Informatik and Mathematical Modelling 29/10 04
DTU mrh

02140 Languages and Parsing
Programming Exercise 6

The purpose of this exercise is to become acquainted with scanning and parsing using the lexer generator `mosmllex` and the parser generator `mosmlyacc`. The starting point is the programs for scanning, parsing and evaluating a simple langauge of expressions, handed out at the lecture on Monday, November 1. (See also the homepage of the course.) The grammar for the expression language used to construct the parser is:

$$\begin{array}{lcll}
\mathit{Expr} & \rightarrow & \mathit{VAR} & \text{(variables — not elaborated further)} \\
 & | & \mathit{INT} & \text{(integers — not elaborated further)} \\
 & | & -\mathit{Expr} & \text{(monadic minus)} \\
 & | & \text{let } \mathit{VAR} = \mathit{Expr} \text{ in } \mathit{Expr} \text{ end} & \text{(local declaration)} \\
 & | & (\mathit{Expr}) & \\
 & | & \mathit{Expr} * \mathit{Expr} & \\
 & | & \mathit{Expr} + \mathit{Expr} & \\
 & | & \mathit{Expr} - \mathit{Expr} &
\end{array}$$

1. Extend the grammar with boolean expressions and an `if-then-else` construction to allow expressions like

   ```
   let x=2 in let y = 1 in if !(x = y & x < 1) | y=4
                           then x+x else x+y
             end
   end
   ```

   where `!`, `&` and `|` denote negation, conjunction and disjunction, respectively.

   Is your grammar ambigious? If so, how would you resolve this ambiguity using `mosmlyacc`?

   Extend the programs accordingly.

2. Extend the grammar with a sequence of local declarations in let constructions to allow expressions like:

   ```
   let x=2 y = 1 z=5
   in if !(x = y & x < 1) | y=4  then x+z else x+y end
   ```

   We want that “inner declarations” override “outer declarations” of the same identifier, e.g. the expression: `let x = 2 x = 3 in x end` must evaluate to 3.

   Extend your programs accordingly.