1) 
   a. A tree-building compiler is sometimes required by source language constructs, such as overloading in Ada, which must be analyzed using a broad context. Also, better code can be generated by a tree-building compiler than by a pass-oriented compiler in many cases.

   b. See pp. 63-64 of the text.

   c. See pp. 69 and 76 of the text.

   d. See Section 4.1.1 of the text.

   e. See pp. 107-108 of the text.

   f. We consider the grammar with embedded connection points to define a new language, by treating the connection points as terminal symbols. The transformations that eliminate left recursion are applied to this augmented grammar. They are guaranteed to leave the language generated by the grammar invariant, and hence the connection points must bear the same relationship to the terminals of the source language that they did before the transformation.

   g. See pp. 187-188 of the text. An intrinsic attribute cannot be classified as either inherited or synthesized, since it if not an element of \( AF(p) \) for any \( p \). Most intrinsic attributes, however, behave as synthesized attributes of leaves.

2) 
   a. The attribute grammar must be partitioned. Absolutely non-circular grammars may require tests to determine the order of evaluation, and these operations are not among the three kinds mentioned. Algorithms implemented using the three operations need not traverse the tree in a regular fashion, however, which means that they do not belong to the alternating class.

   b. If the attribute grammar is partitioned then the lifetimes of all attributes are known from an examination of the grammar, and storage optimization can be performed.

   c. See Section 8.3 of the text.