CS 6340 – Spring 2013 – Assignment 8 **Name** Sam Britt, Shriram Swaminathan,
Assigned: March 4, 2013
Due: March 13, 2013 **Name** and Sivaramachandran Ganesan

At the beginning of class on the due date, submit your neatly presented solution with this page stapled to the front (100 points).

Part 1

Your new position as Test Manager requires that you establish a set of requirements that developers will use for unit testing of the software that they write. Before you establish these requirements, you want to assess the fault-detection ability, expense, tool availability, etc. of various techniques that have been proposed in the literature. To do this, you will use a program, which we'll call **tritype**, that has the following requirements specification

> **tritype** takes as input three integer values. The three values are interpreted as representing the lengths of the sides of a triangle. The program prints a message that states whether the triangle is scalene, isosceles, or equilateral.

You are to do the following:

1. Use the specification to develop a set of test cases (a test suite) for **tritype** using two black box testing methods (both described in "EquivalencePartitioningBoundaryValue:"
   - Equivalence Partitioning
   - Boundary Value Analysis

2. Create a file of test cases (reason for test (in quotes), inputs, expected outputs) that consists of one test case per line; the number of the test case will be the line number in the file. For example, suppose I created two test cases:
   Test Case 1: isosceles 2 2 3 isosceles
   Test Case 2: equilateral 4 4 4 equilateral
   The file should contain
   "isosceles" 2 2 3 isosceles
   "equilateral" 4 4 4 equilateral

3. Send the test cases to Sangmin, and he'll send you the **tritype** program for the second part of the assignment. Let him know whether you want the C version or the Java version.

Sam Britt
Shriram Swaminathan
Sivaramachandran Ganesan

CS 6340
Assignment 8
Mar. 12, 2013

Our test suite is shown in Table 1 below. We achieved 100 % statement coverage with this test suite; Table 2 shows which tests cover each statement.

Our test suite achieved 65.8 % multiple condition coverage. The coverage, per condition, is listed in Table 3. When considering just the subset of conditions necessary for MC/DC coverage, our suite achieves 92.6 % coverage. These data are tabulated in Table 4.

The CFG for `tritype.c` can be seen in Figure 1.

Table 1: Test Suite

Test ID	Reason for test	Input: (i, j, k)	Expected Output	Test Result
1	Violation of triangle inequality	$(10, 1, 1)$	invalid	PASS
2	Violation of triangle inequality	$(1, 10, 1)$	invalid	PASS
3	Violation of triangle inequality	$(1, 1, 10)$	invalid	PASS
4	Negative input	$(-1, 1, 1)$	invalid	PASS
5	Negative input	$(1, -1, 1)$	invalid	PASS
6	Negative input	$(1, 1, -1)$	invalid	PASS
7	Zero input	$(0, 1, 1)$	invalid	PASS
8	Zero input	$(1, 0, 1)$	invalid	PASS
9	Zero input	$(1, 1, 0)$	invalid	FAIL
10	Invalid input	$(\mathtt{w}, 1, 1)$	invalid	PASS
11	Invalid input	$(1, \mathtt{w}, 1)$	invalid	PASS
12	Invalid input	$(1, 1, \mathtt{w})$	invalid	FAIL
13	Equilateral	$(1, 1, 1)$	equilateral	PASS
14	Scalene	$(4, 3, 2)$	scalene	PASS
15	Isosceles	$(2, 2, 3)$	isosceles	PASS
16	Isosceles	$(2, 3, 2)$	isosceles	PASS
17	Isosceles	$(3, 2, 2)$	isosceles	PASS

Table 2: Statement Coverage per Test

Test ID	Statement line number																						
	33	41	43	47	48	49	50	51	52	53	55	63	64	66	74	75	76	77	78	79	80	81	83	86
1	✓	✓		✓	✓		✓		✓	✓	✓				✓		✓		✓		✓		✓	✓
2	✓	✓		✓	✓		✓	✓	✓		✓				✓		✓		✓		✓		✓	✓
3	✓	✓		✓	✓	✓	✓		✓		✓				✓		✓		✓		✓		✓	✓
4	✓	✓	✓																					✓
5	✓	✓	✓																					✓
6	✓	✓	✓																					✓
7	✓	✓	✓																					✓
8	✓	✓	✓																					✓
9	✓	✓		✓	✓	✓	✓		✓		✓				✓		✓	✓						✓
10	✓	✓	✓																					✓
11	✓	✓		✓	✓		✓		✓		✓	✓	✓											✓
12	✓	✓		✓	✓	✓	✓		✓		✓				✓		✓	✓						✓
13	✓	✓		✓	✓		✓	✓	✓	✓	✓				✓	✓								✓
14	✓	✓		✓	✓		✓		✓		✓	✓		✓										✓
15	✓	✓		✓	✓	✓	✓		✓		✓				✓		✓	✓						✓
16	✓	✓		✓	✓		✓	✓	✓		✓				✓		✓		✓	✓				✓
17	✓	✓		✓	✓		✓		✓	✓	✓				✓		✓		✓		✓	✓		✓

Table 3: Multiple Condition Coverage, per Condition

Decision (Result)	Coverage	Conditions (Possible Evaluations)		
Line 41	Covered?	`i<=0`	`j<=0`	`k<0`
F	✓	F	F	F
T	✓	F	F	T
T	✓	F	T	F
T		F	T	T
T	✓	T	F	F
T		T	F	T
T		T	T	F
T		T	T	T
Line 48	Covered?	`i == j`		
T	✓	T		
F	✓	F		
Line 50	Covered?	`i==k`		
T	✓	T		
F	✓	F		
Line 52	Covered?	`j==k`		
T	✓	T		
F	✓	F		
Line 55	Covered?	`triang==0`		
T	✓	T		
F	✓	F		

Decision (Result)	Coverage	Conditions (Possible Evaluations)		
Line 63	Covered?	`i+j<=k`	`j+k<=i`	`i+k<j`
F	✓	F	F	F
T	✓	F	F	T
T		F	T	F
T		F	T	T
T		T	F	F
T		T	F	T
T		T	T	F
T		T	T	T
Line 74	Covered?	`triang>3`		
T	✓	T		
F	✓	F		
Line 76	Covered?	`triang==1`	`i+j>k`	
F		F	F	
F	✓	F	T	
F	✓	T	F	
T	✓	T	T	
Line 78	Covered?	`triang==2`	`i+k>j`	
F		F	F	
F	✓	F	T	
F	✓	T	F	
T	✓	T	T	
Line 80	Covered?	`triang==3`	`j+k>i`	
F		F	F	
F	✓	F	T	
F	✓	T	F	
T	✓	T	T	

Table 4: MC/DC Coverage, per Condition

Decision (Result)	Coverage	Conditions (Possible Evaluations)		
Line 41	Covered?	`i<=0`	`j<=0`	`k<0`
F	✓	F	F	F
T	✓	F	F	T
T	✓	F	T	F
T	✓	T	F	F
Line 48	Covered?	`i == j`		
T	✓	T		
F	✓	F		
Line 50	Covered?	`i==k`		
T	✓	T		
F	✓	F		
Line 52	Covered?	`j==k`		
T	✓	T		
F	✓	F		
Line 55	Covered?	`triang==0`		
T	✓	T		
F	✓	F		

Decision (Result)	Coverage	Conditions (Possible Evaluations)		
Line 63	Covered?	`i+j<=k`	`j+k<=i`	`i+k<j`
F	✓	F	F	F
T	✓	F	F	T
T		F	T	F
T		T	F	F
Line 74	Covered?	`triang>3`		
T	✓	T		
F	✓	F		
Line 76	Covered?	`triang==1`	`i+j>k`	
F	✓	F	T	
F	✓	T	F	
T	✓	T	T	
Line 78	Covered?	`triang==2`	`i+k>j`	
F	✓	F	T	
F	✓	T	F	
T	✓	T	T	
Line 80	Covered?	`triang==3`	`j+k>i`	
F	✓	F	T	
F	✓	T	F	
T	✓	T	T	

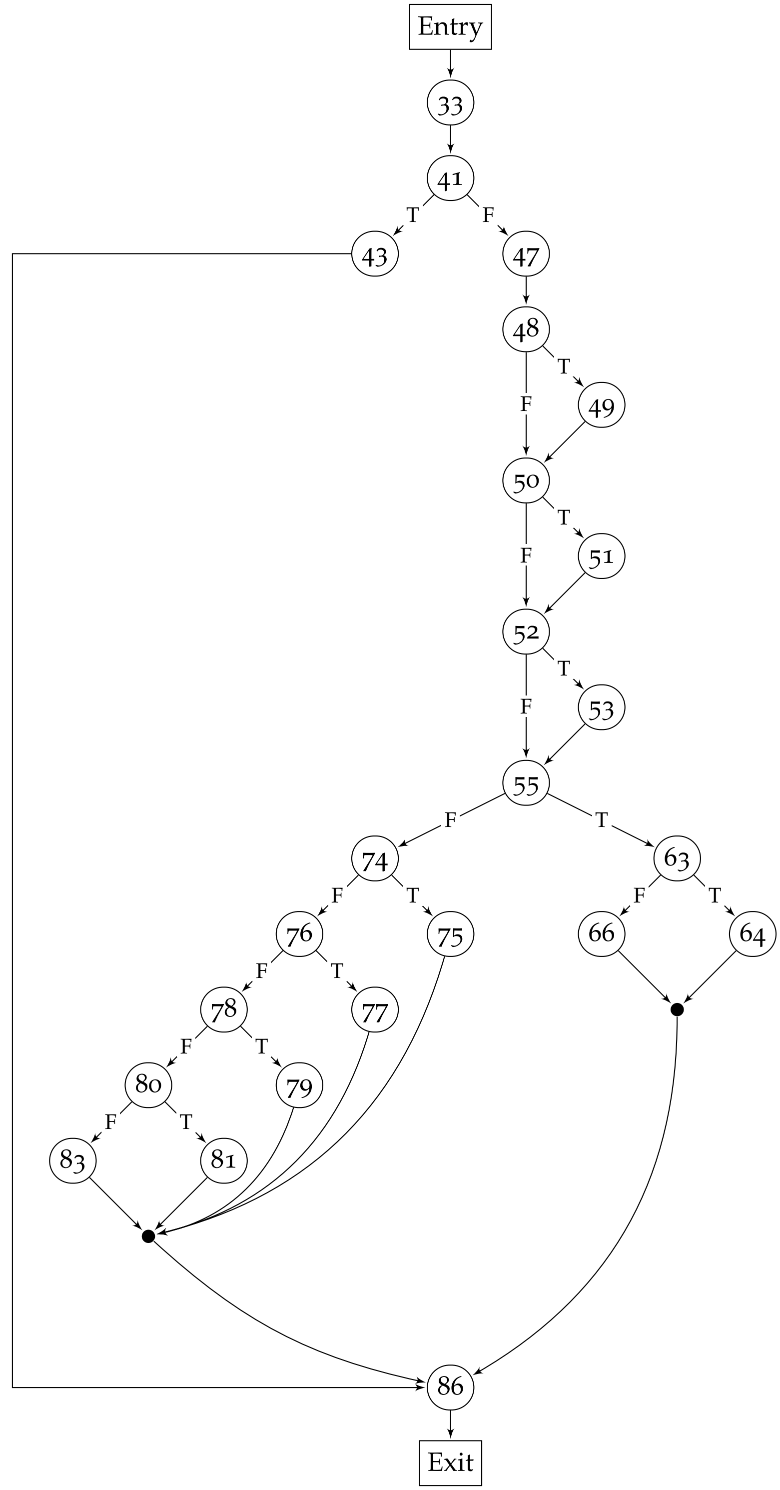

Figure 1: CFG for `tritype.c`