1. CS5391 Survey of Software Engineering
   \{ from Dr. Chen \}
   Explain the following software process models and point out their pros and cons:
   (a) Formal method development model
   (b) Agile development model

2. CS5392 Formal Methods in Software Engineering
   \{ from Dr. Yang \}
   1. Construct the truth table for the formula: \((\neg p \rightarrow q) \rightarrow \neg r\).
   2. Prove the validity of \(q \rightarrow (p \rightarrow r), \neg r, q \vdash \neg p\).
3. Conduct symbolic evaluation of path 1,2,3,5,6,8 and path 1,2,4,5,7,8 for the code fragment below. For each path, explain if this path is feasible. Provide the intermediate and final results for path condition, domain and computation.

1) input x,z;
2) if x > z then
3) y := 3;
   else
4) x := z + 2;
   end if;
5) if x > z + 1 then
6) w := 3;
   else
7) w := 2
   end if;
8) write w;

3. CS5393 Software Quality
   {from Dr. Yang}
   1. Answer questions a-g for the graph defined by the following sets:

   - \( N = \{1, 2, 3, 4\} \)
   - \( N_0 = \{1\} \)
   - \( N_f = \{4\} \)
   - \( E = \{(1, 2), (2, 3), (3, 2), (2, 4)\} \)

   Also consider the following (candidate) test paths:

   - \( t_0 = [1, 2, 4] \)
   - \( t_1 = [2, 3, 2, 4] \)
   - \( t_2 = [1, 2, 3, 2] \)
   - \( t_3 = [1, 2, 3, 4] \)
   - \( t_4 = [1, 2, 3, 2, 4] \)

   (a) Draw the graph.
   (b) Which of the 5 listed candidate paths are test paths? Explain the problem with any path that is not a test path.
   (c) List the test requirements for Node Coverage on the graph.
   (d) List the test requirements for Edge Coverage on the graph.
   (e) List the test requirements for Edge Pair Coverage on the graph.
   (f) List the test requirements Prime Path Coverage on the graph.
   (g) List test paths that achieve Edge Coverage but not Prime Path Coverage on the graph.

2. Consider the method \( \text{Min} \) and its 6 mutants in Figure 1. Provide reachability conditions, infection conditions, propagation conditions, and test case values to kill mutants \( \Delta 1, \Delta 2, \Delta 5 \).
<table>
<thead>
<tr>
<th>Original Method</th>
<th>With Embedded Mutants</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int Min (int A, int B)</code>&lt;br&gt;  {&lt;br&gt;     <code>int minVal;</code>&lt;br&gt;     <code>minVal = A;</code>&lt;br&gt;     <code>if (B &lt; A)</code>&lt;br&gt;     <code>{</code>&lt;br&gt;     <code>    minVal = B;</code>&lt;br&gt;     <code>}</code>&lt;br&gt;     <code>return (minVal);</code>&lt;br&gt;     <code>} // end Min</code></td>
<td><code>int Min (int A, int B)</code>&lt;br&gt;  {&lt;br&gt;     <code>int minVal;</code>&lt;br&gt;     <code>minVal = A;</code>&lt;br&gt;     <code>if (B &lt; A)</code>&lt;br&gt;     <code>{</code>&lt;br&gt;     <code>    minVal = B;</code>&lt;br&gt;     <code>}</code>&lt;br&gt;     <code>Δ1 minVal = B;</code>&lt;br&gt;     <code>if (B &lt; A)</code>&lt;br&gt;     <code>{</code>&lt;br&gt;     <code>    minVal = B;</code>&lt;br&gt;     <code>}</code>&lt;br&gt;     <code>Δ2 minVal = A;</code>&lt;br&gt;     <code>if (B &gt; A)</code>&lt;br&gt;     <code>{</code>&lt;br&gt;     <code>    minVal = B;</code>&lt;br&gt;     <code>}</code>&lt;br&gt;     <code>Δ3 minVal = B;</code>&lt;br&gt;     <code>failOnZero (B);</code>&lt;br&gt;     <code>} // end Min</code></td>
</tr>
</tbody>
</table>