CS 3250: Software Testing (Fall 2026)

POTD 4: Structural Graph for source code – LCM — Solution

Section 1 – Due 10-October-2026, 1pm	Section 2 – Due 14-October-2026, 1pm

Purpose: Create graph representation for source code; apply structural graph coverage to source code; get ready for assignment 3, quiz 3, and the final exam.

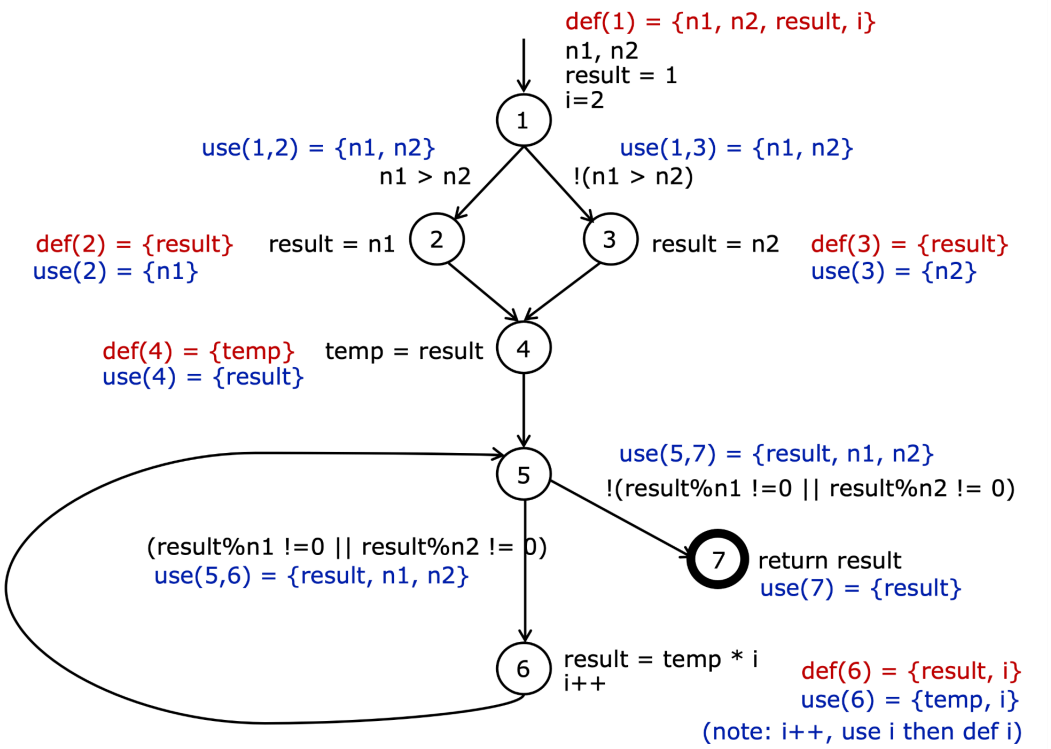
You may make a copy of a worksheet and complete this activity, or type your answers in any text editor. You may work alone or with another student in this course (max team size = 2).

Consider the following Java method

```
// Calculate_LCM takes two numbers and return LCM of the two numbers
// Examples:  LCM of 12 and 15 = 2 * 2 * 3 * 5 = 60
//            LCM of 12 and 18 = 2 * 2 * 3 * 3 = 36
//            LCM of 15 and 18 = 2 * 3 * 3 * 5 = 90
public static int Calculate_LCM(int n1, int n2) {
    int result = 1;
    int i = 2;
    if (n1 > n2)
        result = n1;
    else
        result = n2;

    int temp = result;
    while (result % n1 != 0 || result % n2 != 0) {
        result = temp * i;
        i++;
    }
    return result;
}
```

1. Draw a **Control Flow Graph** for the `Calculate_LCM` method. Annotate the graph with **all** information (code on nodes, code on edges). – Make sure that your graph is ***valid***



1. Apply **Node Coverage** (NC) to design tests

Test requirements	Test paths	Test cases (input values and expected output)
{1, 2, 3, 4, 5, 6, 7}	{ [1,2,4,5,7], [1,3,4,5,6,5,7] }	For test path [1,2,4,5,7] Input: n1=4, n2=2 Expect: 4 For test path [1,3,4,5,6,5,7] Input: n1=10, n2=15 Expect: 30

2. Apply **Edge Coverage** (EC) to design tests

Test requirements	Test paths	Test cases (input values and expected output)
{ (1,2), (1,3), (2,4), (3,4), (4,5), (5,6), (5,7), (6,5) }	{ [1,2,4,5,7], [1,3,4,5,6,5,7] }	For test path [1,2,4,5,7] Input: n1=4, n2=2 Expect: 4 For test path [1,3,4,5,6,5,7] Input: n1=10, n2=15 Expect: 30

3. Apply **Edge-Pair Coverage** (EPC) to design tests

Test requirements	Test paths	Test cases (input values and expected output)
{ (1,2,4), (1,3,4), (2,4,5), (3,4,5), (4,5,6), (4,5,7), (5,6,5), (6,5,6), (6,5,7) }	{ [1,2,4,5,7], [1,3,4,5,7], [1,3,4,5,6,5,6,5,7] } [1,2,4,5,7] directly tours (1,2,4), (2,4,5), (4,5,7) [1,3,4,5,7] directly tours (1,3,4), (3,4,5), (4,5,7) [1,3,4,5,6,5,6,5,7] directly tour (1,3,4), (3,4,5), (4,5,6), (5,6,5), (6,5,6), (6,5,7) -- also tours (3,4,5), (4,5,6), (5,6,5), 6,5,7) with sidetrip	For test path [1,2,4,5,7] Input: n1=4, n2=2 Expect: 4 For test path [1,3,4,5,7] Input: n1=2, n2=4 Expect: 4 For test path [1,3,4,5,6,5,6,5,7] Input: n1=3, n2=14 Expect: 42

4. Derive prime paths

[1] [2] [3] [4] [5] [6] ~~[7]~~**!**	[1,2] [1,3] [2,4] [3,4] [4,5] [5,6] ~~[5,7]~~**!** [6,5]	[1,2,4] [1,3,4] [2,4,5] [3,4,5] ~~[4,5,6]~~**!** **[5,6,5]*** **[6,5,6]*** **[6,5,7]!**	[1,2,4,5] [1,3,4,5] ~~[2,4,5,6]~~**!** ~~[2,4,5,7]~~**!** ~~[3,4,5,6]~~**!** ~~[3,4,5,7]~~**!**	**[1,2,4,5,6]!** **[1,2,4,5,7]!** **[1,3,4,5,6]!** **[1,3,4,5,7]!**

Expanding [4,5,6], [2,4,5,6], [3,4,5,6], [1,2,4,5,6] will result in internal loop (i.e., no longer simple paths)
Reminder: prime path is a simple path that is not a subpath of any other paths.

5. Apply **Prime Path Coverage** (PPC) to design tests

Test requirements	Test paths	Test cases (input values and expected output)
{ (1,2,4,5,7), (1,2,4,5,6), (1,3,4,5,6), (1,3,4,5,7), (5,6,5), (6,5,6), (6,5,7) }	{ [1,2,4,5,7], [1,2,4,5,6,5,7] [1,3,4,5,7], [1,3,4,5,6,5,6,5,7] } [1,2,4,5,7] tours (1,2,4,5,7) directly [1,2,4,5,6,5,7] tours (1,2,4,5,6), (5,6,5), (6,5,7) directly -- also tours (1,2,4,5,7) with sidetrip [1,3,4,5,7] tours (1,3,4,5,7) directly [1,3,4,5,6,5,6,5,7] tours (1,3,4,5,6), (5,6,5), (6,5,6), (6,5,7) directly -- also tours (1,3,4,5,6), (5,6,5), (6,5,6) with sidetrip	For test path [1,2,4,5,7] Input: n1=4, n2=2 Expect: 4 For test path [1,2,4,5,6,5,7] Input: n1=15, n2=10 Expect: 30 For test path [1,3,4,5,7] Input: n1=2, n2=4 Expect: 4 For test path [1,3,4,5,6,5,6,5,7] Input: n1=3, n2=14 Expect: 42

Grading rubric

[Total: 10 points]: Done (or provide evidence of your attempt, full or reasonable effort)

- (5 points) — Providing evidence of your attempt, minimal effort

Section 1:

(-2.5 points) for 24 hours late (submitted after 10-Oct-2026 1pm, by 11-Oct-2026 1pm)
(-5 points) for 48 hours late (submitted after 11-Oct-2026 1pm, by 12-Oct-2026 1pm)

Section 2:

(-2.5 points) for 24 hours late (submitted after 14-Oct-2026 1pm, by 15-Oct-2026 1pm)
(-5 points) for 48 hours late (submitted after 15-Oct-2026 1pm, by 16-Oct-2026 1pm)

Submission

- **Include all team member names and computingIDs in the report.**
- You may do one of the following:
  - Write on the paper and take picture(s) of your POTD, or
  - Type and save your POTD as a .pdf file — *No Word document.*
- Upload your report to ***POTD 4 on Gradescope***.
- Connect your partner to your group on Gradescope so that everyone receives credit
- Each team submits only ***one*** copy

Making your submission available to instructor and course staff is ***your*** responsibility; if we cannot access or open your file, you will not get credit. Be sure to test access to your file before the due date.

Copyright © 2026 Upsorn Praphamontripong

Released under the CC-BY-NC-SA 4.0 license.
Last updated 2026-10-05 19:28