Problem 1
Let \( \text{DOUBLE-SAT} = \{ \langle \phi \rangle \mid \phi \text{ has at least two satisfying assignments} \} \).

Show that \( \text{DOUBLE-SAT} \) is NP-complete.
\( \text{DOUBLE-SAT} \in \text{NP} \), since a nondeterministic Turing machine can decide Double-SAT as follows: On input a Boolean formula \( \Phi(x_1, \ldots, x_n) \), nondeterministically guess 2 assignments and verify whether both satisfy \( \Phi \). To show that \( \text{DOUBLE-SAT} \) is NP-Complete, we give a reduction from SAT to DOUBLE-SAT, as follows: On input \( \Phi(x_1, \ldots, x_n) \):

i. Introduce a new variable \( y \).

ii. Output formula \( \Phi'(x_1, \ldots, x_n, y) = \Phi(x_1, \ldots, x_n) \land (y \lor \neg y) \).

If \( \Phi(x_1, \ldots, x_n) \in \text{SAT} \), then \( \Phi \) has at least 1 satisfying assignment, and therefore \( \Phi'(x_1, \ldots, x_n, y) \) has at least 2 satisfying assignments as we can satisfy the new clause \( (y \lor \neg y) \) by assigning either \( y = 1 \) or \( y = 0 \) to the new variable \( y \), so \( \Phi'(x_1, \ldots, x_n, y) \in \text{DOUBLE-SAT} \). On the other hand, if \( \Phi(x_1, \ldots, x_n) \not\in \text{SAT} \), then clearly \( \Phi'(x_1, \ldots, x_n, y) = \Phi(x_1, \ldots, x_n) \land (y \lor \neg y) \) has no satisfying assignment either, so \( \Phi'(x_1, \ldots, x_n, y) \not\in \text{DOUBLE-SAT} \). Therefore, \( \text{SAT} \leq^p \text{DOUBLE-SAT} \), and hence \( \text{DOUBLE-SAT} \) is NP-Complete.

Problem 5
Let \( \text{DOUBLE-CLIQUE} = \{ \langle G, k \rangle \mid G \text{ has two cliques, each of size greater than or equal to } k \} \).

The nondeterministic polynomial time algorithm is to nondeterministically choose two different sets of vertices of size \( k \). For each of the two sets of vertices check that each is fully connected. This takes 1 nondeterministic step using space \( O(k) \) to find the two sets of size \( k \). And it is polynomial in \( k \) to check that they are all connected \( O(k^2) \).

\( \text{CLIQUE} \leq^p \text{DOUBLE-CLIQUE} \)
On input \( \langle G, k \rangle \) output \( \langle G', k \rangle \) defined s follows:
Create a complete graph of size \( k \). Pick one vertex at random in the clique and attach it with one edge to one single node in \( G \). Output the new graph.
Claim this is a polynomial time reduction from CLIQUE to DOUBLE-CLIQUE. The operation takes \( O(k^2) \) time to produce the complete graph and one extra step to add the edge. It takes \( O(k^2 + n^2) \) to write down the new graph \( G' \).

Now consider the case when \( G \) had a clique of size \( k \). Then we know that \( G' \) has two cliques of size \( k \), the original clique and the newly added clique.
Now consider the case when \( G' \) has two (or more) cliques then we know that the original \( G \) has a at least one clique. The added clique only adds at most one clique to the graph, (since we only attach it at one edge, it adds only one clique). So if \( G' \) has two cliques then one clique was in the original \( G \).