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. Prove the validity of \( p \rightarrow q \land r \vdash (p \rightarrow q) \land (p \rightarrow r) \).
2. Conduct symbolic evaluation of path 1,2,4,5,7,8 for the code fragment below. 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. Find appropriate predicates and their specification to translate the following into predicate logic:
(a) Every child is younger than its father.
(b) No animal is both a bird and a dog.
(c) Not all birds can fly.

3. CS5393 Software Quality
   { from Dr. Yang }
1. Recall a du-path with respect to a variable v is a simple path that is def-clear with respect to v from node ni for which v is in def(ni) to a node nj for which v is in use(nj).

Consider the following code fragment and answer the questions that follow using the node numbers given as comments:

```plaintext
y = m;                       // node 1
if(x>0){                     // node 2
    w++;                     // node 3
}
else{                         // node 4
    w = 2*w;                 // node 5
}
while(y<10){                 // node 6
    x++;                     // node 7
    y++;                     // node 8
}                            // node 9
z = w + x;                   // node 10
```
node 8 is a final node.

(b) Which nodes have defs for variable $w$? Which nodes have uses for variable $w$?

(c) Give a set of paths that would be needed to achieve All-du-paths Coverage with respect to variable $y$. Show the def-use relations and indicate which paths cover which relationships.

2. Consider the method $Min$ and its 6 mutants in Figure 1. For each of mutants $\Delta 1$, $\Delta 3$, $\Delta 5$, provide reachability condition, infection condition, propagation condition, test case values to strongly kill it, and test case values to weakly kill it.

<table>
<thead>
<tr>
<th>Original Method</th>
<th>With Embedded Mutants</th>
</tr>
</thead>
<tbody>
<tr>
<td>int Min (int A, int B) { int minVal; minVal = A; if (B &lt; A) { minVal = B; } return (minVal); } // end Min</td>
<td>int Min (int A, int B) { int minVal; minVal = A; } $\Delta 1$ minVal = B; if (B &lt; A) $\Delta 2$ if (B &gt; A) $\Delta 3$ if (B &lt; minVal) { minVal = B; $\Delta 4$ Bomb(); $\Delta 5$ minVal = A; $\Delta 6$ minVal = failOnZero (B); } return (minVal); } // end Min</td>
</tr>
</tbody>
</table>

Figure 1: Mutants