1) Instances, Variants and Substitutions
1. Suppose $\Theta_1$ and $\Theta_2$ are substitutions and there exist substitutions $\sigma_1$ and $\sigma_2$, such that $\Theta_1 = \Theta_2 \sigma_1$ and $\Theta_2 = \Theta_1 \sigma_2$. Show that there exists a variable-pure substitution $\gamma'$, such that $\Theta_1 = \Theta_2 \gamma'$.

From $\Theta_1 = \Theta_2 \sigma_1$ and $\Theta_2 = \Theta_1 \sigma_2$ follows that $\Theta_1 = \Theta_1 \sigma_2 \sigma_1$. Let $\gamma'$ be $\sigma_1 \sigma_2$ with the domain restricted to $\text{var}(\Theta_1)$. Obviously $\gamma'$ is variable pure. From $\rho_1 \rho_2$ variable pure follows $\rho_1$ and $\rho_2$ are also variable pure. Hence, there must be a variable pure restriction $\gamma$ of $\sigma_1$ to the domain $\text{var}(\Theta_2)$.

2. Which of the following clauses are Instances or Variants of each other?

<table>
<thead>
<tr>
<th>Clause Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(x, y, z)</td>
<td>v</td>
<td>-</td>
<td>v</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>p(x, b, f(z))</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>p(v, w, f(z))</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>p(f(x), y, f(z))</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>p(f(x), y, z)</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
</tbody>
</table>
2) The following Lemma shows that we only need to deal with Herbrand interpretations in order to find a model for any logic program:

Let $C$ be a set of clauses and $\Sigma$ be any signature containing all symbols used in $C$. The grounding of $C$ with respect to $\Sigma$, denoted $C^*$ is the set of all ground instances of the signature $\Sigma$ of clauses in $C$. Let $I$ be an Herbrand interpretation and $C$ be a set of clauses.

Prove that $I \models C$ if and only if $I \models C^*$.

a) $I \models C^* \rightarrow I \models C$.
Assume $I \models C^*$. Without limiting generality we assume that for any two clauses $C_1$ and $C_2$ in $C$, the sets of variables used in $C_1$ and $C_2$ are disjoint. Let $\sigma$ be any ground substitution over $\Sigma$. Then $C\sigma \subseteq C^*$. Hence, it is clear that $I \models C\sigma$. As we can choose any $\sigma$, it follows that $I \models C$.

a) $I \models C \rightarrow I \models C^*$.
Assume $I \models C$, but not $I \models C^*$. Without limiting generality we assume that for any two clauses $C_1$ and $C_2$ in $C$, the sets of variables used in $C_1$ and $C_2$ are disjoint. Let $\sigma$ be any ground substitution over $\Sigma$. Then $C\sigma \subseteq C^*$. It is clear that $I \models C\sigma$. As $C^*$ is the set of all ground instances of the signature $\Sigma$ of clauses in $C$, there must be some $\sigma$ for every clause $C_1$ in $C^*$, such that $C_1 = C\sigma$. Then, however, $I \models C_1$, which is in conflict with our assumption.
3) Program Completion

1. Let the definition of a predicate symbol p be

   \[ p(y) :- q(y), \neg r(a,y). \]
   \[ p(f(z)) :- \neg q(z). \]
   \[ p(b). \]

   Give a completion of p.

   1. \[ p(X) :- (X = y), q(y), \neg r(a,y). \]
   \[ p(X) :- (X = f(z)), \neg q(z). \]
   \[ p(X) :- (X = b). \]
   2. \[ p(X) :- \exists y (X = y), q(y), \neg r(a,y). \]
   \[ p(X) :- \exists z (X = f(z)), \neg q(z). \]
   \[ p(X) :- (X = b). \]
   3. \[ p(X) :- \exists y (X = y), q(y), \neg r(a,y) \lor \exists z (X = f(z)), \neg q(z) \lor (X = b). \]
   4. \[ p(X) = \exists y (X = y), q(y), \neg r(a,y) \lor \exists z (X = f(z)), \neg q(z) \lor (X = b). \]

2. Let P be a normal program and comp(P) it's completion. Prove that P is a logical consequence of comp(P). Hint: P is a logical consequence of comp(P) if

   \[ I \models \text{comp}(P) \rightarrow I \models P. \]

   Let I be a model of comp(P). P is a logical consequence of comp(P) if I \models \text{comp}(P) \rightarrow I \models P .

   As I models comp(P), every clause C in comp(P) is true in I. Now we apply the completion backwards:

   Replace every
   \[ A = G_1 \lor \ldots \lor G_n \]
   by
   \[ A \rightarrow G_1 \lor \ldots \lor G_n. \]
   \[ G_1 \lor \ldots \lor G_n \rightarrow A. \]

   Clearly, I is also a model for the resulting program.

   As for all clauses C in a program P I \models P \rightarrow I \models C, in the following we ignore
   \[ A \rightarrow G_1 \lor \ldots \lor G_n. \]

   Then replace every
   \[ G_1 \lor \ldots \lor G_n \rightarrow A. \]
   by
   \[ G_1 \rightarrow A. \]
   \[ \ldots \]
   \[ G_n. \rightarrow A. \]

   Clearly, I is also a model for the resulting program P'.

   Each Gi in comp(P) results from a clause in P. I only contains ground instances of clauses in P. Now let G be the clause in P, Gi results from. As I is a model for P', all equalities and existentials in Gi must hold. We obtain a ground program P'' by removing all equalities and existentials from P'. As P'' is more general than P', obviously I \models P''. As the last step reverses the first two steps of the completion, P'' is a ground instantiation of P, hence I \models P.