1. Problem 4.6

Production:	**Attribute rule:**
Stmt -> assignment	Stmt.md := assignment.md
Stmt -> subr_call	Stmt.md := subr_call.md
Assignment -> id := expr	Assignment.md := expr.md
Subr_call -> id (arg_list)	Subr_call.md := arg_list.md
Expr -> primary expr_tail	Expr.md := max(primary.md, expr_tail.md)
Expr_tail -> op expr	Expr_tail.md := 0
Primary -> id	Primary.md := 0
Primary -> subr_call	Primary.md := subr_call.md
Primary -> (expr)	Primary.md := expr.md + 1
Op -> + | - | * | /	
Arg_list -> expr agrs_tail	Args_list.md := max(expr.md, args_tail.md)
Args_tail -> , args_list	Args_tail.md := args_list.md
Args_tail -> ϵ	Args_tail.md := 0

2. Problem 6.2
   Fortran and Pascal will evaluate expressions from left to right so there should not be any nonintuitive evaluations of expressions.

3. Problem 6.5
   Parathensizes are required for Lisp otherwise expressions may be evaluated out of order.
   Ex1: (* 2 3 4 5 (- 16 9) 4) = 2 * 3 * 4 * 5 * (16 - 9) * 4 = 2 * 3 * 4 * 5 * 7 * 4 = 3360
   Ex2: (* 2 3 4 5 (- 16 9 4)) = 2 * 3 * 4 * 5 * (16 - 9 - 4) = 2 * 3 * 4 * 5 * 4 = 480

   In Section 6.1.1 we claimed that issues of precedence and associativity do not arise with prefix or postfix notation. Reword this claim to make explicit the hidden assumption.
   This should be reworded to the following: issues of precedence and associativity do not arise with prefix or postfix notation if and only if the number of operands that an operator takes in the prefix or postfix notation is fixed in the language.