First and Follow Sets

	First Set	Follow Set
program	INT, VOID	$
declaration-list	INT, VOID	$
declaration	INT, VOID	INT, VOID, $
declaration'	(, [, ;	INT, VOID, $
var-declaration	INT	INT, (, NUM, ID, ;, {, }, IF, WHILE, RETURN, EPSILON
var-declaration'	[, ;	INT, VOID, $
fun-declaration'	(	INT, VOID, $
params	INT, VOID	)
param-list	INT	)
param	INT	COMMA,)
compound-stmt	{	INT, VOID, $, (, NUM, ID, ;, {, IF, WHILE, RETURN, ELSE, }
local-declarations	INT, EPSILON	(, NUM, ID, ;, {, }, IF, WHILE, RETURN, EPSILON
statement-list	(, NUM, ID, ;, {, IF, WHILE, RETURN, EPSILON	}
statement	(, NUM, ID, ;, {, IF, WHILE, RETURN	(, NUM, ID, ;, {, IF, WHILE, RETURN, ELSE, }
expression-stmt	(, NUM, ID, ;	(, NUM, ID, ;, {, IF, WHILE, RETURN, ELSE, }
selection-stmt	IF	(, NUM, ID, ;, {, IF, WHILE, RETURN, ELSE, }
iteration-stmt	WHILE	(, NUM, ID, ;, {, IF, WHILE, RETURN, ELSE, }
return-stmt	RETURN	(, NUM, ID, ;, {, IF, WHILE, RETURN, ELSE, }
expression	(, NUM, ID	;, COMMA,),],)
expression'	=, (, [, *, /, +, /, EPSILON	;, COMMA,),],)
expression''	=, *, /, +, -, EPSILON	;, COMMA,),],)
var	[, EPSILON	+, -, *, /, ;, COMMA,),],)
simple-expression'	*, /, +, -, <=, <, >, >=, ==, !=, EPSILON	;, COMMA,),],)
relop	<=, <, >, >=, ==, !=	(, NUM, ID

additive-expression	(, NUM, ID	;, COMMA,),],)
additive-expression'	*, /, +, -, EPSILON	*, /, ;, COMMA,),],), <=, <, >, >=, ==, !=
addop	+, -	(, NUM, ID
term	(, NUM, ID	+, -, *, /, ;, COMMA,),],), <=, <, >, >=, ==, !=
term'	*, /, EPSILON	+, -, *, /, ;, COMMA,),],)
mulop	*, /	(, NUM, ID
factor	(, NUM, ID	+, -, *, /, ;, COMMA,),],), <=, <, >, >=, ==, !=
varcall	EPSILON, [, (	+, -, *, /, ;, COMMA,),],)
call	(	+, -, *, /, ;, COMMA,),],)
args	(, NUM, ID, EPSILON	)
arg-list	(, NUM, ID	)