In the 5th coursework I introduced a CFG for the language P. Now we look at this grammar again to determine whether it can be implemented by an LL(1)-parser.

1. Construct First and Follow-sets for all non-terminal symbols of $G$ (using the augmented grammar $G^\$$).

2. Calculate the Lookahead-sets for each production to determine whether the grammar is LL(1).

3. (optional easter homework — deadline )
   Implement an LL(1) parser for P. This can be extended to an interpreter by allowing arbitrary names and numbers for Name and Num and by generating parse trees as implemented in P.java. Test your implementation on the P program prime.p which prints prime numbers — if your interpreter works.

Links

P.java [http://www.cs.nott.ac.uk/~txa/g51mal/P.java](http://www.cs.nott.ac.uk/~txa/g51mal/P.java)
prime.p [http://www.cs.nott.ac.uk/~txa/g51mal/prime.java](http://www.cs.nott.ac.uk/~txa/g51mal/prime.java)

Reminder: The syntax of P is given by the following CFG $G = (V, \Sigma, S, P)$:

- $V = \{Prog, Stmt, Stmts, Expr, Op, Name, Num\}$
- $\Sigma = \{\{,\},(,),+,*,-,div,if,while,print,=,;,0,1,x,y,z\}$
- $S = Prog$
• $P$ is given by:

\[
\begin{align*}
\text{Prog} & \rightarrow \{\text{Stmts}\} \\
\text{Stmts} & \rightarrow \epsilon \mid \text{Stmt} \space \text{Stmts} \\
\text{Stmt} & \rightarrow \text{Name} = \text{Expr}; \\
& \mid \text{if (Expr) Stmt} \\
& \mid \text{while (Expr) Stmt} \\
& \mid \text{print Expr ;} \\
& \mid ; \\
& \mid \text{Prog} \\
\text{Expr} & \rightarrow \text{Name} \mid \text{Num} \mid (\text{Expr} \space \text{Op} \space \text{Expr}) \\
\text{Name} & \rightarrow x \mid y \mid z \\
\text{Op} & \rightarrow + \mid * \mid - \mid \text{div} \\
\text{Num} & \rightarrow 0 \mid 1
\end{align*}
\]