

Table 1: Recursively-parsable LL(1) grammar for arithmetic expressions

		Predict
(1)	$\langle E \rangle \rightarrow \langle T \rangle \langle T_T \rangle$	(, +, -, var, num, func
(2)	$\langle T_T \rangle \rightarrow + \langle T \rangle \langle T_T \rangle$	+
(3)	$\langle T_T \rangle \rightarrow - \langle T \rangle \langle T_T \rangle$	-
(4)	$\langle T_T \rangle \rightarrow \epsilon$), EOF
(5)	$\langle T \rangle \rightarrow \langle F \rangle \langle F_T \rangle$	(, +, -, var, num, func
(6)	$\langle F \rangle \rightarrow \langle N \rangle$	+, -, num
(7)	$\langle F \rangle \rightarrow (\langle E \rangle)$	(
(8)	$\langle F \rangle \rightarrow \text{var}$	var
(9)	$\langle F \rangle \rightarrow \text{func} \langle F \rangle$	func
(10)	$\langle F_T \rangle \rightarrow * \langle F \rangle \langle F_T \rangle$	*
(11)	$\langle F_T \rangle \rightarrow / \langle F \rangle \langle F_T \rangle$	/
(12)	$\langle F_T \rangle \rightarrow ^ \langle F \rangle \langle F_T \rangle$	^
(13)	$\langle F_T \rangle \rightarrow \text{func} \langle F \rangle \langle F_T \rangle$	func
(13)	$\langle F_T \rangle \rightarrow \epsilon$	+, -,), EOF
(14)	$\langle N \rangle \rightarrow \text{num}$	num
(15)	$\langle N \rangle \rightarrow + \text{num}$	+
(16)	$\langle N \rangle \rightarrow - \text{num}$	-

- **num** denotes anything returned as a number from the scanner, e.g. 10.203, .105, 11., 212, 10.1E-15
- **func** includes any of sin, cos, tan, cot, csc, sec, lg, ln, log, exp
- When applying rule (13), implicitly insert a multiplication operator * before **func**.