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

	Batch 14: Ishan Sharma (2016B2A70773P), Sarthak Sahu (2015B5A707	(49P), Sanjeev Singla (2017A7PS0152P), Anirudh Garg (2017A7PS0142P)
No	Grammar Rule	AST Formation Rule
3	program -> moduleDeclarations otherModules1 driverModule otherModules2 moduleDeclarations1 -> moduleDeclaration moduleDeclarations2	program.nptr=new Node(moduleDeclarations.nptr, otherModules1.nptr, driverModule.nptr, otherModules2.nptr) 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)
8	otherModules -> EPSILON driverModule -> DRIVERDEF DRIVER PROGRAM DRIVERENDDEF moduleDef	otherModules.nptr = NULL driverModule.nptr = new Node(moduleDef.nptr)
	module -> DEF MODULE ID ENDDEF TAKES INPUT SQBO input plist SQBC SEMICOL ret	STtype(ID) = MODULE
9	moduleDef	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 STtype(ID) = dataType.type
12	input_plist -> ID COLON dataType input_plist_again	ID.nptr = new Leaf(ID, STentry(ID))
		input_plist.nptr = new Node(ID.nptr, dataType.nptr, input_plist_again.nptr) STtype(ID) = dataType.type
13	input_plist_again1 -> COMMA ID COLON dataType input_plist_again2	ID.nptr = new Leaf(ID, STentry(ID))
14	input plist again -> EPSILON	input_plist_again1.nptr = new Node(ID.nptr, dataType.nptr, input_plist_again2.nptr) input_plist_again.nptr = NULL
		STtype(ID) = type.type
15	output_plist -> ID COLON type output_plist_again	ID.nptr = new Leaf(ID, STentry(ID)) output _plist.nptr = new Node(ID.nptr, type.nptr, output _plist_again.nptr)
		STtype(ID) = type.type
16	output_plist_again1 -> COMMA ID COLON type output_plist_again2	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 19	dataType -> INTEGER dataType -> REAL	dataType.type = Integer dataType.type = Real
	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)
	type -> INTEGER type -> REAL	type.type = Integer type.type = Real
25	type -> BOOLEAN	type.type = Boolean
26	moduleDef -> START statements END statements1 -> statement statements2	moduleDef.nptr = statements.nptr statements1.nptr = new Node(statement.nptr, statements2.nptr)
28	statements ->EPSILON	statements.nptr = NULL
30	statement-> ioStmt statement ->simpleStmt	statement.nptr = ioStmt.nptr statement.nptr = simpleStmt.nptr
31	statement-> declareStmt	statement.nptr = declareStmt.nptr
32	statement-> conditionalStmt statement -> iterativeStmt	statement.nptr = condionalStmt.nptr statement.nptr = iterativeStmt.nptr
34	ioStmt ->GET_VALUE BO ID BC SEMICOL	ID.nptr = new Leaf(ID, STentry(ID))
35	ioStmt -> PRINT BO var BC SEMICOL	ioStmt.nptr = ID.nptr ioStmt.nptr = var.nptr
36	var -> var id num	var.nptr = var_id_num.nptr
-		var.type = var_id_num.type var.nptr = booleanConstants.nptr
37	var-> booleanConstants	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 NUM.nptr = new Leaf(NUM, NUM.value)
39	var_id_num -> NUM	var_id_num.nptr = NUM.nptr
		var_id_num.type = Integer RNUM.nptr = new Leaf(RNUM, RNUM.value)
40	var_id_num-> RNUM	var_id_num.nptr = RNUM.nptr
		var_id_num.type = Real whichId.nptr = index.nptr
41	whichId -> SQBO index SQBC	whichId.type = index.type
	whichId -> EPSILON simpleStmt -> assignmentStm	whichId.nptr = NULL simpleStmt.nptr = assignmentStm.nptr
	simpleStmt > assignmentour 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 -> IvalueIDStmt	whichStmt.nptr = lvalueIDStmt.nptr
		whichStmt.type = IvalueIDStmt.type whichStmt.nptr = IvalueARRStmt.nptr
47	whichStmt -> IvalueARRStmt	whichStmt.type = IvalueARRStmt.type
48	IvalueIDStmt -> ASSIGNOP expression SEMICOL	lvalueIDStmt.nptr = expression.nptr lvalueIDStmt.type = expression.type
49	valueARRStmt -> SQBO index SQBC ASSIGNOP expression SEMICOL	lvalueARRStmt.nptr = new Node(index.nptr, expression.nptr)
50	index -> NUM	IvalueARRStmt.type = expression.type NUM.nptr = new Leaf(NUM, NUM.value) index.nptr = NUM.nptr
51	index -> ID	index.type = Integer ID.nptr = new Leaf(ID, STentry(ID)) index.nptr = ID.nptr
52	moduleReuseStmt -> optional USE MODULE ID WITH PARAMETERS idList SEMICOL	index.type = ID.type ID.nptr = new Leaf(ID, STentry(ID))
	optional -> SQBO idList SQBC ASSIGNOP	moduleReuseStmt.nptr = new Node(optional.nptr, ID.nptr, idList.nptr)
53 54	optional -> SQBO idList SQBC ASSIGNOP optional -> EPSILON	optional.nptr = idList.nptr 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))
57	idList_againt -> COMMA ID IdList_again2 idList_again -> EPSILON	idList_again 1.nptr = new Node(ID.nptr, idList_again 2.nptr) idList_again .nptr = NULL
58	expression -> arithmeticOrBooleanExpr	expression.nptr = arithmeticOrBooleanExpr.nptr
59 60	expression -> unary unary -> unary op new NT	expression.nptr = unary.nptr unary.nptr = new Node(unary op.nptr, new NT.nptr)
61	unary -> unary op new N1 unary_op -> PLUS	PLUS.nptr = new Leaf(PLUS, '+')
-		unary_op.nptr = PLUS.nptr MINUS.nptr = new Leaf(MINUS, '-')
62	unary_op -> 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)
			arithmeticOrBooleanExpr again1.nptr = new Node(logicalOp.nptr,recTerm.nptr,arithmeticOrBooleanExpr again2.nptr)
10 Peter ma- Josebane Constants rea Term again april Peter may a pril Peter may a	67		
10 Peter ma- Josebane Constants rea Term again april Peter may a pril Peter may a	68	recTerm -> arithmeticExpr recTerm again	recTerm.nptr = new Node(arithmeticExpr.nptr.recTerm again.nptr)
The Certer angain 1 - selstood parthemeticky received again 2 Section and part Section S			
Part			
minmonecting rolling imminonectory, again and minimonectory, again and minimonectory again of person and minimonectory again of person and minimonectory again of person and minimonectory again and minimonectory again of person again aga	71	recTerm again -> EPSILON	recTerm again.nptr = NULL
authenticity type - sem type - influenticity gain 2 prec appeal appea		a an a a an a	arithmeticExpr.nptr = new Node(term.nptr. arithmeticExpr again.nptr)
33	72	arithmeticExpr -> term arithmeticExpr_again	
74 antimetic-Exp. again op the NULL 75 sum - Since to run again term up to en Nondiction run put up to many put to the run again put to the run type - factor Pype - artifluents/Office/GenElspraper 78 factor Pype - quil pum factor pype - artifluents/Office/GenElspraper 80 booleanConstants > TRUE factor pype - val. (i. pum pype - factor Pype - factor Pype - pype - pype - factor Pype - pype	73	arithmeticExpr again1 -> prec2 on term arithmeticExpr again2	
175 term - factor term , again term , again term , again , ag	74		
Section			
16	75	term -> factor term_again	
177 term again = FESLON term again appr = NULL	76	term again1 -> prec1 on factor term again?	
Ractor SO arithmetic Of Boolean Expt BC Ractor per arithmetic Of Boolean Expt per			
Section Sect		_ <u>- ~</u>	
	78	factor -> BO arithmeticOrBooleanExpr BC	
Section Sect	79	factor -> var_id_num	
Bay	80	booleanConstants -> TRUE	
Society Soci			
RELUS apt = new Leaf(RLUS, ") prec2_op > PLUS prec2_op = MINUS prec2_op = MINUS prec1_op > MUL prec1_op = MUL prec1_op > MUL prec1_op > MUL prec1_op = MUL prec2_op = MUL prec1_op = MUL prec1_op = MUL prec1_op = M	81	booleanConstants -> FALSE	
1.			
BX prec2_op > MINUS MINUS spr = new LeafMINUS, ") prec2_op and "miNUS aptr new LeafMINUS, "Dinameter new LeafMINUS,	82	prec2_op -> PLUS	
Section Peec Op > MIUL			
MUL. ppt = new Leaf(M.L., **) precl _op. > DIV	83	prec2_op -> MINUS	
Section Sect			
BS prec _op > DIV DIV ppt = new Leaf(DIV, '7') prec _op prec > DIV ppt = DIV ppt DI	84	precl_op -> MUL	
Section Sect			
AND nptr = new Leaf(AND, 'AND')	85	precl_op -> DIV	
Sociation Soci			
ST logicalOp > OR logicalOp = Ne Leaf(OR, 'OR')	86	logicalOp -> AND	
Second S			
SecretarionalOp > LT	87	logicalOp -> OR	
RelationalOp > LE			
Relational Op > LE	88	relationalOp -> LT	
relationalOp apt = LE.nptr relationalOp > GT relationalOp > GT relationalOp > GE relationalOp			
90 relationalOp > GT GT nptr = new Leaf(GT, >>) relationalOp pr = GT nptr GE nptr = new Leaf(GE, >=) relationalOp > EQ relationalOp > NE PeationalOp > NE relationalOp	89	relationalOp -> LE	
PositionalOp => CIT relationalOp aptr = GT.nptr			
91 relationalOp > GE GE.nptr = new Leaf(EC, '>=') relationalOp.nptr = GE.nptr 92 relationalOp > EQ EQ.nptr = new Leaf(EQ, '==') relationalOp.nptr = EQ.nptr 93 relationalOp > NE NE.nptr = new Leaf(EQ, '==') relationalOp.nptr = EQ.nptr 94 declareStmt > DECLARE idList COLON dataType SEMICOL declareStmt.nptr = new Node(idList.nptr, dataType.nptr) 95 declareStmt > DECLARE idList COLON dataType SEMICOL declareStmt.nptr = new Node(idList.nptr, dataType.nptr) 96 declareStmt > SWITCH BO ID BC START caseStmts default END conditionalStmt = SWITCH BO ID BC START caseStmts default END conditionalStmt = new Node(value.nptr, statements.nptr, caseStmts again.nptr 96 caseStmts = CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts again = New Node(value.nptr, statements.nptr, caseStmts again.nptr 97 caseStmts again = PSILON caseStmts again = PSILON caseStmts again = PSILON caseStmts again = PSILON NUM.nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr = NULL 100 value = NUM value NUM value value value = NUM value	90	relationalOp -> GT	
relationalOp > GE relationalOp > FQ relationalOp > EQ relationalOp > EQ relationalOp = EQ.nptr = new Leaf(EQ, '==') relationalOp = EQ.nptr = new Leaf(EQ, '==') relationalOp = EQ.nptr = NE.nptr = N			GE note = now Loof (GE '\=')
P2 relationalOp -> EQ EQ.nptr = new Leaf(EQ, ':=-') relationalOp prir = Q.nptr	91	relationalOp -> GE	
relationalOp.phr = EQ.nptr PationalOp > NE relationalOp > NE relationalOp > NE RelationalOp > NE relationalOp = new Leaf(NE, '!=') relationalOp nptr = new Leaf(NE, '!=') relationalOp nptr = new Node(idList.nptr, dataType.nptr) declareStmt. > DECLARE idList COLON dataType SEMICOL declareStmt. pptr = new Node(idList.nptr, dataType.nptr) declareStmt. pptr = new Node(caseStmts.nptr, default.nptr) conditionalStmt. > SWITCH BO ID BC START caseStmts default END conditionalStmt. > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts. > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts again.nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr) caseStmts again.pptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr) value > NUM value > NUM NUM.nptr = new Leaf(NUM, NUM.value) value.ype = Integer value.type = Integer value.type = booleanConstants.nptr value.type = booleanConstants.type default.ppt = Statements-nptr default.ppt = NULL ld default.ppt = NULL ld default.ppt = NULL ld default.ppt = NULL ld default.ppt = booleanConstants.type ld default.ppt = booleanConstants.nptr value.type = booleanConstants.type ld default.ppt = NULL lD nptr = new Leaf(ID, STentry(ID)) literativeStmt > FOR BO ID IN range BC START statements END literativeStmt.nptr = new Node(cilh.nptr, range.nptr, statements.nptr) iterativeStmt.nptr			
relationalOp > NE relationalOp > NE NE.nptr = new Leaf(NE, '!=') relationalOp.pptr = NE.nptr declareStmt > DECLARE idList COLON dataType SEMICOL declareStmt.pptr = new Node(idList.nptr, dataType.nptr) idList.type = dataType.type conditionalStmt.> SWITCH BO ID BC START caseStmts default END caseStmts > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts = caseStmts again > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts again > EPSILON caseStmts again > EPSILON NUM.nptr = new Leaf(NUM, NUM.value) value > NUM value > booleanConstants value : booleanConstants default > DEFAULT COLON statements BREAK SEMICOL default.nptr = Statements.nptr default.pptr = Statements.nptr default.pptr = NULL ID.nptr = new Leaf(ID, STentry(ID)) iterativeStmt > FOR BO ID IN range BC START statements END iterativeStmt > FOR BO can Expr.nptr, statements.nptr iterativeStmt > WILL BO arithmeticOrBooleanExpr.nptr, statements.nptr) NUM.nptr = new Leaf(ID, STentry(ID)) iterativeStmt.pptr = new Node(calue.nptr, statements.nptr) iterativeStmt.nptr = new Node(caseStmts again.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new Node(dD.nptr, range.nptr, statements.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new Node(caseStmts.nptr) iterativeStmt.nptr = new	92	relationalOp -> EQ	
relationalOp - NE relationalOp - Net - Net - Note(in the interpretor) declareStmt.nptr = new Node(idList.nptr, dataType.nptr) idList.type = dataType.type conditionalStmt - SWITCH BO ID BC START caseStmts default END caseStmts - CASE value COLON statements BREAK SEMICOL caseStmts again - CaseStmts again - Nptr = new Node(value nptr, statements.nptr, caseStmts again.nptr) asseStmts again - Nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr) asseStmts again - Nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr) asseStmts again - Nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr) caseStmts again - Nptr = new Leaf(NIM, NIM.value) value -> NUM NUM.nptr = new Leaf(NIM, NIM.value) value.type = Integer value.type = looleanConstants.type default -> DEFAULT COLON statements BREAK SEMICOL default.ptr = Statements-nptr value.type = booleanConstants.type default.ptr = NULL log default -> DEFAULT COLON statements BREAK SEMICOL default.nptr = NULL log iterativeStmt -> FOR BO ID IN range BC START statements END iterativeStmt.nptr = new Node(arithmeticOrBooleanExpr.nptr, statements.nptr) iterativeStm			
declareStmt.ptr = new Node(idList.nptr, dataType.nptr) idList.type = dataType.type conditionalStmt > SWITCH BO ID BC START caseStmts default END conditionalStmt > SWITCH BO ID BC START caseStmts default END conditionalStmt.ptr = new Node(caseStmts.nptr, default.nptr) caseStmts > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts again CaseStmts again CaseStmts again CaseStmts again CaseStmts again CaseStmts again Details CaseStmts again Details CaseStmts again Details	93	relationalOp -> NE	
declarestint DECLARE titles COLON data type selfitCOL 55 conditionalStmt > SWITCH BO ID BC START caseStmts default END 96 caseStmts > CASE value COLON statements BREAK SEMICOL caseStmts again 97 caseStmts > CASE value COLON statements BREAK SEMICOL caseStmts again 98 caseStmts again > CASE value COLON statements BREAK SEMICOL caseStmts 99 value > NUM 99 value > NUM 100 value > booleanConstants 101 default > DEFAULT COLON statements BREAK SEMICOL 102 default > DEFAULT COLON statements BREAK SEMICOL 103 default > DEFAULT COLON statements BREAK SEMICOL 104 default > DEFAULT COLON statements BREAK SEMICOL 105 terativeStmt > OR BO ID IN range BC START statements END 106 terativeStmt > WILLE BO arithmeticOrBooleanExpr.nptr, statements.nptr 107 terativeStmt > WILLE BO arithmeticOrBooleanExpr.nptr, statements.nptr 108 terativeStmt > WILLE BO arithmeticOrBooleanExpr.nptr, statements.nptr 109 terativeStmt > WILLE BO arithmeticOrBooleanExpr.nptr, statements.nptr 100 terativeStmt > WILLE BO arithmeticOrBooleanExpr.nptr, statements.nptr 104 terativeStmt > WILLE BO arithmeticOrBooleanExpr.nptr, statements.nptr 105 terativeStmt > WILLE BO AND			
95 conditionalStmt -> SWITCH BO ID BC START caseStmts default END conditionalStmt.nptr = new Node(caseStmts.nptr, default.nptr)	94	declareStmt -> DECLARE idList COLON dataType SEMICOL	
96 caseStmts > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr ocaseStmts again > CASE value COLON statements BREAK SEMICOL caseStmts again caseStmts again.nptr = new Node(value.nptr, statements.nptr, caseStmts again.nptr ocaseStmts again.nptr = NULL ocaseStmts again.nptr = NULL NUM.nptr = new Leaf(NUM, NUM.value) value > NUM NUM.value value.type = Integer value.type = booleanConstants.nptr value.type = booleanConstants.nptr value.type = booleanConstants.type default > DEFAULT COLON statements BREAK SEMICOL default.nptr = vstatements.nptr default.nptr = vstatements.nptr default.nptr = NULL default.nptr = NULL lib.nptr = new Leaf(ID, STentry(ID)) lib.nptr = new Leaf(ID, STentry(ID)) literativeStmt > FOR BO ID IN range BC START statements END literativeStmt.nptr = new Node(arithmeticOrBooleanExpr.nptr, statements.nptr) literativeStmt.nptr = new Leaf(NUM1, NUM1.value) NUM2.nptr = new Leaf(NUM2, NUM2, value) NUM2.nptr = new Leaf(NUM2, NUM2, num2, value) NUM2.nptr = new Leaf(NUM2, NUM2, value) NUM2.nptr = new Leaf(NUM2, NUM2, value) NUM2.nptr = new Leaf(NUM2, N	05	conditionalStmt > SWITCH BO ID BC START acceStmts default END	
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 NUM.nptr = new Leaf(NUM, NUM.value) value -> NUM value -> booleanConstants value -> booleanConstants.nptr value -> booleanConstants terativeStmt -> EPSILON iterativeStmt -> FOR BO ID IN range BC START statements END iterativeStmt -> WHILE BO arithmeticOrBooleanExpr BC START statements END iterativeStmt.nptr = new Node(arithmeticOrBooleanExpr.nptr, statements.nptr) arithmeticOrBooleanExpr.nptr, statements.nptr) iterativeStmt.nptr = new Node(arithmeticOrBooleanExpr.nptr, statements.nptr) itera			
99 value -> NUM 100 value -> booleanConstants 100 value -> booleanConstants 101 default -> DEFAULT COLON statements BREAK SEMICOL 102 default -> DEFAULT COLON statements BREAK SEMICOL 103 iterativeStmt -> FOR BO ID IN range BC START statements END 104 iterativeStmt -> WHILE BO arithmeticOrBooleanExpr BC START statements END 105 range -> NUM1 RANGEOP NUM2 106 NUM1.nptr = new Leaf(NUM1, NUM1.value) 107 NUM2.nptr = new Leaf(NUM1, NUM1.value) 108 NUM2.nptr = new Leaf(NUM1, NUM1.value) 109 NUM2.nptr = new Leaf(NUM1, NUM1.value) 100 NUM2.nptr = new Leaf(NUM1, NUM1.value)			
value -> NUM value -> NUM value -> NUM value -> NUM value -> NUMI RANGEOP NUM2 value -> NUM2 value value -> NuM3 value value -> NuM4 value value ->		_ - v	
100 value -> booleanConstants value.ptr = booleanConstants.nptr value.type = booleanConstants.nptr value.type = booleanConstants.type	99	value -> NUM	
Value -> booleanConstants value_type = booleanConstants_type			
101 default -> DEFAULT COLON statements BREAK SEMICOL default.nptr = <statements>.nptr </statements>	100	value -> booleanConstants	
102 default -> EPSILON default.nptr = NULL 103 default -> EPSILON default.nptr = NULL 104 default -> FOR BO ID IN range BC START statements END ID.nptr = new Leaf(ID, STentry(ID))	101	default > DEFAULT COLON statements DDFAV SEMICOL	
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)			
iterativeStmt -> FOR BO ID IN range BC START statements END iterativeStmt.nptr = new Node(iD.nptr, range.nptr, statements.nptr)	102	uciaun ~ El SILON	
IterativeStmtptr = new Node(ILI.nptr, range.nptr, statements.nptr)	103	iterativeStmt -> FOR BO ID IN range BC START statements END	
IterativeStmt -> WFILE BO arithmeticOrBooleanExpr BC START statements END arithmeticOrBooleanExpr.type = Boolean NUM1.nptr = new LeafNUM1, NUM1.value) NUM2.nptr = new LeafNUM1, NUM2.value) NUM2.nptr = new LeafNUM2, NUM2.value)		•	
NUM1.nptr = new Leaf(NUM1, NUM1.value) NUM2.nptr = new Leaf(NUM2, NUM2.value)	104	iterativeStmt -> WHILE BO arithmeticOrBooleanExpr BC START statements END	
105 range -> NUM1 RANGEOP NUM2 NUM2.nptr = new Leaf(NUM2, NUM2.value)		-	
	105	NUMI PANCEODNUMO	NUMI.npir = new Leat(NUMI, NUMI.value)
range.nptr = new Node(NUM1.nptr, NUM2.nptr)	105	Italige -> NUMI KANGEOP NUMZ	
			pange.npu – new roue(NUM1.nptr, NUM2.nptr)