8.3a) The grammar has four trees:

In each case, the attributes can be evaluated in the order given by the partition. Note that the order of evaluation within one element of the partition depends upon the production. For example, $d$ is evaluated before $c$ in the production $x::=u$ but $c$ is evaluated before $d$ in the production $x::=v$.

The partition is admissible because $c$ and $d$ are both synthesized attributes of $X$ and $a$ and $b$ are both inherited attributes of $X$: values for $c$ and $d$ are determined in productions with $X$ on the left-hand side; values for $a$ and $b$ are determined in productions with $X$ on the right-hand side.
8.3b) \( IDP(p) = IDS(X) \) for all \( p \) in this example, and \( IDS(X) \) can be determined by inspection from the graphs of part (a):

![Diagram](attachment:image.png)

The cyclicity of \( IDS(X) \) reflects the fact, noted in the answer to part (a), that different productions must use different orders of evaluation within one partition. Every cycle is wholly contained within one element of the partition; there is no cycle across partitions.

8.3c) Attributes are being used as “temporaries” unnecessarily in the given grammar. The following grammar avoids this use, and hence avoids the cycles:

\[
\begin{align*}
\text{RULE Z ::= sX .} \\
\text{RULE X ::= u .} \\
\text{ATTRIBUTION} \\
&X.a \leftarrow X.c; \\
&X.b \leftarrow X.c; \\
\text{RULE Z ::= tX .} \\
\text{RULE X ::= v .} \\
\text{ATTRIBUTION} \\
&X.b \leftarrow X.d; \\
&X.a \leftarrow X.d;
\end{align*}
\]

Here the evaluation of the attributes within an element of the partition given in (a) is arbitrary because there are no dependencies among them.

8.3d) The modification of (c) corresponds to the definition of NDDP. If dependencies exist between attributes of the same class (inherited, synthesized) on the same side of the production they may always be removed by substitution. Intuitively, information enters the algorithm only via inherited attributes of the left-hand side and synthesized attributes of the right-hand side; only values of synthesized attributes of the left-hand side and inherited attributes of the right-hand side are calculated by the algorithm.
8.10) If no syntactic rule of a grammar leads to recursion then the number of sentences in the language (and hence the number of structure trees) is finite. Since there is a finite number of trees, there is a maximum number of attributes to be computed. Because the grammar is partitioned, all of these attributes can be computed. Suppose that we desire to compute the attributes by making a sequence of passes over the structure tree, with the sequence following one of the given strategies. We will be able to evaluate at least one attribute per pass, and hence $k$ passes ($k$ the maximum number of attributes) will suffice.