1. (6 points). Consider the following BNF grammar:

\[ \langle \text{lstring} \rangle ::= \langle s \rangle \langle s \rangle \]
\[ \langle s \rangle ::= a \mid b \mid c \mid a\langle s \rangle \mid b\langle s \rangle \mid c\langle s \rangle \]

Add appropriate attributes and conditions to the grammar so that only \( \langle \text{lstring} \rangle \)s that satisfy the following condition are allowed: every occurrence of ‘a’ is immediately preceded by a ‘b’ and followed by a ‘c’. In other words, each occurrence of ‘a’ is sandwiched between b and c. (Thus, for example, ‘bacbbebabc’ is legal, but not ‘babbebc’.) Use synthesized attributes or inherited attributes, or a combination. Do not change the BNF grammar; do not compute the whole string and pass it up to the root – use only simple arithmetic functions in your attribute evaluation rules and conditions. If the problem cannot be solved under these constraints, explain why not.

2. (6 points). Consider the following BNF grammar of expressions:

\[ \langle \text{exp} \rangle ::= \langle \text{simple} \rangle \mid \langle \text{exp} \rangle + \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \ast \langle \text{exp} \rangle \]
\[ \langle \text{simple} \rangle ::= \langle \text{number} \rangle \mid \langle \text{variable} \rangle \]

where \( \langle \text{number} \rangle \) and \( \langle \text{variable} \rangle \) correspond to numbers and (program) variables.

This grammar does not impose the proper precedence between + and *. Without changing the productions, introduce appropriate attributes, evaluation rules, and conditions such that the proper precedence is enforced (i.e., parse trees that do not have the proper precedence, although legal according to the BNF productions, are ruled out by having one or more of the conditions evaluating to false).

3. (8 points). Suppose you have a programming language that allows for two types of declarations, variable declarations and procedure declarations. Further, all variable declarations in a declaration sequence must precede all procedure declarations. And if \( P \) and \( Q \) are two procedures declared in a declaration sequence, and the declaration of \( P \) precedes that of \( Q \), then you are allowed to call \( P \) from inside the body of \( Q \) but not vice-versa.

Write down an attribute grammar that captures these requirements. But don’t get carried away with the details of your solution. As long as the essential ideas are clear, you will get full credit.