SLR grammar ('' is ε):

```
(0)  boolop -> <
(1)  boolop -> >
(2)  boolop -> <=
(3)  boolop -> >=
(4)  boolop -> ==
(5)  arithexpr -> multexpr
(6)  arithexprprime
(7)  arithexprprime -> +
(8)  multexpr arithexprprime
(9)  arithexprprime -> -
(10) multexpr arithexprprime
(11) arithexprprime -> ''
(12) multexpr -> simpleexpr
(13) multexprprime
(14) multexprprime -> *
(15) simpleexpr
(16) multexprprime
(17) multexprprime -> /
(18) simpleexpr
(19) multexprprime
(20) multexprprime -> ''
(21) simpleexpr -> ID
(22) simpleexpr -> NUM
(23) simpleexpr -> ( arithexpr )
```

>>

FIRST / FOLLOW table

Nonterminal	FIRST	FOLLOW
program'	{{}	{$}
program	{{}	{$}
stmt	{if,while,ID,{}	{},if,while,ID,{,else}
compoundstmt	{{}	{$,},if,while,ID,{,else}
stmts	{'',if,while,ID,{}	{}}
ifstmt	{if}	{},if,while,ID,{,else}
whilestmt	{while}	{},if,while,ID,{,else}
assgstmt	{ID}	{},if,while,ID,{,else}
boolexpr	{ID,NUM,(}	{)}
boolop	{<,>,<=,>=,==}	{ID,NUM,(}
arithexpr	{ID,NUM,(}	{;,<,>,<=,>=,==,)}
arithexprprime	{+,-,''}	{;,<,>,<=,>=,==,)}
multexpr	{ID,NUM,(}	{+,-,;,<,>,<=,>=,==,)}
multexprprime	{*,/,''}	{+,-,;,<,>,<=,>=,==,)}
simpleexpr	{ID,NUM,(}	{*,/,+,-,;,<,>,<=,>=,==,)}

SLR closure table

Goto	Kernel	State	Closure
	{program' -> .program}	0	{program' -> .program; program -> .compoundstmt; compoundstmt -> .{ stmts }}
goto(0, program)	{program' -> program.}	1	{program' -> program.}
goto(0, compoundstmt)	{program -> compoundstmt.}	2	{program -> compoundstmt.}
goto(0, {)	{compoundstmt -> {.stmts }}	3	{compoundstmt -> {.stmts }; stmts -> .stmt stmts; stmts -> .; stmt -> .ifstmt; stmt -> .whilestmt; stmt -> .assgstmt; stmt -> .compoundstmt; ifstmt -> .if (boolexpr) then stmt else stmt; whilestmt -> .while (boolexpr) stmt; assgstmt -> .ID = arithexpr ;; compoundstmt -> .{ stmts }}
goto(3, stmts)	{compoundstmt -> { stmts.}}	4	{compoundstmt -> { stmts.}}
goto(3, stmt)	{stmts -> stmt.stmts}	5	{stmts -> stmt.stmts; stmts -> .stmt stmts; stmts -> .; stmt -> .ifstmt; stmt -> .whilestmt; stmt -> .assgstmt; stmt -> .compoundstmt; ifstmt -> .if (boolexpr) then stmt else stmt; whilestmt -> .while (boolexpr) stmt; assgstmt -> .ID = arithexpr ;; compoundstmt -> .{ stmts }}
goto(3, ifstmt)	{stmt -> ifstmt.}	6	{stmt -> ifstmt.}
goto(3, whilestmt)	{stmt -> whilestmt.}	7	{stmt -> whilestmt.}
goto(3, assgstmt)	{stmt -> assgstmt.}	8	{stmt -> assgstmt.}
goto(3, compoundstmt)	{stmt -> compoundstmt.}	9	{stmt -> compoundstmt.}
goto(3, if)	{ifstmt -> if.(boolexpr) then stmt else stmt}	10	{ifstmt -> if.(boolexpr) then stmt else stmt}
goto(3, while)	{whilestmt -> while.(boolexpr) stmt}	11	{whilestmt -> while.(boolexpr) stmt}
goto(3, ID)	{assgstmt -> ID.= arithexpr ;}	12	{assgstmt -> ID.= arithexpr ;}
goto(3, {)	{compoundstmt -> {.stmts }}	3	
goto(4, })	{compoundstmt -> { stmts }.}	13	{compoundstmt -> { stmts }.}
goto(5, stmts)	{stmts -> stmt stmts.}	14	{stmts -> stmt stmts.}
goto(5, stmt)	{stmts -> stmt.stmts}	5	
goto(5, ifstmt)	{stmt -> ifstmt.}	6	
goto(5, whilestmt)	{stmt -> whilestmt.}	7	
goto(5, assgstmt)	{stmt -> assgstmt.}	8	
goto(5, compoundstmt)	{stmt -> compoundstmt.}	9	
goto(5, if)	{ifstmt -> if.(boolexpr) then stmt else stmt}	10	
goto(5, while)	{whilestmt -> while.(boolexpr) stmt}	11	
goto(5, ID)	{assgstmt -> ID.= arithexpr ;}	12	
goto(5, {)	{compoundstmt -> {.stmts }}	3	
goto(10, ()	{ifstmt -> if (.boolexpr) then stmt else stmt}	15	{ifstmt -> if (.boolexpr) then stmt else stmt; boolexpr -> .arithexpr boolop arithexpr; arithexpr -> .multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(11, ()	{whilestmt -> while (.boolexpr) stmt}	16	{whilestmt -> while (.boolexpr) stmt; boolexpr -> .arithexpr boolop arithexpr; arithexpr -> .multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(12, =)	{assgstmt -> ID =.arithexpr ;}	17	{assgstmt -> ID =.arithexpr ;; arithexpr -> .multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(15, boolexpr)	{ifstmt -> if (boolexpr.) then stmt else stmt}	18	{ifstmt -> if (boolexpr.) then stmt else stmt}
goto(15, arithexpr)	{boolexpr -> arithexpr.boolop arithexpr}	19	{boolexpr -> arithexpr.boolop arithexpr; boolop -> .<; boolop -> .>; boolop -> .<=; boolop -> .>=; boolop -> .==}
goto(15, multexpr)	{arithexpr -> multexpr.arithexprprime}	20	{arithexpr -> multexpr.arithexprprime; arithexprprime -> .+ multexpr arithexprprime; arithexprprime -> .- multexpr arithexprprime; arithexprprime -> .}
goto(15, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	{multexpr -> simpleexpr.multexprprime; multexprprime -> .* simpleexpr multexprprime; multexprprime -> ./ simpleexpr multexprprime; multexprprime -> .}
goto(15, ID)	{simpleexpr -> ID.}	22	{simpleexpr -> ID.}
goto(15, NUM)	{simpleexpr -> NUM.}	23	{simpleexpr -> NUM.}
goto(15, ()	{simpleexpr -> (.arithexpr)}	24	{simpleexpr -> (.arithexpr); arithexpr -> .multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(16, boolexpr)	{whilestmt -> while (boolexpr.) stmt}	25	{whilestmt -> while (boolexpr.) stmt}
goto(16, arithexpr)	{boolexpr -> arithexpr.boolop arithexpr}	19	
goto(16, multexpr)	{arithexpr -> multexpr.arithexprprime}	20	
goto(16, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	
goto(16, ID)	{simpleexpr -> ID.}	22	
goto(16, NUM)	{simpleexpr -> NUM.}	23	
goto(16, ()	{simpleexpr -> (.arithexpr)}	24	
goto(17, arithexpr)	{assgstmt -> ID = arithexpr.;}	26	{assgstmt -> ID = arithexpr.;}
goto(17, multexpr)	{arithexpr -> multexpr.arithexprprime}	20	
goto(17, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	
goto(17, ID)	{simpleexpr -> ID.}	22	
goto(17, NUM)	{simpleexpr -> NUM.}	23	
goto(17, ()	{simpleexpr -> (.arithexpr)}	24	
goto(18,))	{ifstmt -> if (boolexpr).then stmt else stmt}	27	{ifstmt -> if (boolexpr).then stmt else stmt}
goto(19, boolop)	{boolexpr -> arithexpr boolop.arithexpr}	28	{boolexpr -> arithexpr boolop.arithexpr; arithexpr -> .multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(19, <)	{boolop -> <.}	29	{boolop -> <.}
goto(19, >)	{boolop -> >.}	30	{boolop -> >.}
goto(19, <=)	{boolop -> <=.}	31	{boolop -> <=.}
goto(19, >=)	{boolop -> >=.}	32	{boolop -> >=.}
goto(19, ==)	{boolop -> ==.}	33	{boolop -> ==.}
goto(20, arithexprprime)	{arithexpr -> multexpr arithexprprime.}	34	{arithexpr -> multexpr arithexprprime.}
goto(20, +)	{arithexprprime -> +.multexpr arithexprprime}	35	{arithexprprime -> +.multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(20, -)	{arithexprprime -> -.multexpr arithexprprime}	36	{arithexprprime -> -.multexpr arithexprprime; multexpr -> .simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(21, multexprprime)	{multexpr -> simpleexpr multexprprime.}	37	{multexpr -> simpleexpr multexprprime.}
goto(21, *)	{multexprprime -> *.simpleexpr multexprprime}	38	{multexprprime -> *.simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(21, /)	{multexprprime -> /.simpleexpr multexprprime}	39	{multexprprime -> /.simpleexpr multexprprime; simpleexpr -> .ID; simpleexpr -> .NUM; simpleexpr -> .(arithexpr)}
goto(24, arithexpr)	{simpleexpr -> (arithexpr.)}	40	{simpleexpr -> (arithexpr.)}
goto(24, multexpr)	{arithexpr -> multexpr.arithexprprime}	20	
goto(24, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	
goto(24, ID)	{simpleexpr -> ID.}	22	
goto(24, NUM)	{simpleexpr -> NUM.}	23	
goto(24, ()	{simpleexpr -> (.arithexpr)}	24	
goto(25,))	{whilestmt -> while (boolexpr).stmt}	41	{whilestmt -> while (boolexpr).stmt; stmt -> .ifstmt; stmt -> .whilestmt; stmt -> .assgstmt; stmt -> .compoundstmt; ifstmt -> .if (boolexpr) then stmt else stmt; whilestmt -> .while (boolexpr) stmt; assgstmt -> .ID = arithexpr ;; compoundstmt -> .{ stmts }}
goto(26, ;)	{assgstmt -> ID = arithexpr ;.}	42	{assgstmt -> ID = arithexpr ;.}
goto(27, then)	{ifstmt -> if (boolexpr) then.stmt else stmt}	43	{ifstmt -> if (boolexpr) then.stmt else stmt; stmt -> .ifstmt; stmt -> .whilestmt; stmt -> .assgstmt; stmt -> .compoundstmt; ifstmt -> .if (boolexpr) then stmt else stmt; whilestmt -> .while (boolexpr) stmt; assgstmt -> .ID = arithexpr ;; compoundstmt -> .{ stmts }}
goto(28, arithexpr)	{boolexpr -> arithexpr boolop arithexpr.}	44	{boolexpr -> arithexpr boolop arithexpr.}
goto(28, multexpr)	{arithexpr -> multexpr.arithexprprime}	20	
goto(28, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	
goto(28, ID)	{simpleexpr -> ID.}	22	
goto(28, NUM)	{simpleexpr -> NUM.}	23	
goto(28, ()	{simpleexpr -> (.arithexpr)}	24	
goto(35, multexpr)	{arithexprprime -> + multexpr.arithexprprime}	45	{arithexprprime -> + multexpr.arithexprprime; arithexprprime -> .+ multexpr arithexprprime; arithexprprime -> .- multexpr arithexprprime; arithexprprime -> .}
goto(35, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	
goto(35, ID)	{simpleexpr -> ID.}	22	
goto(35, NUM)	{simpleexpr -> NUM.}	23	
goto(35, ()	{simpleexpr -> (.arithexpr)}	24	
goto(36, multexpr)	{arithexprprime -> - multexpr.arithexprprime}	46	{arithexprprime -> - multexpr.arithexprprime; arithexprprime -> .+ multexpr arithexprprime; arithexprprime -> .- multexpr arithexprprime; arithexprprime -> .}
goto(36, simpleexpr)	{multexpr -> simpleexpr.multexprprime}	21	
goto(36, ID)	{simpleexpr -> ID.}	22	
goto(36, NUM)	{simpleexpr -> NUM.}	23	
goto(36, ()	{simpleexpr -> (.arithexpr)}	24	
goto(38, simpleexpr)	{multexprprime -> * simpleexpr.multexprprime}	47	{multexprprime -> * simpleexpr.multexprprime; multexprprime -> .* simpleexpr multexprprime; multexprprime -> ./ simpleexpr multexprprime; multexprprime -> .}
goto(38, ID)	{simpleexpr -> ID.}	22	
goto(38, NUM)	{simpleexpr -> NUM.}	23	
goto(38, ()	{simpleexpr -> (.arithexpr)}	24	
goto(39, simpleexpr)	{multexprprime -> / simpleexpr.multexprprime}	48	{multexprprime -> / simpleexpr.multexprprime; multexprprime -> .* simpleexpr multexprprime; multexprprime -> ./ simpleexpr multexprprime; multexprprime -> .}
goto(39, ID)	{simpleexpr -> ID.}	22	
goto(39, NUM)	{simpleexpr -> NUM.}	23	
goto(39, ()	{simpleexpr -> (.arithexpr)}	24	
goto(40,))	{simpleexpr -> (arithexpr).}	49	{simpleexpr -> (arithexpr).}
goto(41, stmt)	{whilestmt -> while (boolexpr) stmt.}	50	{whilestmt -> while (boolexpr) stmt.}
goto(41, ifstmt)	{stmt -> ifstmt.}	6	
goto(41, whilestmt)	{stmt -> whilestmt.}	7	
goto(41, assgstmt)	{stmt -> assgstmt.}	8	
goto(41, compoundstmt)	{stmt -> compoundstmt.}	9	
goto(41, if)	{ifstmt -> if.(boolexpr) then stmt else stmt}	10	
goto(41, while)	{whilestmt -> while.(boolexpr) stmt}	11	
goto(41, ID)	{assgstmt -> ID.= arithexpr ;}	12	
goto(41, {)	{compoundstmt -> {.stmts }}	3	
goto(43, stmt)	{ifstmt -> if (boolexpr) then stmt.else stmt}	51	{ifstmt -> if (boolexpr) then stmt.else stmt}
goto(43, ifstmt)	{stmt -> ifstmt.}	6	
goto(43, whilestmt)	{stmt -> whilestmt.}	7	
goto(43, assgstmt)	{stmt -> assgstmt.}	8	
goto(43, compoundstmt)	{stmt -> compoundstmt.}	9	
goto(43, if)	{ifstmt -> if.(boolexpr) then stmt else stmt}	10	
goto(43, while)	{whilestmt -> while.(boolexpr) stmt}	11	
goto(43, ID)	{assgstmt -> ID.= arithexpr ;}	12	
goto(43, {)	{compoundstmt -> {.stmts }}	3	
goto(45, arithexprprime)	{arithexprprime -> + multexpr arithexprprime.}	52	{arithexprprime -> + multexpr arithexprprime.}
goto(45, +)	{arithexprprime -> +.multexpr arithexprprime}	35	
goto(45, -)	{arithexprprime -> -.multexpr arithexprprime}	36	
goto(46, arithexprprime)	{arithexprprime -> - multexpr arithexprprime.}	53	{arithexprprime -> - multexpr arithexprprime.}
goto(46, +)	{arithexprprime -> +.multexpr arithexprprime}	35	
goto(46, -)	{arithexprprime -> -.multexpr arithexprprime}	36	
goto(47, multexprprime)	{multexprprime -> * simpleexpr multexprprime.}	54	{multexprprime -> * simpleexpr multexprprime.}
goto(47, *)	{multexprprime -> *.simpleexpr multexprprime}	38	
goto(47, /)	{multexprprime -> /.simpleexpr multexprprime}	39	
goto(48, multexprprime)	{multexprprime -> / simpleexpr multexprprime.}	55	{multexprprime -> / simpleexpr multexprprime.}
goto(48, *)	{multexprprime -> *.simpleexpr multexprprime}	38	
goto(48, /)	{multexprprime -> /.simpleexpr multexprprime}	39	
goto(51, else)	{ifstmt -> if (boolexpr) then stmt else.stmt}	56	{ifstmt -> if (boolexpr) then stmt else.stmt; stmt -> .ifstmt; stmt -> .whilestmt; stmt -> .assgstmt; stmt -> .compoundstmt; ifstmt -> .if (boolexpr) then stmt else stmt; whilestmt -> .while (boolexpr) stmt; assgstmt -> .ID = arithexpr ;; compoundstmt -> .{ stmts }}
goto(56, stmt)	{ifstmt -> if (boolexpr) then stmt else stmt.}	57	{ifstmt -> if (boolexpr) then stmt else stmt.}
goto(56, ifstmt)	{stmt -> ifstmt.}	6	
goto(56, whilestmt)	{stmt -> whilestmt.}	7	
goto(56, assgstmt)	{stmt -> assgstmt.}	8	
goto(56, compoundstmt)	{stmt -> compoundstmt.}	9	
goto(56, if)	{ifstmt -> if.(boolexpr) then stmt else stmt}	10	
goto(56, while)	{whilestmt -> while.(boolexpr) stmt}	11	
goto(56, ID)	{assgstmt -> ID.= arithexpr ;}	12	
goto(56, {)	{compoundstmt -> {.stmts }}	3	

LR table

State	ACTION {	}	if	(	)	then	else	while	ID	=	;	<	>	<=	>=	==	+	-	*	/	NUM	$	GOTO program'	program	stmt	compoundstmt	stmts	ifstmt	whilestmt	assgstmt	boolexpr	boolop	arithexpr	arithexprprime	multexpr	multexprprime	simpleexpr
0	s3																							1		2											
1																						acc															
2																						r1															
3	s3	r8	s10					s11	s12																5	9		4	6	7	8							
4		s13																																				
5	s3	r8	s10					s11	s12																5	9		14	6	7	8							
6	r2	r2	r2				r2	r2	r2																													
7	r3	r3	r3				r3	r3	r3																													
8	r4	r4	r4				r4	r4	r4																													
9	r5	r5	r5				r5	r5	r5																													
10				s15																																		
11				s16																																		
12										s17																												
13	r6	r6	r6				r6	r6	r6													r6															
14		r7																																				
15				s24					s22												s23									18		19		20		21	
16				s24					s22												s23									25		19		20		21	
17				s24					s22												s23											26		20		21	
18					s27																																	
19												s29	s30	s31	s32	s33																28						
20					r21						r21	r21	r21	r21	r21	r21	s35	s36																	34				
21					r25						r25	r25	r25	r25	r25	r25	r25	r25	s38	s39																	37		
22					r26						r26	r26	r26	r26	r26	r26	r26	r26	r26	r26																			
23					r27						r27	r27	r27	r27	r27	r27	r27	r27	r27	r27																			
24				s24					s22												s23											40		20		21	
25					s41																																	
26											s42																											
27						s43																																
28				s24					s22												s23											44		20		21	
29				r13					r13												r13																
30				r14					r14												r14																
31				r15					r15												r15																
32				r16					r16												r16																
33				r17					r17												r17																
34					r18						r18	r18	r18	r18	r18	r18																						
35				s24					s22												s23													45		21	
36				s24					s22												s23													46		21	
37					r22						r22	r22	r22	r22	r22	r22	r22	r22																				
38				s24					s22												s23															47	
39				s24					s22												s23															48	
40					s49																																	
41	s3		s10					s11	s12																50	9			6	7	8							
42	r11	r11	r11				r11	r11	r11																													
43	s3		s10					s11	s12																51	9			6	7	8							
44					r12																																	
45					r21						r21	r21	r21	r21	r21	r21	s35	s36																	52				
46					r21						r21	r21	r21	r21	r21	r21	s35	s36																	53				
47					r25						r25	r25	r25	r25	r25	r25	r25	r25	s38	s39																	54		
48					r25						r25	r25	r25	r25	r25	r25	r25	r25	s38	s39																	55		
49					r28						r28	r28	r28	r28	r28	r28	r28	r28	r28	r28																			
50	r10	r10	r10				r10	r10	r10																													
51							s56																															
52					r19						r19	r19	r19	r19	r19	r19																						
53					r20						r20	r20	r20	r20	r20	r20																						
54					r23						r23	r23	r23	r23	r23	r23	r23	r23																				
55					r24						r24	r24	r24	r24	r24	r24	r24	r24																				
56	s3		s10					s11	s12																57	9			6	7	8							
57	r9	r9	r9				r9	r9	r9																													

Input (tokens): id + id * id

Maximum number of steps: 100

PARSE

Trace				Tree
Step	Stack	Input	Action	
1	0	id + id * id $		