# *Systems III (420-E31-HR)*

# *Lab 09 –White Box Testing – Code Inspections and Code Coverage*

Date assigned: Wednesday, November 1, 2017

Date due: Wednesday, November 1, 2017, 12:00pm

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

* Identify the different types of technical reviews
* Perform a code inspection

To do:

Save this document as a Word document named **YourUserName\_E31\_L09\_WhiteBox.docx**. The document will hold your answers for your lab.

**Part A – Code Coverage Difference Questions [4 marks]**

1. What is the main drawback to statement coverage to the alternative methods? **[1 mark]**

Statement coverage will only cover true conditions, and only each line once.

1. What is the main difference between decision coverage and statement coverage? **[1 mark]**

Only checks logical flow at control structures throughout the code. Tests different conditions at an if statement, at a while loop, etc. Big difference from statement coverage is that it checks false statements as well as true. So it can have tests for a loop that it never enters.

1. What is the main difference between condition coverage and decision coverage? **[1 mark]**

Like decision coverage, but covers all possible combinations, alternate flows, etc. In a “a || b” situation, it would cover:

true || true, true || false, false || true, false || false

1. What is the main difference between path coverage and condition coverage? **[1 mark]**

This covers each possible path through a program. So it will check what happens after going through every possible if statement / while loop, instead of checking IF it enters that if statement or while loop. It checks the paths, not the conditions.

**Part B – Code Coverage (12 marks)**

public int function(int a, boolean b1, boolean b2, boolean b3, boolean b4){

if (b1 || b2)

a = a + 1;

else if (b3)

a = a + 2;

if(b4)

a = a + 5;

else if (b3 || b2)

a = a + 6;

if (b3 || !b4)

a = 0;

else

a = a+1;

return a;

}

1. How many test cases are required for *statement coverage*? Include a table with each test case, what values are used and why you included it’s needed. **(4 marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Int a | Boolean b1 | Boolean b2 | Boolean b3 | Boolean b4 | Expected output | Why this test |
| 0 | True | True | True | True | A = 0 | This test covers entering the if condition on all 3 statements |
| 0 | False | False | True | False | A = 0 | This test covers entering the else if condition on both statements |

The else on the third if statement doesn’t need to be tested here because that’s a false case and statement coverage won’t go into that.

1. How many test cases are required for *decision coverage*? Include a table with each test case, what values are used and why you included it’s needed. **(4 marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Int a | Boolean b1 | Boolean b2 | Boolean b3 | Boolean b4 | Expected output | Why this test |
| 0 | True | True | True | True | A = 0 | This test covers entering the if condition on all 3 statements |
| 0 | False | False | True | False | A = 0 | This test covers entering the else if condition on both statements |
| 0 | False | False | False | True | A = 6 | This test covers entering skipping over the first if/else if entirely and entering the else on the last if. |
| 0 | False | False | False | False | A = 0 | Covers skipping the first two if/else if statements and entering the if block on the last one. |

1. How many test cases are required for *condition coverage*? Include a table with each test case, what values are used and why you included it’s needed. **(4 marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Int a | Boolean b1 | Boolean b2 | Boolean b3 | Boolean b4 | Expected output | Why this test |
| 0 | False | False | False | False | A = 0 | These 16 test cases cover all possible combinations of true and false for the 4 parameters to the method. They therefore also cover all possible conditions that can be met entering each if / else if / else statement. |
| 0 | False | False | False | True | A = 6 |
| 0 | False | False | True | False | A = 0 |
| 0 | False | False | True | True | A = 0 |
| 0 | False | True | False | False | A = 0 |
| 0 | False | True | False | True | A = 7 |
| 0 | False | True | True | False | A = 0 |
| 0 | False | True | True | True | A = 0 |
| 0 | True | False | False | False | A = 0 |
| 0 | True | False | False | True | A = 1 |
| 0 | True | False | True | False | A = 0 |
| 0 | True | False | True | True | A = 0 |
| 0 | True | True | False | False | A = 0 |
| 0 | True | True | False | True | A = 8 |
| 0 | True | True | True | False | A = 0 |
| 0 | True | True | True | True | A = 0 |

**Part E – Path Coverage (16 marks)**

1. Refer to the following sample of code to answer the questions related to code coverage.

int f1(int x, int y) {

while (x != y) {

if (x>y)

x=x-y;

else

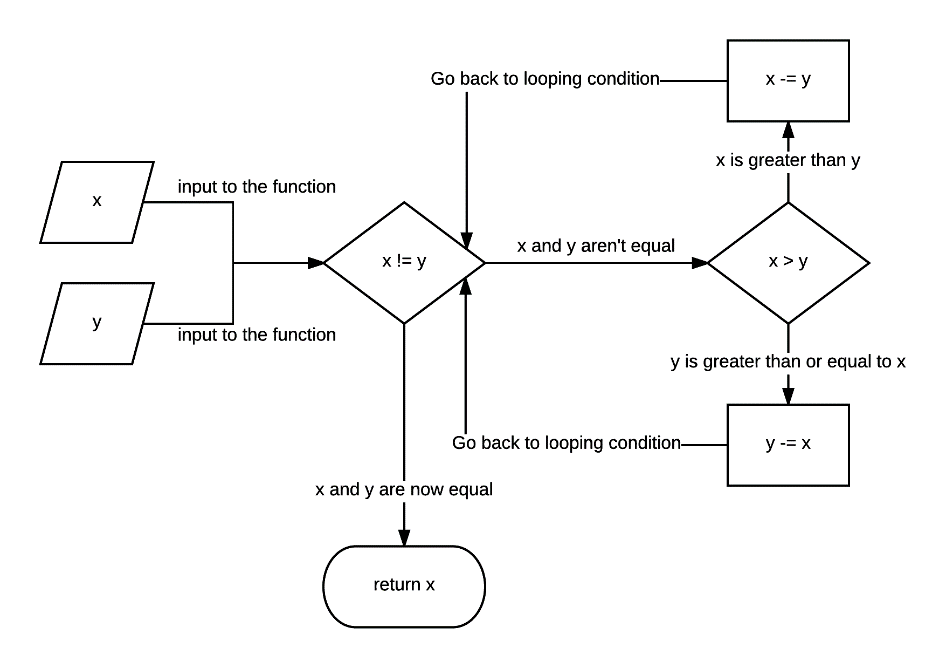
y=y-x;

}

return x;

}

* 1. Draw a flow graph for the code, using the Flowchart, Data Flow Diagram in Microsoft Visio, Microsoft Word, or another tool of your choice. Copy and paste the diagram into this document. **(5 marks)**

****

* 1. List the paths to be tested **(4 marks)**
     1. **Never enter loop (x == y) – return x**
     2. **Enter loop**
        1. **X > Y**
           1. **Exit loop**

**Return x**

* + 1. **Enter loop**
       1. **Y > X** 
          1. **Exit loop**

**Return x**

* + 1. **Enter loop**
       1. **X > Y**
          1. **Y > X**

**X > Y**

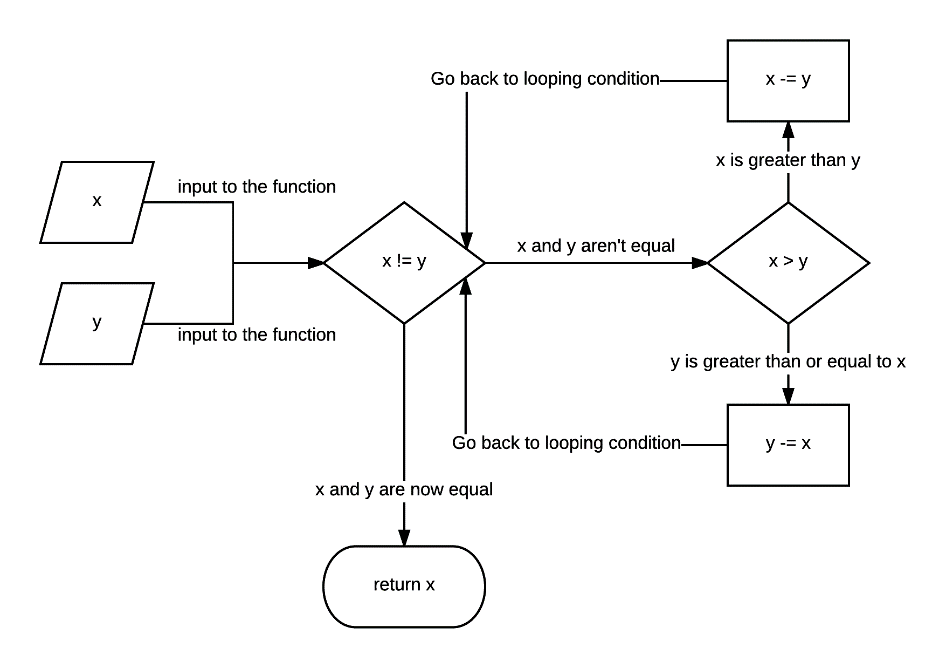
**Exit loop**

**Return X**

* 1. For each path, list the test data. **(4marks)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Int x | Int y | What happens at loop | What happens at if | Return |
| 5 | 5 | 5 == 5 – Do not enter loop | N/A | 5 |
| 2 | 1 | 1. 2 != 1 – Enter loop 2. 1 == 1 – Exit loop | 1. X > Y : 2(x) = 2(x) – 1(y) | 1 |
| 1 | 2 | 1. 1 != 2 – Enter loop 2. 1 == 1 – Exit loop | 1. X < Y : 2(y) = 2(y) – 1(x) | 1 |
| 3 | 5 | 1. 3 != 5 – Enter loop 2. 3 != 2 – Continue loop 3. 1 != 2 – Continue loop 4. 1 == 1 – Exit loop | 1. X < Y : 5(y) = 5(y) – 1(x) 2. X > Y : 3(x) = 3(x) – 2(y) 3. X < Y : 2(y) = 2(y) – 1(x) | 1 |

* 1. Calculate the cyclomatic complexity of your graph, showing the derivation of your answer. **(3 marks)**

****

Since inputs into the function don’t count as edges in the graph, input x and y don’t count. They also don’t count towards nodes in the graph.

E = 6

N = 5

P = 1

CCC = E – N + 2P

= 6 – 5 + 2(1)

= 1 + 2

= 3

The Cyclomatic complexity of this function is 3

**To submit**

When you have completed the assignment:

* Upload **YourUserName\_E31\_L09\_WhiteBox.docx** to Moodle.