>>

SLR grammar ('' is ε):

```
(0)  CODE -> VDECL CODE
(1)  CODE -> FDECL CODE
(2)  CODE -> CDECL CODE
(3)  CODE -> ' '
(4)  VDECL -> vtype id semi
(5)  VDECL -> vtype ASSIGN semi
(6)  ASSIGN -> id assign RHS
(7)  RHS -> EXPR
(8)  RHS -> literal
(9)  RHS -> character
(10) RHS -> boolstr
(11) EXPR -> EXPR addsub EXPR
(12) EXPR -> EXPR multdiv EXPR
(13) EXPR -> lparen EXPR rparen
(14) EXPR -> id
(15) EXPR -> num
(16) FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN
     rbrace
(17) ARG -> vtype id MOREARGS
(18) ARG -> ' '
(19) MOREARGS -> comma vtype id MOREARGS
(20) MOREARGS -> ' '
(21) BLOCK -> STMT BLOCK
(22) BLOCK -> ' '
(23) STMT ->  VDECL
(24) STMT -> ASSIGN semi
(25) STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE
(26) STMT -> while lparen COND rparen lbrace BLOCK rbrace
(27) COND -> COND comp COND
(28) COND -> boolstr
(29) ELSE -> else lbrace BLOCK rbrace
(30) ELSE -> ' '
(31) RETURN -> return RHS semi
(32) CDECL -> class id lbrace ODECL rbrace
(33) ODECL -> VDECL ODECL
(34) ODECL -> FDECL ODECL
(35) ODECL -> ' '
```

FIRST / FOLLOW table

Nonterminal	FIRST	FOLLOW
CODE	{',vtype,class}	{$}
VDECL	{vtype}	{',vtype,class,id,if,while}
ASSIGN	{id}	{semi}
RHS	{literal,character,boolstr,lparen,id,num}	{semi}
EXPR	{lparen,id,num}	{semi,addsub,multdiv,rparen}
FDECL	{vtype}	{',vtype,class}
ARG	{vtype,'}	{rparen}
MOREARGS	{comma,'}	{rparen}
BLOCK	{',vtype,id,if,while}	{return,rbrace}
STMT	{vtype,id,if,while}	{',vtype,id,if,while}
COND	{boolstr}	{rparen,comp}
ELSE	{else,'}	{',vtype,id,if,while}
RETURN	{return}	{rbrace}
CDECL	{class}	{',vtype,class}
ODECL	{vtype,'}	{rbrace}

SLR closure table

Goto	Kernel	State	Closure
	{CODE -> .VDECL CODE}	0	{CODE -> .VDECL CODE; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi}
goto(0, VDECL)	{CODE -> VDECL.CODE}	1	{CODE -> VDECL.CODE; CODE -> .VDECL CODE; CODE -> .FDECL CODE; CODE -> .CDECL CODE; CODE -> .' '; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace; CDECL -> .class id lbrace ODECL rbrace}
goto(0, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi}	2	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; ASSIGN -> .id assign RHS}
goto(1, CODE)	{CODE -> VDECL CODE.}	3	{CODE -> VDECL CODE.}
goto(1, VDECL)	{CODE -> VDECL.CODE}	1	
goto(1, FDECL)	{CODE -> FDECL.CODE}	4	{CODE -> FDECL.CODE; CODE -> .VDECL CODE; CODE -> .FDECL CODE; CODE -> .CDECL CODE; CODE -> .' '; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace; CDECL -> .class id lbrace ODECL rbrace}
goto(1, CDECL)	{CODE -> CDECL.CODE}	5	{CODE -> CDECL.CODE; CODE -> .VDECL CODE; CODE -> .FDECL CODE; CODE -> .CDECL CODE; CODE -> .' '; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace; CDECL -> .class id lbrace ODECL rbrace}
goto(1, ')	{CODE -> '.'}	6	{CODE -> '.'}
goto(1, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace}	7	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace; ASSIGN -> .id assign RHS}
goto(1, class)	{CDECL -> class.id lbrace ODECL rbrace}	8	{CDECL -> class.id lbrace ODECL rbrace}
goto(2, id)	{VDECL -> vtype id.semi; ASSIGN -> id.assign RHS}	9	{VDECL -> vtype id.semi; ASSIGN -> id.assign RHS}
goto(2, ASSIGN)	{VDECL -> vtype ASSIGN.semi}	10	{VDECL -> vtype ASSIGN.semi}
goto(4, CODE)	{CODE -> FDECL CODE.}	11	{CODE -> FDECL CODE.}
goto(4, VDECL)	{CODE -> VDECL.CODE}	1	
goto(4, FDECL)	{CODE -> FDECL.CODE}	4	
goto(4, CDECL)	{CODE -> CDECL.CODE}	5	
goto(4, ')	{CODE -> '.'}	6	
goto(4, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace}	7	
goto(4, class)	{CDECL -> class.id lbrace ODECL rbrace}	8	
goto(5, CODE)	{CODE -> CDECL CODE.}	12	{CODE -> CDECL CODE.}
goto(5, VDECL)	{CODE -> VDECL.CODE}	1	
goto(5, FDECL)	{CODE -> FDECL.CODE}	4	
goto(5, CDECL)	{CODE -> CDECL.CODE}	5	
goto(5, ')	{CODE -> '.'}	6	
goto(5, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace}	7	
goto(5, class)	{CDECL -> class.id lbrace ODECL rbrace}	8	
goto(6, ')	{CODE -> ' '.}	13	{CODE -> ' '.}
goto(7, id)	{VDECL -> vtype id.semi; FDECL -> vtype id.lparen ARG rparen lbrace BLOCK RETURN rbrace; ASSIGN -> id.assign RHS}	14	{VDECL -> vtype id.semi; FDECL -> vtype id.lparen ARG rparen lbrace BLOCK RETURN rbrace; ASSIGN -> id.assign RHS}
goto(7, ASSIGN)	{VDECL -> vtype ASSIGN.semi}	10	
goto(8, id)	{CDECL -> class id.lbrace ODECL rbrace}	15	{CDECL -> class id.lbrace ODECL rbrace}
goto(9, semi)	{VDECL -> vtype id semi.}	16	{VDECL -> vtype id semi.}
goto(9, assign)	{ASSIGN -> id assign.RHS}	17	{ASSIGN -> id assign.RHS; RHS -> .EXPR; RHS -> .literal; RHS -> .character; RHS -> .boolstr; EXPR -> .EXPR addsub EXPR; EXPR -> .EXPR multdiv EXPR; EXPR -> .lparen EXPR rparen; EXPR -> .id; EXPR -> .num}
goto(10, semi)	{VDECL -> vtype ASSIGN semi.}	18	{VDECL -> vtype ASSIGN semi.}
goto(14, semi)	{VDECL -> vtype id semi.}	16	
goto(14, lparen)	{FDECL -> vtype id lparen.ARG rparen lbrace BLOCK RETURN rbrace}	19	{FDECL -> vtype id lparen.ARG rparen lbrace BLOCK RETURN rbrace; ARG -> .vtype id MOREARGS; ARG -> .' '}
goto(14, assign)	{ASSIGN -> id assign.RHS}	17	
goto(15, lbrace)	{CDECL -> class id lbrace.ODECL rbrace}	20	{CDECL -> class id lbrace.ODECL rbrace; ODECL -> .VDECL ODECL; ODECL -> .FDECL ODECL; ODECL -> .' '; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace}
goto(17, RHS)	{ASSIGN -> id assign RHS.}	21	{ASSIGN -> id assign RHS.}
goto(17, EXPR)	{RHS -> EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}	22	{RHS -> EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}
goto(17, literal)	{RHS -> literal.}	23	{RHS -> literal.}
goto(17, character)	{RHS -> character.}	24	{RHS -> character.}
goto(17, boolstr)	{RHS -> boolstr.}	25	{RHS -> boolstr.}
goto(17, lparen)	{EXPR -> lparen.EXPR rparen}	26	{EXPR -> lparen.EXPR rparen; EXPR -> .EXPR addsub EXPR; EXPR -> .EXPR multdiv EXPR; EXPR -> .lparen EXPR rparen; EXPR -> .id; EXPR -> .num}
goto(17, id)	{EXPR -> id.}	27	{EXPR -> id.}
goto(17, num)	{EXPR -> num.}	28	{EXPR -> num.}
goto(19, ARG)	{FDECL -> vtype id lparen ARG.rparen lbrace BLOCK RETURN rbrace}	29	{FDECL -> vtype id lparen ARG.rparen lbrace BLOCK RETURN rbrace}
goto(19, vtype)	{ARG -> vtype.id MOREARGS}	30	{ARG -> vtype.id MOREARGS}
goto(19, ')	{ARG -> '.'}	31	{ARG -> '.'}
goto(20, ODECL)	{CDECL -> class id lbrace ODECL.rbrace}	32	{CDECL -> class id lbrace ODECL.rbrace}
goto(20, VDECL)	{ODECL -> VDECL.ODECL}	33	{ODECL -> VDECL.ODECL; ODECL -> .VDECL ODECL; ODECL -> .FDECL ODECL; ODECL -> .' '; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace}
goto(20, FDECL)	{ODECL -> FDECL.ODECL}	34	{ODECL -> FDECL.ODECL; ODECL -> .VDECL ODECL; ODECL -> .FDECL ODECL; ODECL -> .' '; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; FDECL -> .vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace}
goto(20, ')	{ODECL -> '.'}	35	{ODECL -> '.'}
goto(20, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace}	7	
goto(22, addsub)	{EXPR -> EXPR addsub.EXPR}	36	{EXPR -> EXPR addsub.EXPR; EXPR -> .EXPR addsub EXPR; EXPR -> .EXPR multdiv EXPR; EXPR -> .lparen EXPR rparen; EXPR -> .id; EXPR -> .num}
goto(22, multdiv)	{EXPR -> EXPR multdiv.EXPR}	37	{EXPR -> EXPR multdiv.EXPR; EXPR -> .EXPR addsub EXPR; EXPR -> .EXPR multdiv EXPR; EXPR -> .lparen EXPR rparen; EXPR -> .id; EXPR -> .num}
goto(26, EXPR)	{EXPR -> lparen EXPR.rparen; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}	38	{EXPR -> lparen EXPR.rparen; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}
goto(26, lparen)	{EXPR -> lparen.EXPR rparen}	26	
goto(26, id)	{EXPR -> id.}	27	
goto(26, num)	{EXPR -> num.}	28	
goto(29, rparen)	{FDECL -> vtype id lparen ARG rparen.lbrace BLOCK RETURN rbrace}	39	{FDECL -> vtype id lparen ARG rparen.lbrace BLOCK RETURN rbrace}
goto(30, id)	{ARG -> vtype id.MOREARGS}	40	{ARG -> vtype id.MOREARGS; MOREARGS -> .comma vtype id MOREARGS; MOREARGS -> .' '}
goto(31, ')	{ARG -> ' '.}	41	{ARG -> ' '.}
goto(32, rbrace)	{CDECL -> class id lbrace ODECL rbrace.}	42	{CDECL -> class id lbrace ODECL rbrace.}
goto(33, ODECL)	{ODECL -> VDECL ODECL.}	43	{ODECL -> VDECL ODECL.}
goto(33, VDECL)	{ODECL -> VDECL.ODECL}	33	
goto(33, FDECL)	{ODECL -> FDECL.ODECL}	34	
goto(33, ')	{ODECL -> '.'}	35	
goto(33, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace}	7	
goto(34, ODECL)	{ODECL -> FDECL ODECL.}	44	{ODECL -> FDECL ODECL.}
goto(34, VDECL)	{ODECL -> VDECL.ODECL}	33	
goto(34, FDECL)	{ODECL -> FDECL.ODECL}	34	
goto(34, ')	{ODECL -> '.'}	35	
goto(34, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi; FDECL -> vtype.id lparen ARG rparen lbrace BLOCK RETURN rbrace}	7	
goto(35, ')	{ODECL -> ' '.}	45	{ODECL -> ' '.}
goto(36, EXPR)	{EXPR -> EXPR addsub EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}	46	{EXPR -> EXPR addsub EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}
goto(36, lparen)	{EXPR -> lparen.EXPR rparen}	26	
goto(36, id)	{EXPR -> id.}	27	
goto(36, num)	{EXPR -> num.}	28	
goto(37, EXPR)	{EXPR -> EXPR multdiv EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}	47	{EXPR -> EXPR multdiv EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}
goto(37, lparen)	{EXPR -> lparen.EXPR rparen}	26	
goto(37, id)	{EXPR -> id.}	27	
goto(37, num)	{EXPR -> num.}	28	
goto(38, rparen)	{EXPR -> lparen EXPR rparen.}	48	{EXPR -> lparen EXPR rparen.}
goto(38, addsub)	{EXPR -> EXPR addsub.EXPR}	36	
goto(38, multdiv)	{EXPR -> EXPR multdiv.EXPR}	37	
goto(39, lbrace)	{FDECL -> vtype id lparen ARG rparen lbrace.BLOCK RETURN rbrace}	49	{FDECL -> vtype id lparen ARG rparen lbrace.BLOCK RETURN rbrace; BLOCK -> .STMT BLOCK; BLOCK -> .' '; STMT -> .VDECL; STMT -> .ASSIGN semi; STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE; STMT -> .while lparen COND rparen lbrace BLOCK rbrace; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; ASSIGN -> .id assign RHS}
goto(40, MOREARGS)	{ARG -> vtype id MOREARGS.}	50	{ARG -> vtype id MOREARGS.}
goto(40, comma)	{MOREARGS -> comma.vtype id MOREARGS}	51	{MOREARGS -> comma.vtype id MOREARGS}
goto(40, ')	{MOREARGS -> '.'}	52	{MOREARGS -> '.'}
goto(46, addsub)	{EXPR -> EXPR addsub.EXPR}	36	
goto(46, multdiv)	{EXPR -> EXPR multdiv.EXPR}	37	
goto(47, addsub)	{EXPR -> EXPR addsub.EXPR}	36	
goto(47, multdiv)	{EXPR -> EXPR multdiv.EXPR}	37	
goto(49, BLOCK)	{FDECL -> vtype id lparen ARG rparen lbrace BLOCK.RETURN rbrace}	53	{FDECL -> vtype id lparen ARG rparen lbrace BLOCK.RETURN rbrace; RETURN -> .return RHS semi}
goto(49, STMT)	{BLOCK -> STMT.BLOCK}	54	{BLOCK -> STMT.BLOCK; BLOCK -> .STMT BLOCK; BLOCK -> .' '; STMT -> .VDECL; STMT -> .ASSIGN semi; STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE; STMT -> .while lparen COND rparen lbrace BLOCK rbrace; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; ASSIGN -> .id assign RHS}
goto(49, ')	{BLOCK -> '.'}	55	{BLOCK -> '.'}
goto(49, VDECL)	{STMT -> VDECL.}	56	{STMT -> VDECL.}
goto(49, ASSIGN)	{STMT -> ASSIGN.semi}	57	{STMT -> ASSIGN.semi}
goto(49, if)	{STMT -> if.lparen COND rparen lbrace BLOCK rbrace ELSE}	58	{STMT -> if.lparen COND rparen lbrace BLOCK rbrace ELSE}
goto(49, while)	{STMT -> while.lparen COND rparen lbrace BLOCK rbrace}	59	{STMT -> while.lparen COND rparen lbrace BLOCK rbrace}
goto(49, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi}	2	
goto(49, id)	{ASSIGN -> id.assign RHS}	60	{ASSIGN -> id.assign RHS}
goto(51, vtype)	{MOREARGS -> comma vtype.id MOREARGS}	61	{MOREARGS -> comma vtype.id MOREARGS}
goto(52, ')	{MOREARGS -> ' '.}	62	{MOREARGS -> ' '.}
goto(53, RETURN)	{FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN.rbrace}	63	{FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN.rbrace}
goto(53, return)	{RETURN -> return.RHS semi}	64	{RETURN -> return.RHS semi; RHS -> .EXPR; RHS -> .literal; RHS -> .character; RHS -> .boolstr; EXPR -> .EXPR addsub EXPR; EXPR -> .EXPR multdiv EXPR; EXPR -> .lparen EXPR rparen; EXPR -> .id; EXPR -> .num}
goto(54, BLOCK)	{BLOCK -> STMT BLOCK.}	65	{BLOCK -> STMT BLOCK.}
goto(54, STMT)	{BLOCK -> STMT.BLOCK}	54	
goto(54, ')	{BLOCK -> '.'}	55	
goto(54, VDECL)	{STMT -> VDECL.}	56	
goto(54, ASSIGN)	{STMT -> ASSIGN.semi}	57	
goto(54, if)	{STMT -> if.lparen COND rparen lbrace BLOCK rbrace ELSE}	58	
goto(54, while)	{STMT -> while.lparen COND rparen lbrace BLOCK rbrace}	59	
goto(54, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi}	2	
goto(54, id)	{ASSIGN -> id.assign RHS}	60	
goto(55, ')	{BLOCK -> ' '.}	66	{BLOCK -> ' '.}
goto(57, semi)	{STMT -> ASSIGN semi.}	67	{STMT -> ASSIGN semi.}
goto(58, lparen)	{STMT -> if lparen.COND rparen lbrace BLOCK rbrace ELSE}	68	{STMT -> if lparen.COND rparen lbrace BLOCK rbrace ELSE; COND -> .COND comp COND; COND -> .boolstr}
goto(59, lparen)	{STMT -> while lparen.COND rparen lbrace BLOCK rbrace}	69	{STMT -> while lparen.COND rparen lbrace BLOCK rbrace; COND -> .COND comp COND; COND -> .boolstr}
goto(60, assign)	{ASSIGN -> id assign.RHS}	17	
goto(61, id)	{MOREARGS -> comma vtype id.MOREARGS}	70	{MOREARGS -> comma vtype id.MOREARGS; MOREARGS -> .comma vtype id MOREARGS; MOREARGS -> .' '}
goto(63, rbrace)	{FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace.}	71	{FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace.}
goto(64, RHS)	{RETURN -> return RHS.semi}	72	{RETURN -> return RHS.semi}
goto(64, EXPR)	{RHS -> EXPR.; EXPR -> EXPR.addsub EXPR; EXPR -> EXPR.multdiv EXPR}	22	
goto(64, literal)	{RHS -> literal.}	23	
goto(64, character)	{RHS -> character.}	24	
goto(64, boolstr)	{RHS -> boolstr.}	25	
goto(64, lparen)	{EXPR -> lparen.EXPR rparen}	26	
goto(64, id)	{EXPR -> id.}	27	
goto(64, num)	{EXPR -> num.}	28	
goto(68, COND)	{STMT -> if lparen COND.rparen lbrace BLOCK rbrace ELSE; COND -> COND.comp COND}	73	{STMT -> if lparen COND.rparen lbrace BLOCK rbrace ELSE; COND -> COND.comp COND}
goto(68, boolstr)	{COND -> boolstr.}	74	{COND -> boolstr.}
goto(69, COND)	{STMT -> while lparen COND.rparen lbrace BLOCK rbrace; COND -> COND.comp COND}	75	{STMT -> while lparen COND.rparen lbrace BLOCK rbrace; COND -> COND.comp COND}
goto(69, boolstr)	{COND -> boolstr.}	74	
goto(70, MOREARGS)	{MOREARGS -> comma vtype id MOREARGS.}	76	{MOREARGS -> comma vtype id MOREARGS.}
goto(70, comma)	{MOREARGS -> comma.vtype id MOREARGS}	51	
goto(70, ')	{MOREARGS -> '.'}	52	
goto(72, semi)	{RETURN -> return RHS semi.}	77	{RETURN -> return RHS semi.}
goto(73, rparen)	{STMT -> if lparen COND rparen.lbrace BLOCK rbrace ELSE}	78	{STMT -> if lparen COND rparen.lbrace BLOCK rbrace ELSE}
goto(73, comp)	{COND -> COND comp.COND}	79	{COND -> COND comp.COND; COND -> .COND comp COND; COND -> .boolstr}
goto(75, rparen)	{STMT -> while lparen COND rparen.lbrace BLOCK rbrace}	80	{STMT -> while lparen COND rparen.lbrace BLOCK rbrace}
goto(75, comp)	{COND -> COND comp.COND}	79	
goto(78, lbrace)	{STMT -> if lparen COND rparen lbrace.BLOCK rbrace ELSE}	81	{STMT -> if lparen COND rparen lbrace.BLOCK rbrace ELSE; BLOCK -> .STMT BLOCK; BLOCK -> .' '; STMT -> .VDECL; STMT -> .ASSIGN semi; STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE; STMT -> .while lparen COND rparen lbrace BLOCK rbrace; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; ASSIGN -> .id assign RHS}
goto(79, COND)	{COND -> COND comp COND.; COND -> COND.comp COND}	82	{COND -> COND comp COND.; COND -> COND.comp COND}
goto(79, boolstr)	{COND -> boolstr.}	74	
goto(80, lbrace)	{STMT -> while lparen COND rparen lbrace.BLOCK rbrace}	83	{STMT -> while lparen COND rparen lbrace.BLOCK rbrace; BLOCK -> .STMT BLOCK; BLOCK -> .' '; STMT -> .VDECL; STMT -> .ASSIGN semi; STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE; STMT -> .while lparen COND rparen lbrace BLOCK rbrace; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; ASSIGN -> .id assign RHS}
goto(81, BLOCK)	{STMT -> if lparen COND rparen lbrace BLOCK.rbrace ELSE}	84	{STMT -> if lparen COND rparen lbrace BLOCK.rbrace ELSE}
goto(81, STMT)	{BLOCK -> STMT.BLOCK}	54	
goto(81, ')	{BLOCK -> '.'}	55	
goto(81, VDECL)	{STMT -> VDECL.}	56	
goto(81, ASSIGN)	{STMT -> ASSIGN.semi}	57	
goto(81, if)	{STMT -> if.lparen COND rparen lbrace BLOCK rbrace ELSE}	58	
goto(81, while)	{STMT -> while.lparen COND rparen lbrace BLOCK rbrace}	59	
goto(81, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi}	2	
goto(81, id)	{ASSIGN -> id.assign RHS}	60	
goto(82, comp)	{COND -> COND comp.COND}	79	
goto(83, BLOCK)	{STMT -> while lparen COND rparen lbrace BLOCK.rbrace}	85	{STMT -> while lparen COND rparen lbrace BLOCK.rbrace}
goto(83, STMT)	{BLOCK -> STMT.BLOCK}	54	
goto(83, ')	{BLOCK -> '.'}	55	
goto(83, VDECL)	{STMT -> VDECL.}	56	
goto(83, ASSIGN)	{STMT -> ASSIGN.semi}	57	
goto(83, if)	{STMT -> if.lparen COND rparen lbrace BLOCK rbrace ELSE}	58	
goto(83, while)	{STMT -> while.lparen COND rparen lbrace BLOCK rbrace}	59	
goto(83, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi}	2	
goto(83, id)	{ASSIGN -> id.assign RHS}	60	
goto(84, rbrace)	{STMT -> if lparen COND rparen lbrace BLOCK rbrace.ELSE}	86	{STMT -> if lparen COND rparen lbrace BLOCK rbrace.ELSE; ELSE -> .else lbrace BLOCK rbrace; ELSE -> .' '}
goto(85, rbrace)	{STMT -> while lparen COND rparen lbrace BLOCK rbrace.}	87	{STMT -> while lparen COND rparen lbrace BLOCK rbrace.}
goto(86, ELSE)	{STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE.}	88	{STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE.}
goto(86, else)	{ELSE -> else.lbrace BLOCK rbrace}	89	{ELSE -> else.lbrace BLOCK rbrace}
goto(86, ')	{ELSE -> '.'}	90	{ELSE -> '.'}
goto(89, lbrace)	{ELSE -> else lbrace.BLOCK rbrace}	91	{ELSE -> else lbrace.BLOCK rbrace; BLOCK -> .STMT BLOCK; BLOCK -> .' '; STMT -> .VDECL; STMT -> .ASSIGN semi; STMT -> .if lparen COND rparen lbrace BLOCK rbrace ELSE; STMT -> .while lparen COND rparen lbrace BLOCK rbrace; VDECL -> .vtype id semi; VDECL -> .vtype ASSIGN semi; ASSIGN -> .id assign RHS}
goto(90, ')	{ELSE -> ' '.}	92	{ELSE -> ' '.}
goto(91, BLOCK)	{ELSE -> else lbrace BLOCK.rbrace}	93	{ELSE -> else lbrace BLOCK.rbrace}
goto(91, STMT)	{BLOCK -> STMT.BLOCK}	54	
goto(91, ')	{BLOCK -> '.'}	55	
goto(91, VDECL)	{STMT -> VDECL.}	56	
goto(91, ASSIGN)	{STMT -> ASSIGN.semi}	57	
goto(91, if)	{STMT -> if.lparen COND rparen lbrace BLOCK rbrace ELSE}	58	
goto(91, while)	{STMT -> while.lparen COND rparen lbrace BLOCK rbrace}	59	
goto(91, vtype)	{VDECL -> vtype.id semi; VDECL -> vtype.ASSIGN semi}	2	
goto(91, id)	{ASSIGN -> id.assign RHS}	60	
goto(93, rbrace)	{ELSE -> else lbrace BLOCK rbrace.}	94	{ELSE -> else lbrace BLOCK rbrace.}

LR table

ACTION

State	'	vtype	id	semi	assign	literal	character	boolstr	addsub	multdiv	lparen	rparen	num	lbrace	rbrace	comma	if	while	comp	else	return	class	$
0		s2																					
1	s6	s7																				s8	
2			s9																				
3																							acc
4	s6	s7																				s8	
5	s6	s7																				s8	
6	s13																						
7			s14																				
8			s15																				
9				s16	s17																		
10				s18																			
11																							r_{1}
12																							r_{2}
13																							r_{3}
14				s16	s17						s19												
15														s20									
16	r_{4}	r_{4}	r_{4}														r_{4}	r_{4}				r_{4}	
17			s27			s23	s24	s25			s26		s28										
18	r_{5}	r_{5}	r_{5}														r_{5}	r_{5}				r_{5}	
19	s31	s30																					
20	s35	s7																					
21				r_{6}																			
22				r_{7}					s36	s37													
23				r_{8}																			
24				r_{9}																			
25				r_{10}																			
26			s27								s26		s28										
27				r_{14}					r_{14}	r_{14}		r_{14}											
28				r_{15}					r_{15}	r_{15}		r_{15}											
29												s39											
30			s40																				
31	s41																						
32															s42								
33	s35	s7																					
34	s35	s7																					
35	s45																						
36			s27								s26		s28										
37			s27								s26		s28										
38									s36	s37		s48											
39														s49									
40	s52															s51							
41												r_{18}											
42	r_{32}	r_{32}																				r_{32}	
43															r_{33}								
44															r_{34}								
45															r_{35}								
46				r_{11}					• s36 / ○ r_{11}	• s37 / ○ r_{11}		r_{11}											
47				r_{12}					• s36 / ○ r_{12}	• s37 / ○ r_{12}		r_{12}											
48				r_{13}					r_{13}	r_{13}		r_{13}											
49	s55	s2	s60														s58	s59					
50												r_{17}											
51		s61																					
52	s62																						
53																					s64		
54	s55	s2	s60														s58	s59					
55	s66																						
56	r_{23}	r_{23}	r_{23}														r_{23}	r_{23}					
57				s67																			
58											s68												
59											s69												
60					s17																		
61			s70																				
62												r_{20}											
63															s71								
64			s27			s23	s24	s25			s26		s28										
65															r_{21}						r_{21}		
66															r_{22}						r_{22}		
67	r_{24}	r_{24}	r_{24}														r_{24}	r_{24}					
68								s74															
69								s74															
70	s52															s51							
71	r_{16}	r_{16}																				r_{16}	
72				s77																			
73												s78							s79				
74												r_{28}							r_{28}				
75												s80							s79				
76												r_{19}											
77															r_{31}								
78														s81									
79								s74															
80														s83									
81	s55	s2	s60														s58	s59					
82												r_{27}							• s79 / ○ r_{27}				
83	s55	s2	s60														s58	s59					
84															s86								
85															s87								
86	s90																			s89			
87	r_{26}	r_{26}	r_{26}														r_{26}	r_{26}					
88	r_{25}	r_{25}	r_{25}														r_{25}	r_{25}					
89														s91									
90	s92																						
91	s55	s2	s60														s58	s59					
92	r_{30}	r_{30}	r_{30}														r_{30}	r_{30}					
93															s94								
94	r_{29}	r_{29}	r_{29}														r_{29}	r_{29}					

GOTO

State	CODE	VDECL	ASSIGN	RHS	EXPR	FDECL	ARG	MOREARGS	BLOCK	STMT	COND	ELSE	RETURN	CDECL	ODECL
0		1													
1	3	1				4								5	
2			10												
3															
4	11	1				4								5	
5	12	1				4								5	
6															
7			10												
8															
9															
10															
11															
12															
13															
14															
15															
16															
17				21	22										
18															
19							29								
20		33				34									32
21															
22															
23															
24															
25															
26					38										
27															
28															
29															
30															
31															
32															
33		33				34									43
34		33				34									44
35															
36					46										
37					47										
38															
39															
40								50							
41															
42															
43															
44															
45															
46															
47															
48															
49		56	57						53	54					
50															
51															
52															
53													63		
54		56	57						65	54					
55															
56															
57															
58															
59															
60															
61															
62															
63															
64				72	22										
65															
66															
67															
68											73				
69											75				
70								76							
71															
72															
73															
74															
75															
76															
77															
78															
79											82				
80															
81		56	57						84	54					
82															
83		56	57						85	54					
84															
85															
86												88			
87															
88															
89															
90															
91		56	57						93	54					
92															
93															
94															

Input (tokens): id + id * id

Maximum number of steps: 100

PARSE

Trace

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