Week 1

1. Answer:
   a. No.
   b. Full node coverage does not necessarily imply edge coverage. In this particular example it does. So the answer is “no”.
   c. Yes, e.g. \{ [1,2,3,2,4] \}
   d. 2, e.g. \{ [1,2,3,2,2,4], [1,2,4] \}

2. Answer:
   a. No they are not. TR2 also requires all edges to be covered, which is stronger than TR1.
   b. 2
   c. 6
   d. 15, namely:
      \[
      [1,2,4,6,1]\ [1,2,4,5,6,1]
      [2,3,2]\ [2,4,6,1,2]\ [2,4,5,6,1,2]
      [3,2,3]\ [3,2,4,6,1,7]\ [3,2,4,5,6,1,7]
      [4,6,1,2,4]\ [4,5,6,1,2,4]\ [4,6,1,2,3]\ [4,5,6,1,2,3]
      [5,6,1,2,4,5]
      [6,1,2,4,6]\ [6,1,2,4,5,6]
      \]

3. Answer: I will only mention the strongest touring. Note that direct touring implies the other two, and side trip implies detour.
   a. Direct tour
   b. Detour
   c. Side trip
Week2

1. Answers:
   a. No. It misses for example the “greater than one” block of q1.
   b. Yes.
   c. One possibility is given below. Black were the already given test cases. Reds are the new ones.

   \[
   (2,2,2) \quad (2,1,-1) \quad (2,0,0) \quad (2,-1,1) \\
   (1,2,1) \quad (1,1,2) \quad (1,0,-1) \quad (1,-1,0) \\
   (0,2,0) \quad (0,1,1) \quad (0,0,2) \quad (0,-1,-1) \\
   (-1,2,-1) \quad (-1,1,0) \quad (-1,0,1) \quad (-1,-1,2)
   \]

   d. Above there are 16 tests cases, providing full PWC. Since the theoretical lower found is \(N^2\) where \(N\) is the number of blocks per characteristic, assuming that this number is the same, so this 16 must also be the minimum test set size for this particular case.

2. Answers:
   a. Yes. MBCC subsumes ECC.
   b. No. Any combination that is fully over non-base blocks is not requires.
   c. No.
   d. We need 18 test cases, if duplicates are not removed:

   Each base test induces these test requirements (combinations to cover):

   - 2 combinations over characteristic q1 (per base test)
   - 3 combinations over characteristic q2 (per base test)
   - 3 combinations over characteristic q1 (per base test)

   There are two base tests, so together they induce 16 combinations to cover; plus the base tests themselves. Total: 18.

   However, the combinations over q1 would be duplicates. Removing this, you end up with 16 unique combinations to cover, and therefore you would need 16 test cases to cover them all.
Week 3

1. a. Yes. Consider a program with a CGG in the form of a linear list [1,2,3,4], where 1 and 3 define x, and 2 and 4 use x. The du-paths (over x) are [1,2] and [3,4]. None is a prime path. As a side note: a du-path is required to be a simple path; it is not required to be a prime path.

b. No.

c. In principle only in n1. This may also happen in n3, but it is then assumed to happen after the (last) use of the variable there so that the path in between defin-clear is.

2. a. 5, listed below. I organize that in “def-pair” sets:

\[
du(0,7,x) = \text{just 1x } \rightarrow \ [0, 1, 7] \ (dp1) \\
du(0,5,x) = \text{also 1x } \rightarrow \ [0, 1, 2, 4, 5] \ (dp2) \\
du(3,5,x) = \text{just 1x } \rightarrow \ [3, 2, 4, 5] \ (dp3) \\
du(3,7,x) = \text{we have 2x } \rightarrow \ [3, 2, 4, 6, 1, 7] \ (dp4) \text{ and } [3, 2, 4, 5, 6, 1, 7] \ (dp5)
\]

b. Just 2. As the TR you can use one of the du-paths about that start at the def of x in 0, and one more that starts in 3.

c. 4. You need to include one path from each def-pair above in your TR.

3. a. 7,2,2 for x,y,a respectively. They are listed below.

\[
du(0,1,x) = [0,1] \\
du(0,4,x) = [0,1,2,4] \\
du(0,7,x) = [0,1,7] \\
du(3,4,x) = [3,2,4] \\
du(6,1,x) = [6,1] \\
du(6,7,x) = [6,1,7] \\
du(6,4,x) = [6,1,2,4] \\
du(0,2,y) = [0,1,2] \\
du(3,2,y) = [3,2] \\
du(4,6,a) = [4,6] \\
du(5,6,a) = [5,6]
\]

b. Obviously 2.

c. As many as the number of def-pairs for y. So: 2.

d. 7
Week 4

1. 4 (for call point 8)
   a. 4 (for call point 8)
      for linecnt: [6,8,c0,c1,c3,c4,c5]
      for lastdelimit: [7,2,6,8,c0,c1]
      for curword: [7,2,6,8,c0,c1,c3,c4] [c7,c8,8,1,2,3,4,7]

   b. 4 (for call point 5)
      for linecnt: [0,1,2,3,4,5, c0,c1,c3,c4,c5]

      **Note:** some of us might argue that [6,8,1,2,3,4,5,c0,c1,c3,c4,c5] is also a coupling du-path for linecnt. Note that this path goes through a program call (in node-8). The argument for counting this path as a coupling du-path for linecnt is that the call in node-8 does not re-define linecnt, so the path is still def-clear with respect to linecnt, so the influence of the def of linecnt at node-0 would still directly affect the call in node-5.
      The argument against counting the path as a coupling du-path for linecnt is that the path is that the actual control flow in node-8 first goes into the called program, and then back to node-8, and therefore the full path from node-6 ... node-8 ... and finally the call site at node-5 is not even a simple path.
      Let's however stick with the strict interpretation of the definition of du-path and therefore reject the above as a coupling du-path.

      for lastdelimit: [7,2,3,4,5,c0,c1]
      for curword: : [7,2,3,4,5, c0,c1,c3,c4] [c7,c8,5,2,3,4,7]

c. 1 for each call site; well there is only one coupling du-path for lastdelimit for each of the call sites in question.

d. 2 for each call site. Both start from different def-locations anyway. So btw, for full ACD coverage you would need to cover these same paths as well.