Exercise 0: Try building "parse trees" based on the grammar described here for the following inputs:

```
a
a * b
a + b * c
a * b + c
a * ( b + c )
( a + b ) * c
a + b * c + b
```

Exercises

Exercise 1 Start with the grammar G6, repeated here:

G6: <exp> ::= <exp> + <mulexp> | <mulexp>
<mulexp> ::= <mulexp> * <rootexp> | <rootexp>
<rootexp> ::= (<exp>)
| a | b | c

Modify it in the following ways:

a. Add subtraction and division operators (- and /) with the customary precedence and associativity.
b. Then add a left-associative operator % between + and * in precedence.
c. Then add a right-associative operator = at lower precedence than any of the other operators.

Exercise 2 Give an EBNF grammar for each of the languages of Exercise 1. Use the EBNF extensions wherever possible to simplify the grammars. Include whatever notes to the reader are required to make the associativity of the operators clear.

Exercise 3 Show that each of the following grammars is ambiguous. (To show that a grammar is ambiguous, you must demonstrate that it can generate two parse trees for the same string.)

a. The grammar G4, repeated here:

G4: <exp> ::= <exp> + <exp>
| <exp> * <exp>
| (<exp>)
| a | b | c

b. This grammar:

<person> ::= <woman> | <man>
<woman> ::= wilma | betty | <empty>
<man> ::= fred | barney | <empty>

c. The following grammar for strings of balanced parentheses. (A language of any number of different kinds of balanced parentheses is called a Dyck language. This type of language plays an interesting role in the theory of formal languages.)

<S> ::= <S> <S> | (<S>) | ()