T Dinesh Ram Kumar
2014A3A70302P

Sl No	Production	Semantic Rules
1	PROGRAM -> main sqo sqc STMT STMTS end	STMTS.inh = STMT.node PROGRAM.node = new Node(main, STMTS.syn)
2	STMTS -> null	STMTS.syn = STMTS.inh
3	STMTS -> STMT STMTS1	STMTS1.inh = STMT.node STMTS.syn = list(STMTS.inh, STMTS1.syn)
4	STMT -> DECL_STMT	STMT.node = DECL_STMT.node
5	STMT -> COND_STMT	STMT.node = COND_STMT.node
6	STMT -> IO_STMT	STMT.node = IO_STMT.node
7	STMT -> FUNC_DEF	STMT.node = FUNC_DEF.node
8	STMT -> FUNC_CALL semicolon	STMT.node = FUNC_CALL.node
9	STMT -> ASSIGN_STMT	STMT.node = ASSIGN_STMT.node
10	DECL_STMT -> TYPE VARLIST semicolon	DECL_STMT.node = new Node(TYPE.type, VARLIST.node)
11	TYPE -> int	TYPE.type = int
12	TYPE -> real	TYPE.type = real
13	TYPE -> string	TYPE.type = string
14	TYPE -> matrix	TYPE.type = matrix
15	VARLIST -> id MOREVARS	MOREVARS.inh = new Leaf(id, id.val) VARLIST.node = MOREVARS.syn
16	MOREVARS -> comma id MOREVARS1	MOREVARS1.inh = new Leaf(id, id.val) MOREVARS.syn = list(MOREVARS.inh, MOREVARS1.syn)
17	MOREVARS -> null	MOREVARS.syn = MOREVARS.inh
18	COND_STMT -> if op CONDITION cl STMT STMTS ELSE_STMT endif	STMTS.inh = STMT.node COND_STMT.node = new Node(if, CONDITION.node, STMTS.syn, ELSE_STMT.node)
19	CONDITION -> BOOL_EXPR	CONDITION.node = BOOL_EXPR.node
20	CONDITION -> not op CONDITION1 cl	CONDITION.node = new Node(not, CONDITION1.node)
21	CONDITION -> op CONDITION1 cl BOOL_OP op CONDITION2 cl semicolon	CONDITION.node = new Node(BOOL_OP.type, CONDITION1.node, CONDITION2.node)
22	BOOL_OP -> and	BOOL_OP.type = and
23	BOOL_OP -> or	BOOL_OP.type = or
24	BOOL_EXPR -> BOOL_OPERAND1 REL_OP BOOL_OPERAND2	BOOL_EXPR.node = new Node(REL_OP.type, BOOL_OPERAND1.node, BOOL_OPERAND2.node)
25	BOOL_OPERAND -> rnum	BOOL_OPERAND.node = new Leaf(rnum,rnum.val)
26	BOOL_OPERAND -> num	BOOL_OPERAND.node = new Leaf(num,num.val)
27	BOOL_OPERAND -> id	BOOL_OPERAND.node = new Leaf(id, id.val)
28	REL_OP -> lt	REL_OP.type = lt
29	REL_OP -> le	REL_OP.type = le
30	REL_OP -> gt	REL_OP.type = gt
31	REL_OP -> ge	REL_OP.type = ge
32	REL_OP -> eq	REL_OP.type = eq
33	REL_OP -> ne	REL_OP.type = ne
34	ELSE_STMT -> else STMT STMTS	STMTS.inh = STMT.node ELSE_STMT.node = STMTS.syn

35	ELSE_STMT -> null	ELSE_STMT.node = NULL
36	IO_STMT -> read op id cl semicolon	IO_STMT.node = new Node(read, new Leaf(id,id.val))
37	IO_STMT -> print op id cl semicolon	IO_STMT.node = new Node(print, new Leaf(id,id.val))
38	FUNC_DEF -> function sqo PARAMS1 sqc assign funid sqo PARAMS2 sqc STMT STMTS end semicolon	STMTS.inh = STMT.node FUNC_DEF.node = new Node(function, new Leaf(funid, funid.val), PARAMS1.node, PARAMS2.node, STMTS.syn)
39	PARAMS -> TYPE id MORE_PARAMS	MORE_PARAMS.inh = new Node(TYPE.type, new Leaf(id, id.val)) PARAMS.node = MORE_PARAMS.syn
40	MORE_PARAMS -> comma TYPE id MORE_PARAMS1	MORE_PARAMS1.inh = new Node(TYPE.type, new Leaf(id, id.val)) MORE_PARAMS.syn = list(MORE_PARAMS.inh, MORE_PARAMS1.syn)
41	MORE_PARAMS -> null	MORE_PARAMS.syn = MORE_PARAMS.inh
42	FUNC_CALL -> funid op ARG ARGS cl	ARGS.inh = ARG.node FUNC_CALL.syn = new Node(funcall,new Leaf(funid, funid.vale), ARGS.syn)
43	ARGS -> comma ARG ARGS1	ARGS1.inh = ARG.node ARGS.syn = list(ARGS.inh, ARGS1.syn)
44	ARGS -> null	ARGS.syn = ARGS.inh
45	ARG -> rnum	ARG.node = new Leaf(rnum, rnum.val)
46	ARG -> num	ARG.node = new Leaf(num, num.val)
47	ARG -> str	ARG.node = new Leaf(str, str.val)
48	ARG -> id MATRIX_ELEMENT	MATRIX_ELEMENT.inh = id.val ARG.node = MATRIX_ELEMENT.syn
49	ARG -> MATRIX	ARG.node = MATRIX.node
50	MATRIX_ELEMENT -> sqo num1 comma num2 sqc	MATRIX_ELEMENT.syn = new Node(MATRIX_ELEMENT.inh, new Leaf(num, num1.val), new Leaf(num, num2.val))
51	MATRIX_ELEMENT -> null	MATRIX_ELEMENT.syn = new Leaf(id, MATRIX_ELEMENT.inh)
52	MATRIX -> sqo ROW MORE_ROWS sqc	MORE_ROWS.inh = ROW.node MATRIX.node = MORE_ROWS.syn
53	ROW -> num MORE_IN_ROW	MORE_IN_ROW.inh = new Leaf(num,num.val) ROW.node = MORE_IN_ROW.syn
54	MORE_IN_ROW -> comma num MORE_IN_ROW1	MORE_IN_ROW1.inh = new Leaf(num,num.val) MORE_IN_ROW.syn = list(MORE_IN_ROW.inh, MORE_IN_ROW1.syn)
55	MORE_IN_ROW -> null	MORE_IN_ROW.syn = MORE_IN_ROW.inh
56	MORE_ROWS -> semicolon ROW MORE_ROWS1	MORE_ROWS1.inh = ROW.node MORE_ROWS.syn = list(MORE_ROWS.inh, MORE_ROWS1.syn)
57	MORE_ROWS -> null	MORE_ROWS.syn = MORE_ROWS.inh
58	ASSIGN_STMT -> id assign EXPR1 semicolon	ASSIGN_STMT.node = new Node(assign, new Leaf(id,id.val), EXPR1.node)
59	ASSIGN_STMT -> sqo VARLIST sqc assign EXPR2 semicolon	ASSIGN_STMT.node = new Node(assign, VARLIST.node, EXPR2.node)
60	EXPR2 -> size id	EXPR2.node = new Node(size, new Leaf(id,id.val))
61	EXPR2 -> FUNC_CALL	EXPR2.node = FUNC_CALL.node
62	EXPR1 -> size id	EXPR1.node = new Node(size, new Leaf(id,id.val))
63	EXPR1 -> FUNC_CALL	EXPR1.node = FUNC_CALL.node

64	EXPR1 -> ARITH_EXPR	EXPR1.node = ARITH_EXPR.node
65	ARITH_EXPR -> PROD_TERM MORE_IN_ARITH_EXPR	MORE_IN_ARITH_EXPR.inh = PROD_TERM.node ARITH_EXPR.node = MORE_IN_ARITH_EXPR.syn
66	MORE_IN_ARITH_EXPR -> plus PROD_TERM MORE_IN_ARITH_EXPR1	MORE_IN_ARITH_EXPR1.inh = new Node(plus, MORE_IN_ARITH_EXPR.inh, PROD_TERM.node) MORE_IN_ARITH_EXPR.syn = MORE_IN_ARITH_EXPR1.syn
67	MORE_IN_ARITH_EXPR -> minus PROD_TERM MORE_IN_ARITH_EXPR1	MORE_IN_ARITH_EXPR1.inh = new Node(minus, MORE_IN_ARITH_EXPR.inh, PROD_TERM.node) MORE_IN_ARITH_EXPR.syn = MORE_IN_ARITH_EXPR1.syn
68	MORE_IN_ARITH_EXPR -> null	MORE_IN_ARITH_EXPR.syn = MORE_IN_ARITH_EXPR.inh
69	PROD_TERM -> RVALUE MORE_IN_PROD	MORE_IN_PROD.inh = RVALUE.node PROD_TERM.node = MORE_IN_PROD.syn
70	MORE_IN_PROD -> mul RVALUE MORE_IN_PROD1	MORE_IN_PROD1.inh = new Node(mul, MORE_IN_PROD.inh, RVALUE.node) MORE_IN_PROD.syn = MORE_IN_PROD1.syn
71	MORE_IN_PROD -> div RVALUE MORE_IN_PROD	MORE_IN_PROD1.inh = new Node(div, MORE_IN_PROD.inh, RVALUE.node) MORE_IN_PROD.syn = MORE_IN_PROD1.syn
72	MORE_IN_PROD -> null	MORE_IN_PROD.syn = MORE_IN_PROD.inh
73	RVALUE -> op ARITH_EXPR cl	RVALUE.node = ARITH_EXPR.node
74	RVALUE -> rnum	RVALUE.node = new Leaf(rnum,rnum.val)
75	RVALUE -> num	RVALUE.node = new Leaf(num,num.val)
76	RVALUE -> str	RVALUE.node = new Leaf(str,str.val)
77	RVALUE -> id MATRIX_ELEMENT	RVALUE.node = MATRIX_ELEMENT.syn MATRIX_ELEMENT.inh = id.val
78	RVALUE -> MATRIX	RVALUE.node = MATRIX.node