Semantic Rules for AST

Batch 14: Ishan Sharma (2016B2A70773P), Sarthak Sahu (2015B5A70749P), Sanjeev Singla (2017A7PS0152P), Anirudh Garg (2017A7PS0142P)

No	Grammar Rule	AST Formation Rule
1	$	-
2	program -> moduleDeclarations otherModules1 driverModule otherModules2	program.nptr=new Node(moduleDeclarations.nptr, otherModules1.nptr, driverModule.nptr, otherModules2.nptr)
3	moduleDeclarations1 -> moduleDeclaration moduleDeclarations2	moduleDeclarations1.nptr = new Node(moduleDeclaration.nptr, moduleDeclarations2.nptr)
4	moduleDeclarations -> EPSILON	moduleDeclarations.nptr = NULL
5	moduleDeclaration -> DECLARE MODULE ID SEMICOL	STtype(ID) = MODULE moduleDeclaration.nptr = new Leaf(ID, STentry(ID))
6	otherModules1 -> module otherModules2	otherModules1.nptr = new Node(module.nptr, otherModules2.nptr)
7	otherModules -> EPSILON	otherModules.nptr = NULL
8	driverModule -> DRIVERDEF DRIVER PROGRAM DRIVERENDDEF moduleDef	driverModule.nptr = new Node(moduleDef.nptr)
9	module -> DEF MODULE ID ENDDEF TAKES INPUT SQBO input_plist SQBC SEMICOL ret moduleDef	STtype(ID) = MODULE ID.nptr = new Leaf(ID, STentry(ID)) module.nptr = new Node(ID.nptr, input_plist.nptr, ret.nptr, moduleDef.nptr)
10	ret -> RETURNS SQBO output_plist SQBC SEMICOL	ret.nptr = new Node(output_plist.nptr)
11	ret -> EPSILON	ret.nptr = NULL
12	input_plist -> ID COLON dataType input_plist_again	STtype(ID) = dataType.type ID.nptr = new Leaf(ID, STentry(ID)) input_plist.nptr = new Node(ID.nptr, dataType.nptr, input_plist_again.nptr)
13	input_plist_again1 -> COMMA ID COLON dataType input_plist_again2	STtype(ID) = dataType.type ID.nptr = new Leaf(ID, STentry(ID)) input_plist_again1.nptr = new Node(ID.nptr, dataType.nptr, input_plist_again2.nptr)
14	input_plist_again -> EPSILON	input_plist_again.nptr = NULL
15	output_plist -> ID COLON type output_plist_again	STtype(ID) = type.type ID.nptr = new Leaf(ID, STentry(ID)) output_plist.nptr = new Node(ID.nptr, type.nptr, output_plist_again.nptr)
16	output_plist_again1 -> COMMA ID COLON type output_plist_again2	STtype(ID) = type.type ID.nptr = new Leaf(ID, STentry(ID)) output_plist_again1.nptr = new Node(ID.nptr, type.nptr, output_plist_again2.nptr)
17	output_plist_again -> EPSILON	output_plist_again.nptr = NULL
18	dataType -> INTEGER	dataType.type = Integer
19	dataType -> REAL	dataType.type = Real
20	dataType -> BOOLEAN	dataType.type = Boolean
21	dataType -> ARRAY SQBO range_arrays SQBC OF type	dataType.nptr = new Node(range_arrays.nptr, type.nptr) dataType.type = type.type
22	range_arrays -> index1 RANGEOP index2	range_arrays.nptr = new Node(index1.nptr, index2.nptr)
23	type -> INTEGER	type.type = Integer
24	type -> REAL	type.type = Real
25	type -> BOOLEAN	type.type = Boolean
26	moduleDef -> START statements END	moduleDef.nptr = statements.nptr
27	statements1 -> statement statements2	statements1.nptr = new Node(statement.nptr, statements2.nptr)
28	statements ->EPSILON	statements.nptr = NULL
29	statement-> ioStmt	statement.nptr = ioStmt.nptr
30	statement ->simpleStmt	statement.nptr = simpleStmt.nptr
31	statement-> declareStmt	statement.nptr = declareStmt.nptr
32	statement-> conditionalStmt	statement.nptr = condionalStmt.nptr
33	statement ->iterativeStmt	statement.nptr = iterativeStmt.nptr
34	ioStmt ->GET_VALUE BO ID BC SEMICOL	ID.nptr = new Leaf(ID, STentry(ID)) ioStmt.nptr = ID.nptr
35	ioStmt -> PRINT BO var BC SEMICOL	ioStmt.nptr = var.nptr
36	var -> var_id_num	var.nptr = var_id_num.nptr var.type = var_id_num.type
37	var-> booleanConstants	var.nptr = booleanConstants.nptr var.type = booleanConstants.type
38	var_id_num -> ID whichId	ID.nptr = new Leaf(ID, STentry(ID)) var_id_num.nptr = new Node(ID.nptr, whichId.nptr) var_id_num.type = ID.type
39	var_id_num -> NUM	NUM.nptr = new Leaf(NUM, NUM.value) var_id_num.nptr = NUM.nptr var_id_num.type = Integer
40	var_id_num-> RNUM	RNUM.nptr = new Leaf(RNUM, RNUM.value) var_id_num.nptr = RNUM.nptr var_id_num.type = Real
41	whichId -> SQBO index SQBC	whichId.nptr = index.nptr whichId.type = index.type
42	whichId -> EPSILON	whichId.nptr = NULL
43	simpleStmt -> assignmentStm	simpleStmt.nptr = assignmentStm.nptr
44	simpleStmt -> moduleReuseStmt	simpleStmt.nptr = moduleReuseStmt.nptr
45	assignmentStmt -> ID whichStmt	ID.nptr = new Leaf(ID, STentry(ID)) assignmentStmt.nptr = new Node(ID.nptr, whichStmt.nptr)
46	whichStmt -> lvalueIDStmt	whichStmt.nptr = lvalueIDStmt.nptr whichStmt.type = lvalueIDStmt.type
47	whichStmt -> lvalueARRStmt	whichStmt.nptr = lvalueARRStmt.nptr whichStmt.type = lvalueARRStmt.type
48	lvalueIDStmt -> ASSIGNOP expression SEMICOL	lvalueIDStmt.nptr = expression.nptr lvalueIDStmt.type = expression.type
49	lvalueARRStmt -> SQBO index SQBC ASSIGNOP expression SEMICOL	lvalueARRStmt.nptr = new Node(index.nptr, expression.nptr) lvalueARRStmt.type = expression.type
50	index -> NUM	NUM.nptr = new Leaf(NUM, NUM.value) index.nptr = NUM.nptr index.type = Integer
51	index -> ID	ID.nptr = new Leaf(ID, STentry(ID)) index.nptr = ID.nptr index.type = ID.type
52	moduleReuseStmt -> optional USE MODULE ID WITH PARAMETERS idList SEMICOL	ID.nptr = new Leaf(ID, STentry(ID)) moduleReuseStmt.nptr = new Node(optional.nptr, ID.nptr, idList.nptr)
53	optional -> SQBO idList SQBC ASSIGNOP	optional.nptr = idList.nptr
54	optional -> EPSILON	optional.nptr = NULL
55	idList -> ID idList_again	ID.nptr = new Leaf(ID, STentry(ID)) idList.nptr = new Node(ID.nptr, idList_again.nptr)
56	idList_again1 -> COMMA ID idList_again2	ID.nptr = new Leaf(ID, STentry(ID)) idList_again 1.nptr = new Node(ID.nptr, idList_again 2.nptr)
57	idList_again -> EPSILON	idList_again .nptr = NULL
58	expression -> arithmeticOrBooleanExpr	expression.nptr = arithmeticOrBooleanExpr.nptr
59	expression -> unary	expression.nptr = unary.nptr
60	unary -> unary_op new_NT	unary.nptr = new Node(unary_op.nptr, new_NT.nptr)
61	unary_op -> PLUS	PLUS.nptr = new Leaf(PLUS, '+') unary_op.nptr = PLUS.nptr
62	unary_op -> MINUS	MINUS.nptr = new Leaf(MINUS, '-') unary_op.nptr = MINUS.nptr
63	new_NT -> BO arithmeticExpr BC	new_NT.nptr = arithmeticExpr.nptr new_NT.type = arithmeticExpr.type
64	new_NT -> var_id_num	new_NT.nptr = var_id_num.nptr new_NT.type = var_id_num.type

65	arithmeticOrBooleanExpr -> recTerm arithmeticOrBooleanExpr_again	arithmeticOrBooleanExpr.nptr = new Node(recTerm.nptr,arithmeticOrBooleanExpr_again.nptr)
66	arithmeticOrBooleanExpr_again1 -> logicalOp recTerm arithmeticOrBooleanExpr_again2	arithmeticOrBooleanExpr_again1.nptr = new Node(logicalOp.nptr,recTerm.nptr,arithmeticOrBooleanExpr_again2.nptr)
67	arithmeticOrBooleanExpr_again -> EPSILON	arithmeticOrBooleanExpr_again.nptr = NULL
68	recTerm -> arithmeticExpr recTerm_again	recTerm.nptr = new Node(arithmeticExpr.nptr,recTerm_again.nptr)
69	recTerm -> booleanConstants recTerm_again	recTerm.nptr = new Node(booleanConstants.nptr,recTerm_again.nptr)
70	recTerm_again1 -> relationalOp arithmeticExpr recTerm_again2	recTerm_again1.nptr = new Node(relationalOp.nptr,arithmeticExpr.nptr,recTerm_again2.nptr)
71	recTerm_again -> EPSILON	recTerm_again.nptr = NULL
72	arithmeticExpr -> term arithmeticExpr_again	arithmeticExpr.nptr = new Node(term.nptr, arithmeticExpr_again.nptr) arithmeticExpr.type = term.type
73	arithmeticExpr_again1 -> prec2_op term arithmeticExpr_again2	arithmeticExpr_again1.nptr = new Node(prec2_op.nptr,term.nptr,arithmeticExpr_again2.nptr)
74	arithmeticExpr_again -> EPSILON	arithmeticExpr_again.nptr = NULL
75	term -> factor term_again	term.nptr = new Node(factor.nptr,term_again.nptr) term.type = factor.type
76	term_again1 -> prec1_op factor term_again2	term_again1.nptr = new Node(prec1_op.nptr,factor.nptr,term_again2.nptr)
77	term_again -> EPSILON	term_again.nptr = NULL
78	factor -> BO arithmeticOrBooleanExpr BC	factor.nptr = arithmeticOrBooleanExpr.nptr factor.type = arithmeticOrBooleanExpr.type
79	factor -> var_id_num	factor.nptr = var_id_num.nptr factor.type = var_id_num.type
80	booleanConstants -> TRUE	TRUE.nptr = new Leaf(TRUE, 'true') booleanConstants.nptr =TRUE.nptr
81	booleanConstants -> FALSE	FALSE.nptr = new Leaf(FALSE, 'false') booleanConstants.nptr = FALSE.nptr
82	prec2_op -> PLUS	PLUS.nptr = new Leaf(PLUS, '+') prec2_op.nptr = PLUS.nptr
83	prec2_op -> MINUS	MINUS.nptr = new Leaf(MINUS, '-') prec2_op.nptr = MINUS.nptr
84	prec1_op -> MUL	MUL.nptr = new Leaf(MUL, '*') prec1_op.nptr = MUL.nptr
85	prec1_op -> DIV	DIV.nptr = new Leaf(DIV, '/') prec1_op.nptr = DIV.nptr
86	logicalOp -> AND	AND.nptr = new Leaf(AND, 'AND') logicalOp.nptr = AND.nptr
87	logicalOp -> OR	OR.nptr = new Leaf(OR, 'OR') logicalOp.nptr = OR.nptr
88	relationalOp -> LT	LT.nptr = new Leaf(LT, '<') relationalOp.nptr = LT.nptr
89	relationalOp -> LE	LE.nptr = new Leaf(LE, '<=') relationalOp.nptr = LE.nptr
90	relationalOp -> GT	GT.nptr = new Leaf(GT, '>') relationalOp.nptr = GT.nptr
91	relationalOp -> GE	GE.nptr = new Leaf(GE, '>=') relationalOp.nptr = GE.nptr
92	relationalOp -> EQ	EQ.nptr = new Leaf(EQ, '==') relationalOp.nptr = EQ.nptr
93	relationalOp -> NE	NE.nptr = new Leaf(NE, '!=') relationalOp.nptr = NE.nptr
94	declareStmt -> DECLARE idList COLON dataType SEMICOL	declareStmt.nptr = new Node(idList.nptr, dataType.nptr) idList.type = dataType.type
95	conditionalStmt -> SWITCH BO ID BC START caseStmts default END	conditionalStmt.nptr = new Node(caseStmts.nptr, default.nptr)
96	caseStmts -> CASE value COLON statements BREAK SEMICOL caseStmts_again	caseStmts.nptr = new Node(value.nptr, statements.nptr, caseStmts_again.nptr)
97	caseStmts_again1 -> CASE value COLON statements BREAK SEMICOL caseStmts_again2	caseStmts_again1.nptr = new Node(value.nptr, statements.nptr, caseStmts_again2.nptr)
98	caseStmts_again -> EPSILON	caseStmts_again.nptr = NULL
99	value -> NUM	NUM.nptr = new Leaf(NUM, NUM.value) value.type = Integer
100	value -> booleanConstants	value.nptr = booleanConstants.nptr value.type = booleanConstants.type
101	default -> DEFAULT COLON statements BREAK SEMICOL	default.nptr = <statements>.nptr
102	default -> EPSILON	default.nptr = NULL
103	iterativeStmt -> FOR BO ID IN range BC START statements END	ID.nptr = new Leaf(ID, STentry(ID)) iterativeStmt.nptr = new Node(ID.nptr, range.nptr, statements.nptr)
104	iterativeStmt -> WHILE BO arithmeticOrBooleanExpr BC START statements END	iterativeStmt.nptr = new Node(arithmeticOrBooleanExpr.nptr, statements.nptr) arithmeticOrBooleanExpr.type = Boolean
105	range -> NUM1 RANGEOP NUM2	NUM1.nptr = new Leaf(NUM1, NUM1.value) NUM2.nptr = new Leaf(NUM2, NUM2.value) range.nptr = new Node(NUM1.nptr, NUM2.nptr)