In answering the following questions, assume that the translations are those given by the published solution to Homework #3. Also assume that the tree-building strategy will be used. The translator is to be written in C, and converts MINILAX to VAX Assembly code.

It is usually best for a tree-building compiler to build a “shaped tree” instead of a structure tree. The shaped tree is more suitable for the defined translation than the structure tree derived from the denotational definition. For example, source-language constant expressions can usually be folded during analysis and replaced in the shaped tree by their values. Also, the nodes of the shaped tree may reflect operator identification and type analysis: Instead of one “expression” node type, there may be several. The criterion for deciding on the set of node types is the information requirements of the translation. If two kinds of expression require different kinds of information for their translation, then they are represented by different kinds of node in the shaped tree.

1) Identify two classes of expression rules (“E ::= · · ·”) in the MINILAX abstract syntax that require different kinds of information to support the translations given in the solution to Homework #3. List all of the expression rules that belong to each class.

2) For each of the two classes that you identified in (1), state all of the information that will be required and show how this information determines the translation.

3) The nodes you identified in (1) must be constructed on the basis of left context information available in the definition table and the phrase stack. (Note that not all of the information need be provided from these sources, but the compiler must be able to decide on the type of the node and its linkages.) What information must be held in these two data structures to make construction of these nodes possible?

4) Your answer to (3) may provide the source of some of the information you listed in (2). Explain how each of the items not covered by that answer is obtained, and how it reaches the node at which it is used in the translation.

5) Space requirements for a tree-building compiler can be reduced by coding and discarding the trees for particular phrases when those phrases have been completed. In effect, the pass-oriented and tree-building strategies are combined. Can this approach be taken with MINILAX, given the postulated translation? If so, which phrases would you code when their trees had been completely built? If not, briefly explain why not.
The purpose of this assignment is to establish the information requirements for the analysis module of the compiler. It is worth 10 points, and is due at the beginning of the lecture on February 11.