CS160 Proj2 Grammar

Original:

Start	→	*ExprList*
ExprList	→	*ExprList* **;** *Expression*
	|	*Expression*
Expression	→	*Expression* **+** *Expression*
	|	*Expression* **-** *Expression*
	|	*Expression* ***** *Expression*
	|	*Expression* **/** *Expression*
	|	*Expression* **mod** *Expression*
	|	**(** *Expression* **)**
	|	**num**

With precedence:

Start -> List

List -> List ; Expression
Expression

Expression -> Expression + Term
Expression - Term
Term

Term -> Term * Factor
Term / Factor
Term mod Factor
Factor

Factor -> (Expression)
-> num

With Right Recursive, Left factoring:

Start	->	List
List	->	Expression List′
List′	->	; Expression List′
	->	e
Expression	->	Term Expression′
Expression′	->	+ Term Expression′
	->	- Term Expression′
	->	e
Term	->	Factor Term′
Term′	->	* Factor Term′
	->	/ Factor Term′
	->	mod Factor Term′
	->	e
Factor	->	(Expression)
	->	num

FIRST Set:

Start	{ (, num}
List	{ (, num }
List′	{ ; , e }
Expression	{ (, num }
Expression′	{ + , - , e }
Term	{ (, num }
Term′	{ * , / , mod , e }

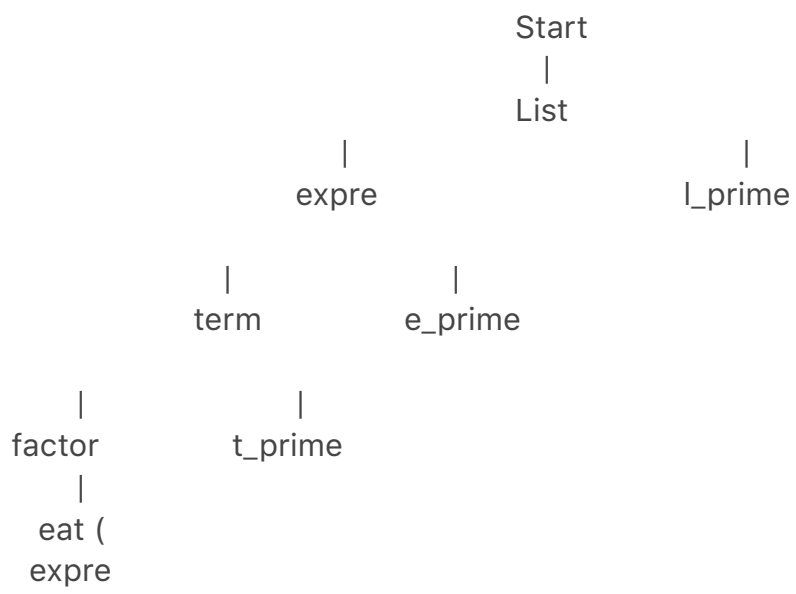
Factor	{ (, num }

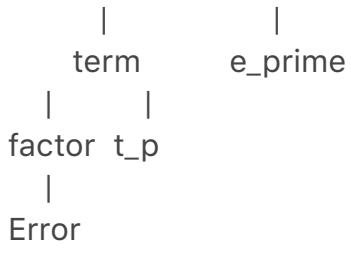
FOLLOW Set:

Start	{ $ }
List	{ $ }
List’	{ $ }
Expression	{ $, ; ,) }
Expression’	{ $, ; ,) }
Term	{ $, + , - , ; ,) }
Term’	{ $, + , - , ; ,) }
Factor	{ $, * , / , mod , + , - , ; ,) }

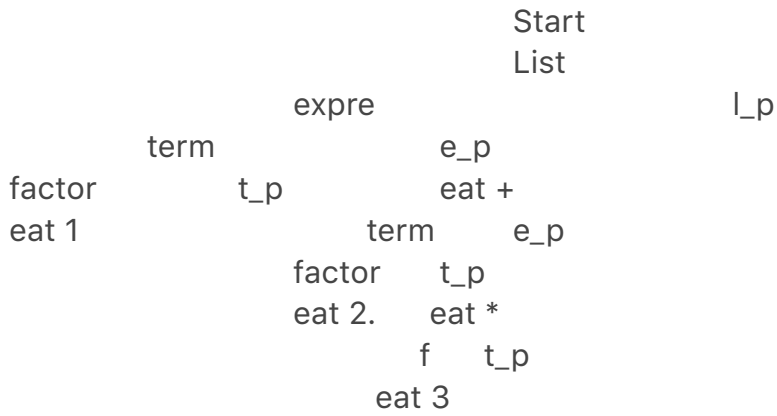
Test Case

( )




term
e_prime
factor
t_p
Error

1 + 2 * 3


Start
List
expre
l_p
term
e_p
factor
t_p
eat +
eat 1
term
e_p
factor
t_p
eat 2.
eat *
f
t_p
eat 3

3 * (2 + 1)


start
list
expr
l_p
term
e_p
factor
t_p

3 * factor t_p

(expr)

term e_p

factor t_p. + term e_p

2 f t_p

1

(1 + 2) * 3

start

list

expr

l_p

term

e_p

factor t_p

(expr) * factor t_p

term e_p 3

factor t_p + term e_p

1 e f t_p e

2 e

81mod3+10/3*3mod3

((2+43)mod6+12)/3; (2*3*2*19)mod8

Start->
List->
Expression list_prime->
Term expression_prime list_prime->
Factor term_prime expression_prime list_prime->
(Expression) term_prime expression_prime list_prime->
(Term expression_prime) term_prime expression_prime list_prime->
(Factor term_prime expression_prime) term_prime expression_prime list_prime->
((Expression) term_prime expression_prime) term_prime expression_prime
list_prime->
((Term expression_prime) term_prime expression_prime) term_prime
expression_prime list_prime->
((Factor term_prime expression_prime) term_prime expression_prime) term_prime
expression_prime list_prime->
((2 term_prime expression_prime) term_prime expression_prime) term_prime
expression_prime list_prime->
((2 expression_prime) term_prime expression_prime) term_prime
expression_prime list_prime->
((2 + term expression_prime) term_prime expression_prime) term_prime
expression_prime list_prime->
((2 + factor term_prime expression_prime) term_prime expression_prime)

term_prime expression_prime list_prime->
((2 + 43 term_prime expression_prime) term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43 expression_prime) term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod factor term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + term expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + factor term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12 term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12 expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / factor term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 list_prime->
((2 + 43) mod 6 + 12) / 3 ; expression list_prime->
((2 + 43) mod 6 + 12) / 3 ; term expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; factor term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (expression) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (term expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (factor term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * factor term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * factor term_prime expression_prime) term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 term_prime expression_prime) term_prime

expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * factor term_prime expression_prime)
term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19 term_prime expression_prime)
term_prime expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19 expression_prime) term_prime
expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19) term_prime expression_prime
list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19) mod factor term_prime
expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19) mod 8 term_prime expression_prime
list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19) mod 8 expression_prime list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19) mod 8 list_prime->
((2 + 43) mod 6 + 12) / 3 ; (2 * 3 * 2 * 19) mod 8