CSC364/CSCM64 Lab 5

To be solved in groups of two or three.
Last day for lab sign-off: 4th April 2022

Consider the following specification and program:

Require: x and y are non-negative integers.
Ensure: GCD(x,y) returns the greatest common divisor of x and y.

```
1: function GCD(int x, int y)
2:     int temp;
3:     while (y != 0) do
4:         temp = y;
5:         y = x mod y;
6:         x = temp;
7:     end while
8:     return x;
9: end function
```

Task 1. Draw the program graph of the program GCD.

Task 2.

1. Give a minimal test suite for C_0 coverage. Give a brief argument why your test suite covers and why it is minimal.

2. Give a minimal test suite for C_1 coverage. Give a brief argument why your test suite covers and why it is minimal.

3. Give a minimal test suite for modified Decision/Condition coverage. Give a brief argument why your test suite covers and why it is minimal.

4. Give a minimal test suite for $C_i(3)$ coverage. Give a brief argument why your test suite covers and why it is minimal.

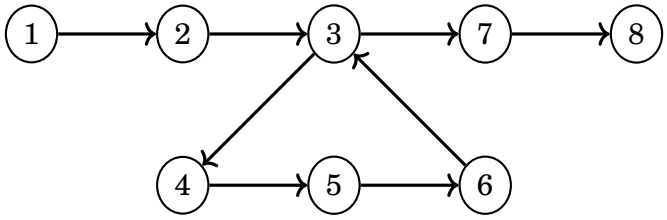
Task 3.

1. Give a table that lists the define/use nodes for the variables x,y, and temp.

2. Give a minimal test suite for the All Uses criterion. Give a brief argument why your test suite covers and why it is minimal.

Answers

Answer 1. We have not given a very strict definition of the notion of "program graph", so there are a number of sensible interpretations. Here is one possibility:



Answer 2. The following is a minimal test suite for questions 1 – 3:

Case	x	y	Expected Output
1	1	1	1

The execution path on input $(1,1)$ is equal to $\langle 1,2,3,4,5,6,3,7,8\rangle$. The test suite is minimal in each case, because it has only one test case.

1. C_0 coverage is provided since the execution path contains every node of the graph.
2. C_1 coverage is provided since the execution path contains every edge of the graph.
3. MC/DC is provided because the decision $y \neq 0$ takes both possible outcomes at least once during execution of our single test case.

It remains to give a minimal test suite for $C_i(3)$ coverage. The following test suite will work:

Case	x	y	Expected Output
1	1	0	1
2	1	1	1
3	4	6	2
4	5	3	1

One easily verifies that the k^{th} test case leads to $k-1$ executions of the loop. It is obvious that 4 test cases are necessary.

Answer 3. 1. Here is the table:

Variable	Defined at Node	Used at Node
x	1, 6	5, 8
y	1, 5	3, 4, 5
temp	4	6

2. We need to cover feasible definition-clear paths from all defining nodes of all variables to all used nodes and all successors of use nodes of those variables, provided such paths exist.

   The following table helps keep track of this data:

Variable	Define Node	Target Node	Feasible DC-path
x	1	5	$\langle 1,2,3,4,5\rangle$
x	1	6	$\langle 1,2,3,4,5,6\rangle$
x	1	8	$\langle 1,2,3,7,8\rangle$
x	6	5	$\langle 6,3,4,5\rangle$
x	6	6	$\langle 6\rangle$
x	6	8	$\langle 6,3,7,8\rangle$
y	1	3	$\langle 1,2,3\rangle$
y	1	4	$\langle 1,2,3,4\rangle$
y	1	7	$\langle 1,2,3,7\rangle$
y	1	5	$\langle 1,2,3,4,5\rangle$
y	1	6	No DC-Path
y	5	3	$\langle 5,6,3\rangle$
y	5	4	$\langle 5,6,3,4\rangle$
y	5	7	$\langle 5,6,3,7\rangle$
y	5	5	$\langle 5\rangle$
y	5	6	$\langle 5,6\rangle$
temp	4	6	$\langle 4,5,6\rangle$
temp	4	3	$\langle 4,5,6,3\rangle$

   We extract from the paths in this table the maximal elements with respect to the subpath-ordering:

   $$\langle 1,2,3,4,5,6\rangle, \langle 1,2,3,7,8\rangle, \langle 6,3,4,5\rangle, \langle 6,3,7,8\rangle, \langle 5,6,3,4\rangle, \langle 5,6,3,7\rangle, \langle 4,5,6,3\rangle.$$

   A minimal set of execution paths that contains these maximal elements as subpaths is given as follows:

   $$\langle 1,2,3,4,5,6,3,4,5,6,3,7,8\rangle, \langle 1,2,3,7,8\rangle.$$

   The first path contains almost all of the maximal paths as subpaths, except the second. It is obvious that there exists no execution path that contains the two paths as a subpath, so that this set of paths is minimal.

   A test suite covering the two paths is given as follows:

Case	x	y	Expected Output
1	4	6	2
2	1	0	1