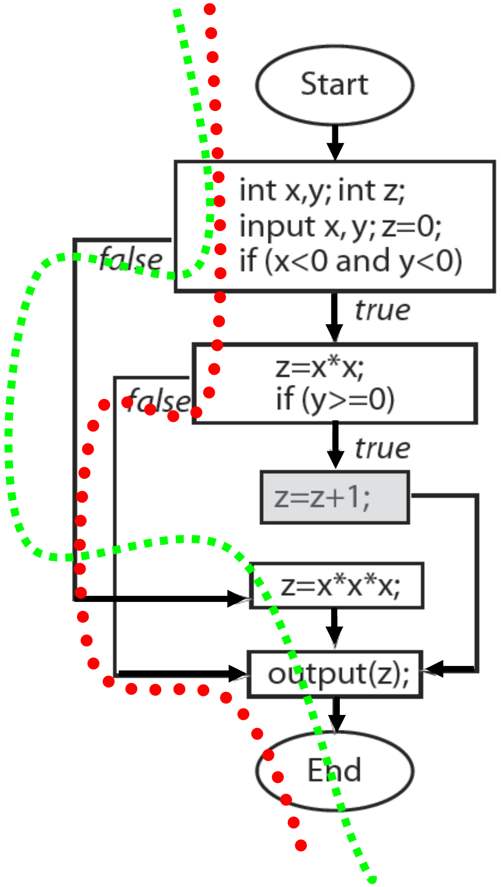
- *T* is considered adequate with respect to the statement coverage criterion if the statement coverage of *T* with respect to (*P*, *R*) is 1 (i.e., 100%).

Statement Coverage: An Example

```
begin
  int X, Y;
  int Z;
  input (X, Y);
  Z = 0;
  if (X < 0 and Y < 0) {
     Z = X*X;
     if (Y >= 0)
        Z = Z + 1;
  }
  else
     Z = X*X*X;
  output (Z);
end
```



Statement Coverage: An Example (cont'd)



- S_e={4, 5, 6, 7, 8, 9, 12, 13}
  Let T_1={t_1:<x= –1, y= – 1>, t_2:<x=1, y=1>}
- Statements covered:
  - t_1: follow the red line
  - t_2: follow the green line
- S_c=7, S_i=1, S_e=8. The statement coverage for T_1 is 7/(8 – 1)=1
- Note that *9 is infeasible*
- Thus, we conclude that T_1 is adequate with respect to the statement coverage criterion.

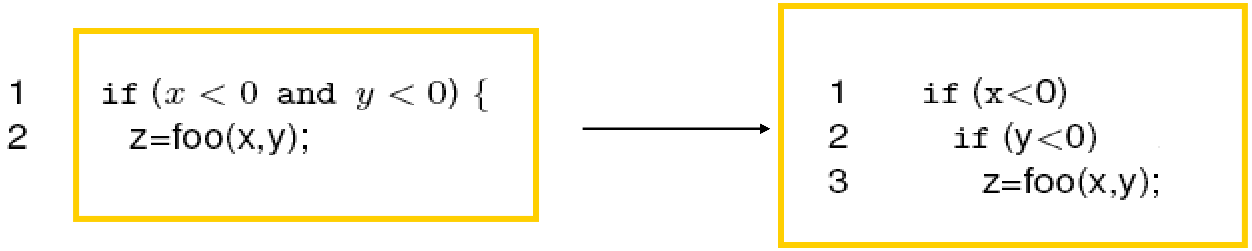
Decision Coverage

Decision versus Simple Condition

- A decision can be composed of *a simple condition* such as $x<0$, or of *a more complex condition*, such as ($x<0$ AND $y<0$).

- A simple condition is considered covered if it evaluates to *true* and *false* in one or more executions of the program in which it occurs.

- A *compound condition* is considered covered if each *simple condition* it is comprised of is also covered.



```
if (x < 0 and y < 0) {
  z=foo(x,y);
```

```
if (x<0)
  if (y<0)
    z=foo(x,y);
```

one decision &
two simple conditions

two decisions &
two simple conditions

How to Cover a Decision?

- A decision is considered covered if the flow of control has been diverted to all possible destinations that correspond to this decision, i.e., *all outcomes of the decision have been taken.*

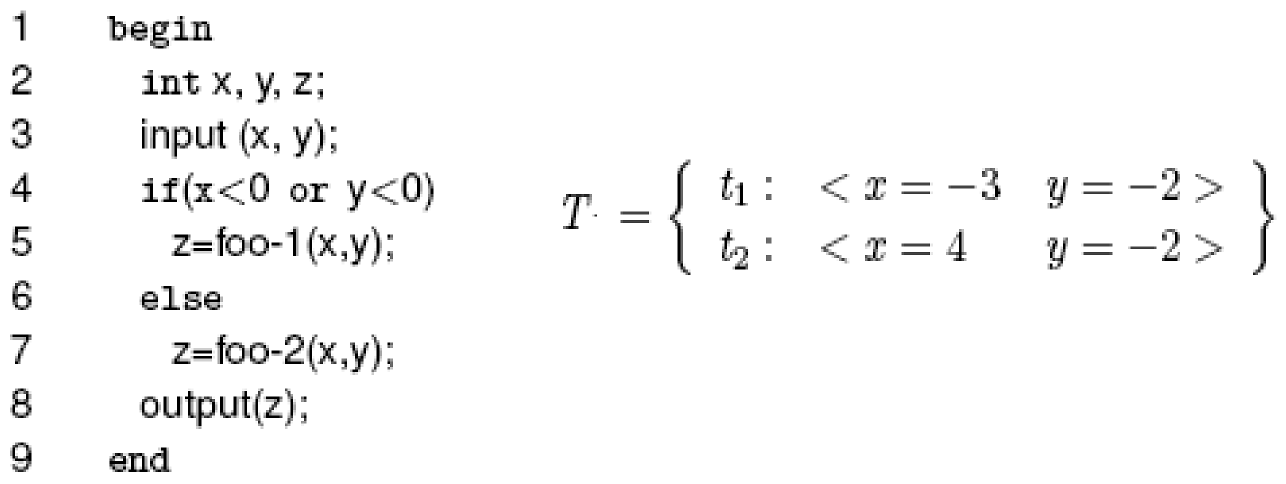
- This implies that, for example, the expression in the *if* or a *while* statement has evaluated to *true* in some execution of the program under test and to *false* in the same or another execution.
  - Examples
    - ❑ if (x < 0) {…..}
    - ❑ if ((x<0) and (y>0) {…..}
    - ❑ while (x < 3) {…..}

Decision Coverage: Computation

- The decision coverage of a test set *T* with respect to (*P*, *R*) is computed as $D_c/(D_e\text{-}D_i)$, where
  - D_c is the number of decisions covered,
  - D_i is the number of *infeasible decisions*, and
  - D_e is the *total number of decisions* in the program to be tested.
- *T* is considered adequate with respect to the decision coverage criterion if the decision coverage of *T* with respect to (*P*, *R*) is 1 (i.e., 100%).

Quiz

- Consider the following program and a test set



```
begin
  int x, y, z;
  input (x, y);
  if(x<0 or y<0)
    z=foo-1(x,y);
  else
    z=foo-2(x,y);
  output(z);
end
```

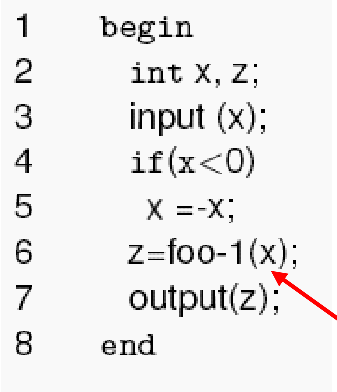
$$T = \left\{ \begin{array}{lll} t_1: & < x = -3 & y = -2 > \\ t_2: & < x = 4 & y = -2 > \end{array} \right\}$$

- Is *T* adequate with respect to decision coverage?

Statement Coverage
versus
Decision Coverage

A Program with a Bug

- This following program inputs an integer *x*
  - if *x* < 0, transforms it into a positive value before invoking foo-1 to compute the output *z*
  - if *x*≥0, compute z using foo-2



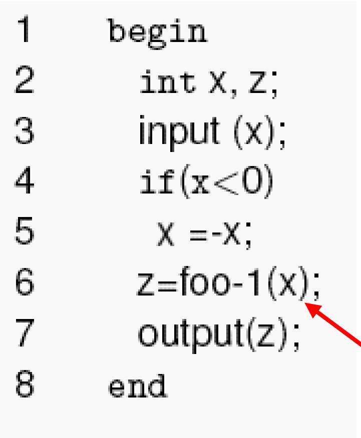
```
begin
  int x, z;
  input (x);
  if(x<0)
   x =-x;
  z=foo-1(x);
  output(z);
end
```

Where is the bug?

There should have been an else clause for *x*≥0 before this statement.

Is Statement Coverage Sufficient?

- Consider a test set T={t_1:<x= –5>}.
- It is adequate with respect to *statement* coverage criterion, but does not reveal the bug.

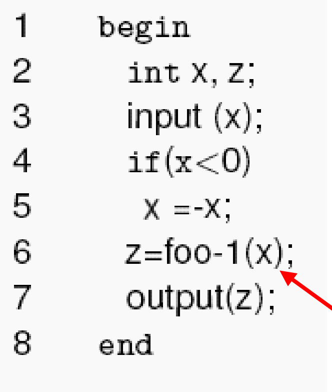


```
begin
  int x, z;
  input (x);
  if(x<0)
   x =-x;
  z=foo-1(x);
  output(z);
end
```

There should have been an *else* clause for *x*≥0 before this statement.

Is Decision Coverage Sufficient?

- Consider another test set $T'=\{t_1{:}<x = -5>\ \ t_2{:}<x = 3>\}$
- T' is decision adequate, but not T.
- Also, T' reveals the bug, but not T.



```
begin
  int x, z;
  input (x);
  if(x<0)
    x =-x;
  z=foo-1(x);
  output(z);
end
```

There should have been an *else* clause for $x \geq 0$ before this statement.

- This example illustrates *how and why decision coverage might help in revealing a bug that is not revealed* by a test set adequate with respect to *statement coverage*.