COMP 520: Compilers

Written Assignment 3

Assigned: Thu Feb 17
Due: Tue Feb 22 (start of class)

(4 points each, 12 total)

(a) Show a stratified LL(1) grammar for parsing arithmetic expressions with three terminals {*num*, +, -} so that "–" can be used both as a binary subtraction operator and a unary negation operator. The grammar should yield a concrete syntax tree reflecting that unary negation is right-associative and binds more tightly than addition and subtraction. Addition and subtraction have the same precedence and are left-associative. For example, the expression `2 – 3 + -4 - - -5` should have a concrete syntax tree reflecting the ordering `(((2-3)+(-4))-(-(-5)))`.

(b) Describe how you would modify the simpleAST example on the class website to build Expr ASTs using *BinExpr*, *UnaryExpr* and *NumExpr* nodes. Describe your extensions to the scanner, parser, AbstractSyntaxTrees, and visitors. It's not necessary to write out complete code, but if you are interested you can extend the simpleAST example to try it out. (Use additional space on the back if needed)

(c) Draw the AST for `2 - 3 + -4 - - -5` using *BinExpr*, *UnaryExpr* and *NumExpr* nodes, and show the (spelling of the) tokens at the leaves of the AST.