

## CSE 755, Assignment #1

Due: Jan. 17, '12.

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

$$\begin{aligned}\langle lstring \rangle &::= \langle s \rangle \langle s \rangle \\ \langle s \rangle &::= a \mid b \mid c \mid a \langle s \rangle \mid b \langle s \rangle \mid c \langle s \rangle\end{aligned}$$

Add appropriate attributes and conditions to the grammar so that only  $\langle lstring \rangle$ 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, 'bacbbcbac' is legal, but not 'babbbcb'.) 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:

$$\begin{aligned}\langle exp \rangle &::= \langle simple \rangle \mid \langle exp \rangle + \langle exp \rangle \mid \langle exp \rangle * \langle exp \rangle \\ \langle simple \rangle &::= \langle number \rangle \mid \langle variable \rangle\end{aligned}$$

where  $\langle number \rangle$  and  $\langle variable \rangle$  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.

The assignment is due in class on Jan. 17. If you don't turn it in on the 17th, but turn it in by the *start* of the *next* class (Jan. 19), you will be penalised 20%. If you don't turn in the assignment by the start of class on the 19th, you will receive no credit for the assignment.