	Grammar Rule	Semantic Rule

1	<program'> <program> $	//bottom - up
		<program'>.addr = <program>.addr
		freeNode(<program>)
		freeNode($)

2	<program> <moduleD eclarations><otherModules>_1 <dr	// bottom up
		<program>.addr = createNode("program",<moduleDeclarations>.syn_list,<otherModules>_1.syn_list,<driverModule>.addr,
		<otherModules>_2.syn_list)
		freeNode(<moduleDeclarations>)
		freeNode(<otherModules>_1)
		freeNode(<driverModule>)
		freeNode(<otherModules>_2)

3	<moduleDeclarations> <moduleDeclaration> <moduleDec	// INITIALISE <moduleDeclarations>.inh_list to empty list
		
		// bottom up
		<moduleDeclarations>.syn_list = <moduleDeclarations>_1.syn_list
		insertAtBeg(<moduleDeclarations>_1.syn_list,<moduleDeclaration>.addr)
		freeNode(<moduleDeclarations>_1)

4	<moduleDeclarations> epsilon	// INITIALISE <moduleDeclarations>.inh_list to empty list
		freeNode(epsilon)

5	<moduleDeclaration> DECLARE MODULE ID SEMICOL	//<moduleDeclaration>.addr = createNode("moduleDeclaration",ID.addr)
		<moduleDeclaration>.addr = ID.addr
		freeNode(DECLARE)
		freeNode(MODULE)
		freeNode(SEMICOL)

6	<otherModules> <module> <otherModules>_1	// INITIALISE <otherModules>.inh_list to empty list
		
		// bottom up
		<otherModules>.syn_list = <otherModules>_1.syn_list
		insertAtBeg(<otherModules>_1.syn_list,<module>.addr)
		freeNode(<otherModules>_1)
		//freeNode(<module>)

7	<otherModules> epsilon	// INITIALISE <otherModules>.inh_list to empty list
		freeNode(epsilon)

8	<driverModule> DRIVERDEF DRIVER PROGRAM DRIVEREN	<driverModule>.addr = createNode("driverModule",<moduleDef>.addr) freeNode(<moduleDef>)

9	<module> DEF MODULE ID ENDDEF TAKES INPUT SQBO <i	<module>.addr = createNode("module",<input_plist>.syn_list,<ret>.addr,<moduleDef>.addr)
		freeNode(<input_plist>)
		freeNode(<ret>)
		freeNode(<moduleDef>)

10	<ret> RETURNS SQBO <output_plist> SQBC SEMICOL	<ret>.addr = <output_plist>.syn_list free(<output_plist>)
11	<ret> epsilon	<ret>.addr = NULL free(epsilon)

12	<input_plist> ID COLON <dataType> <N1>	// INITIALISE <input_plist>.inh_list to empty list <input_plist>.node = createNode(label: “formal_parameters”, ID.addr, <dataType>.addr); Insert_at_end(<input_plist>.node, <input_plist>.inh_list); <N1>.inh_list = <input_plist>.inh_list // bottom up <input_plist>.syn_list = <N1>.syn_list FreeNode COLON, <N1> //freeNode(dataType)
13	<N1> COMMA ID COLON <dataType> <N1>_1	<N1>.node = createNode(label: “formal_parameters”, ID.addr, <dataType>.addr); Insert_at_end(<N1>.node, <N1>.inh_list); <N1>_1.inh_list = <N1>.inh_list; // bottom up <N1>.syn_list = <N1>_1.syn_list FreeNode COMMA, COLON, <N1>_1
14	<N1> epsilon	<N1>.syn_list = <N1>.inh_list // bottom up FreeNode epsilon
15	<output_plist> ID COLON <type> <N2>	// INITIALISE <output_plist>.inh_list to empty list <output_plist>.node = createNode(label: “formal_parameters”, ID.addr, <type>.addr); Insert_at_end(<output_plist>.node, <output_plist>.inh_list); <N1>.inh_list = <output_plist>.inh_list // bottom up <output_plist>.syn_list = <N1>.syn_list FreeNode COLON, <N1>
16	<N2> COMMA ID COLON <type> <N2>_1	<N2>.node = createNode(label: “formal_parameters”, ID.addr, <type>.addr); Insert_at_end(<N2>.node, <N2>.inh_list); <N2>_1.inh_list = <N2>.inh_list; // bottom up <N2>.syn_list = <N2>_1.syn_list FreeNode COMMA, COLON, <N2>_1
17	<N2> epsilon	<N2>.syn_list = <N2>.inh_list // bottom up FreeNode epsilon
18	<dataType> ARRAY SQBO <range_arrays> SQBC OF <type>	<dataType>.addr = createNode("array", <range_arrays>.addr, <type>.addr)
19	<dataType> <type>	<dataType>.addr = <type>.addr
20	<range_arrays> <index_arr> RANGEOP <index_arr>_1	<range_arrays>.node = createNode("arr_range", <index_arr>.addr, <index_arr>_1.addr) <range_arrays>.addr = <range_arrays>.node // This populates the addr field, which is used upwards in the tree FreeNode RANGEOP
21	<type> INTEGER	<type>.addr = INTEGER.addr

22	<type> REAL	<type>.addr = REAL.addr
23	<type> BOOLEAN	<type>.addr = BOOLEAN.addr
24	<index_arr> <sign> <new_index>	// top down <index_arr>.node = createNode("index", <sign>.addr, <new_index>.addr) <index_arr>.addr = <index_arr>.node
25	<new_index> NUM	<new_index>.addr = NUM.addr
26	<new_index> ID	<new_index>.addr = ID.addr
27	<sign> PLUS	// top down <sign>.addr = PLUS.addr
28	<sign> MINUS	// top down <sign>.addr = MINUS.addr

29	<sign> epsilon	// top down
		<sign>.addr = NULL
		
		// bottom up
		FreeNode epsilon

30	<moduleDef> START_TK <statements> END	Initialise <statements>.inh_list to empty // bottom up <moduleDef>.addr = statements.syn_list;
31	<statements> <statement> <statements>_1	<statements>.node = createNode(label: “statement”, <statement>.addr) Insert_at_end(<statements>.inh_list, <statements>.node) <statements>_1.inh_list = <statements>.inh_list; // bottom up <statements>.syn_list = <statements>_1.syn_list FreeNode <statements>_1
32	<statements> epsilon	<statements>.syn_list = <statements>.inh_list FreeNode epsilon
33	<statement> <ioStmt>	<statement>.addr = <ioStmt>.addr
34	<statement> <simpleStmt>	<statement>.addr = <simpleStmt>.addr
35	<statement> <declareStmt>	<statement>.addr = <declareStmt>.addr
36	<statement> <condionalStmt>	<statement>.addr = <conditionalStmt>.add
37	<statement> <iterativeStmt>	<statement>.addr = <iterativeStmt>.addr
38	<ioStmt> GET_VALUE BO ID BC SEMICOL	<ioStmt>.node = createNode("input", GET_VALUE.addr, ID.addr) <ioStmt>.addr = <ioStmt>.node FreeNode BO, BC, SEMICOL
39	<ioStmt> PRINT BO <var_print> BC SEMICOL	<ioStmt>.node = createNode("output", PRINT.addr, <var_print>.addr) <ioStmt>.addr = <ioStmt>.node FreeNode BO, BC, SEMICOL
40	<boolConstt> TRUE	<boolConstt>.addr = TRUE.addr
41	<boolConstt> FALSE	<boolConstt>.addr = FALSE.addr

42	<var_print> ID <P1>	<var_print>.node = createNode("printElement", ID.addr, <P1>.addr) <var_print>.addr = <var_print>.node
43	<var_print> NUM	<var_print>.addr = NUM.addr
44	<var_print> RNUM	<var_print>.addr = RNUM.addr
45	<var_print> <boolConstt>	<var_print>.addr = <boolConstt>.addr
46	<P1> SQBO <index_arr> SQBC	<P1>.addr = <index_arr>.addr FreeNode SQBO, SQBC
47	<P1> epsilon	<P1>.addr = NULL FreeNode epsilon
48	<simpleStmt> <assignmentStmt>	<simpleStmt>.addr = <assignmentStmt>.addr freeNode(<assignmentStmt>)
49	<simpleStmt> <moduleReuseStmt>	<simpleStmt>.addr = <moduleReuseStmt>.addr freeNode(<moduleReuseStmt>)
50	<assignmentStmt> ID <whichStmt>	<assignmentStmt> = createNode("assignmentStmt",ID.addr,<whichStmt>.addr) freeNode(<whichStmt>)
51	<whichStmt> <lvalueIDStmt>	<whichStmt>.addr = <lvalueIDStmt>.addr freeNode(<lvalueIDStmt>)
52	<whichStmt> <lvalueARRStmt>	<whichStmt>.addr = <lvalueARRStmt>.addr freeNode(<lvalueARRStmt>)
53	<lvalueIDStmt> ASSIGNOP <expression> SEMICOL	//bottom up <lvalueIDStmt>.addr = <expression>.addr freeNode(<expression>)
54	<lvalueARRStmt> SQBO <element_index_with_expression	<lvalueARRStmt> SQBO <element_index_with_expressions> SQBC ASSIGNOP <expression> SEMICOL <lvalueARRStmt>.addr = createNode("lvalueARRStmt",<element_index_with_expressions>.syn_list,<expression>.addr) freeNode SQBO SQBC ASSIGNOP SEMICOL

55	<moduleReuseStmt> <optional> USE MODULE ID WITH PA	//bottom up
		<moduleReuseStmt>.addr = createNode("moduleReuseStmt",<optional>.addr,<actual_para_list>.syn_list)
		freeNode(<optional>)
		freeNode(<actual_para_list>)

56	<actual_para_list> <list_item> <actual_para_list'>	//top down
		//Initialise <actual_para_list>.inh_list to empty
		
		//bottom up
		<actual_para_list>.syn_list = <actual_para_list'>.syn_list
		insertAtBeg(<actual_para_list>.syn_list,<list_item>.addr)
		freeNode(<actual_para_list'>)
		freeNode(COMMA)
		freeNode(<list_item>)

57	<actual_para_list'> COMMA <list_item> <actual_para_list'	//top down
		 //Initialise <actual_para_list>.inh_list to empty
		
		 //bottom up
		 <actual_para_list'>.syn_list = <actual_para_list'>_1.syn_list
		 insertAtBeg(<actual_para_list'>.syn_list,<list_item>.addr)
		 freeNode(<actual_para_list'>_1)
		 freeNode(COMMA)
		 freeNode(<list_item>)

58	<actual_para_list'> epsilon	<actual_para_list'>.addr = NULL freeNode(epsilon)

59	<list_item> <sign> <actual_list_item>	<list_item>.addr = createNode("list_item",<sign>.addr,<actual_list_item>.addr)
		 freeNode(<sign>)
		 freeNode(<actual_list_item>)

60	<list_item> <boolConstt>	<list_item>.addr = <boolConstt>.addr freeNode(<boolConstt>)
61	<actual_list_item> NUM	<actual_list_item>.addr = NUM.addr
62	<actual_list_item> RNUM	<actual_list_item>.addr = RNUM.addr
63	<actual_list_item> ID <N_11>	<actual_list_item>.node = createNode("item", ID.addr, <N_11>.addr)
64	<N_11> SQBO <element_index_with_expressions> SQBC	<N_11>.addr = <element_index_with_expressions>.addr FreeNode SQBO,SQBC
65	<N_11> epsilon	<N_11>.addr = NULL FreeNode epsilon

66	<optional> SQBO <idList> SQBC ASSIGNOP	//bottom up
		 <optional>.addr = createNode("optional",<idList>.syn_list,ASSIGNOP.addr)
		 freeNode(<idList>)
		 freeNode(SQBO)
		 freeNode(SQBC)

67	<optional> epsilon	<optional>.addr = NULL freeNode(epsilon)

68	<idList> ID <N3>	//bottom up
		 <idList>.syn_list = <N3>.syn_list
		 insertAtBeg(<idList>.syn_list,ID.addr)
		 freeNode(<N3>)

69	<N3> COMMA ID <N3>_1	//bottom up
		 <N3>.syn_list = <N3>_1.syn_list
		 insertAtBeg(<N3.syn_list>,ID.addr)
		 freeNode(COMMA)
		 freeNode(<N3>_1.addr)

70	<N3> epsilon	//bottom up <N3>.syn_list = empty list

71	<expression> <arithmeticOrBooleanExpr>	// Bottom up <expression>.addr = <arithmeticOrBooleanExpr>.addr freeNode <arithOrBooleanExp>
72	<expression> <U>	// Bottom Up <expression>.addr = <U>.addr freeNode <U>
73	<U> <unary_op> <new_NT>	// bottom Up <U>.addr = createNode(<unary_op>.addr, <new_NT>.addr) freeNode <unary_op>, <new_NT>
74	<new_NT> BO <arithmeticExpr> BC	// bottom up <new_NT>.addr = <arithmeticExpr>.addr freeNode BO, BC, <arithmeticExpr>
75	<new_NT> <var_id_num>	// bottom up <new_NT>.addr = <var_id_num>.addr freeNode <var_id_num>
76	<var_id_num> ID	// bottom up <var_id_num>.addr = ID.addr
77	<var_id_num> NUM	// bottom up <var_id_num>.addr = NUM.addr
78	<var_id_num> RNUM	// bottom up <var_id_num>.addr = RNUM.addr
79	<unary_op> PLUS	<unary_op>.addr = PLUS.addr
80	<unary_op> MINUS	<unary_op>.addr = MINUS.addr

81	<arithmeticOrBooleanExpr> <AnyTerm> <N7>	// top down
		<N7>.inh_addr = <arithmeticOrBooleanExpr>.addr
		
		// bottom up
		<arithmeticOrBooleanExpr>.addr = <Anyterm>.addr
		freeNode <AnyTerm>

82	<N7> <logicalOp> <AnyTerm> <N7>	// top down
		<N7>.addr = createNode(<logicalOp>.addr, <N7>.inh_addr, <AnyTerm>.addr)
		<N7>_1.inh_addr = <N7>.addr
		freeNode <AnyTerm>
		
		// bottom up
		<N7>.syn_addr = <N7>_1.syn_addr
		freeNode <N7>_1

83	<N7> epsilon	// bottom up <N7>.syn_addr = <N7>.inh_addr

84	<AnyTerm> <arithmeticExpr> <N8>	// top down
		<N8>.inh_addr = <AnyTerm>.addr
		
		// bottom up
		<AnyTerm>.addr = <arithmeticExpr>.addr
		<AnyTerm>.syn_addr = <N8>.syn_addr
		freeNode <arithmeticExpr>

85	<AnyTerm> <boolConstt>	<AnyTerm>.addr = <boolConstt>.addr freeNode(<boolConstt>)
86	<N8> <relationalOp> <arithmeticExpr>	// bottom up <N8>.syn_addr = createNode(<relationalOp>.addr, <N8>.inh_addr, <arithmeticExpr>.syn_addr) freeNode <relationalOp>
87	<N8> epsilon	freeNode epsilon

88	<arithmeticExpr> <term> <N4>	// bottom up
		<arithmeticExpr>.addr = <term>.addr
		<N4>.inh_addr = <arithmeticExpr>.addr
		freeNode <term>
		
		// top down
		<arithmeticExpr>.syn_addr = <N4>.syn_addr
		freeNode <N4>

89	<N4> <op1> <term> <N4>	// top down
		<N4>.addr = createNode(<op1>.addr, <term>.addr, <N4>.inh_addr)
		<N4>_1.inh_addr = <N4>.addr
		freeNode <op1>, <term>
		
		// bottom up
		<N4>.syn_addr = <N4>_1.syn_addr
		freeNode <N4>_1

90	<N4> epsilon	// bottom up <N4>.syn_addr = <N4>.inh_addr freeNode <N4>

91	<term> <factor> <N5>	// bottom up
		<term>.addr = <factor>.addr
		<term>syn_addr = <N5>.syn_addr // note -- order of traversal is note direct
		freeNode <factor>
		
		// top down
		<N5>.inh_addr = <term>.addr

92	<N5> <op2> <factor> <N5>_1	<N5> <op2> <factor> <N5>_1
		// top down
		<N5>.addr = createNode(<op2>.addr, <N5>.inh_addr, <factor>.addr)
		<N5>_1.inh_addr = <N5>.addr
		freeNode <op2> factor
		
		// bottom up
		<N5>.syn_addr = <N5>_1.syn_addr
		freeNode <N5>_1

93	<N5> epsilon	<N5> epsilon // bottom up <N5>.syn_addr = <N5>.inh_addr freeNode epsilon

94	<factor> BO <arithmeticOrBooleanExpr> BC	//bottom up
		<factor>.addr = <arithmeticOrBooleanExpr>.addr
		freeNode(BO)
		freeNode(<arithmeticOrBooleanExpr>)
		freeNode(BC)

95	<factor> NUM	<factor>.addr = NUM.addr
96	<factor> RNUM	<factor>.addr = RNUM.addr
97	<factor> <boolConstt>	<factor>.addr = <boolConstt>.addr free(<boolConstt>)
98	<factor> ID <N_11>	//bottom-Up <factor>.addr = createNewNode("factor", ID.addr, <N_11>.addr) free(<N_11>)
99	<arrExpr> <arrTerm> <arr_N4>	// INITIALISE <arrExpr>.inh_list to empty list <arr_N4>.inh_list = <arrExpr>.inh_list <arrExpr>.node = createNode("arithExpr", <arrTerm>.addr, <arr_N4>.addr) <arrExpr>.addr = <arrExpr>.node

100	<arr_N4> <op1> <arrTerm> <arr_N4>_1	<arr_N4>.node = createNode(label: “addSub”, <op1>.addr, <arrTerm>.addr);
		Insert_at_end(<arr_N4>.node, <arr_N4>.inh_list);
		<arr_N4>_1.inh_list = <arr_N4>.inh_list;
		
		// bottom up
		<arr_N4>.syn_list = <arr_N4>_1.syn_list
		<arr_N4>.addr = <arr_N4>.syn_list

101	<arr_N4> epsilon	<arr_N4>.syn_list = <arr_N4>.inh_list FreeNode epsilon
102	<arrTerm> <arrFactor> <arr_N5>	// INITIALISE <arrTerm>.inh_list to empty list <arr_N5>.inh_list = <arrTerm>.inh_list <arrTerm>.node = createNode("arithTerm", <arrFactor>.addr, <arr_N5>.addr) <arrTerm>.addr = <arrTerm>.node

103	<arr_N5> <op2> <arrFactor> <arr_N5>_1	<arr_N5>.node = createNode(label: “mulDiv”, <op2>.addr, <arrFactor>.addr);
		Insert_at_end(<arr_N5>.node, <arr_N5>.inh_list);
		<arr_N5>_1.inh_list = <arr_N5>.inh_list;
		
		// bottom up
		<arr_N5>.syn_list = <arr_N5>_1.syn_list
		<arr_N5>.addr = <arr_N5>.syn_list

104	<arr_N5> epsilon	<arr_N5>.syn_list = <arr_N5>.inh_list FreeNode epsilon
105	<arrFactor> ID	<arrFactor>.addr = ID.addr
106	<arrFactor> NUM	<arrFactor>.addr = NUM.addr
107	<arrFactor> <boolConstt>	<arrFactor>.addr = <boolConstt>.addr
108	<arrFactor> BO <arrExpr> BC	<arrFactor>.node = createNode("doFirst", <arrExpr>.addr) <arrFactor>.addr = <arrFactor>.node FreeNode BO, BC

109	<element_index_with_expressions> <arrExpr>	// INITIALISE <element_index_with_expressions>.inh_list to empty list
		<arrExpr>.inh_list = <element_index_with_expressions>.inh_list
		
		// bottom up
		<element_index_with_expressions>.syn_list = <arrExpr>.syn_list
		<element_index_with_expressions>.addr = <element_index_with_expressions>.syn_list

110	<element_index_with_expressions> <sign> <N_10>	<element_index_with_expressions>.node = createNode("signedArrExpr", <sign>.addr, <N_10>.addr) <element_index_with_expressions>.addr = <element_index_with_expressions>.node
111	<N_10> <new_index>	<N_10>.addr = <new_index>.addr

112	<N_10> BO <arrExpr> BC	// INITIALISE <N_10>.inh_list to empty list
		<arrExpr>.inh_list = <N_10>.inh_list
		
		// bottom up
		<N_10>.syn_list = <arrExpr>.syn_list
		<N_10>.addr = <arrExpr>.syn_list
		FreeNode BO,BC

113	<op1> PLUS	<op1>.addr = PLUS.addr
114	<op1> MINUS	<op1>.addr = MINUS.addr
115	<op2> MUL	<op2>.addr = MUL.addr
116	<op2> DIV	<op2>.addr = DIV.addr
117	<logicalOp> AND	<logicalOP>.addr = AND.addr
118	<logicalOp> OR	<logicalOP>.addr = OR.addr
119	<relationalOp> LT	<relationalOp>.addr = LT.addr
120	<relationalOp> LE	<relationalOp>.addr = LE.addr
121	<relationalOp> GT	<relationalOp>.addr = GT.addr
122	<relationalOp> GE	<relationalOp>.addr = GE.addr
123	<relationalOp> EQ	<relationalOp>.addr = EQ.addr

124	<relationalOp> NE	<relationalOp>.addr = NE.addr

125	<declareStmt> DECLARE <idList> COLON <dataType> SEMI	<declareStmt>.node = createNode("declaration", DECLARE.addr, <idList>.addr, <dataType>.addr)
		  <declareStmt>.addr = <declareStmt>.node
		  FreeNode COLON, SEMICOL

126	<condionalStmt> SWITCH BO ID BC START_TK <caseStmts>	// Initialize <conditionalStmt>.inh_list to an empty list
		  <caseStmts>.inh_list = <conditionalStmt>.inh_list
		
		  <conditionalStmt>.syn_list = <caseStmts>.syn_list
		  <conditionalStmt>.node = createNode("switchStmt", SWITCH.addr, ID.addr, <caseStmts>.addr, <default>.addr)
		  <conditionalStmt>.addr = <conditionalStmt>.node
		
		  FreeNode BO, BC, START_TK, END

127	<caseStmts> CASE <value> COLON <statements> BREAK SE	<caseStmts>.node = createNode("case", <value>.addr, <statements>.addr)
		  insert_at_end(<caseStmts>.node, <caseStmts>.inh_list)
		
		  <N9>.inh_list = <caseStmts>.inh_list
		
		  // bottom up
		  <caseStmts>.syn_list = <N9>.syn_list
		  <caseStmts>.addr = <caseStmts>.syn_list
		
		  FreeNode CASE, COLON, BREAK, SEMICOL

128	<N9> CASE <value> COLON <statements> BREAK SEMICOL	<N9>.node = createNode(label: "caseNode", <value>.addr, <statements>.addr)
		  Insert_at_end(<N9>.node, <N9>.inh_list)
		  <N9>_1.inh_list = <N9>.inh_list
		
		  // bottom up
		  <N9>.syn_list = <N9>_1.syn_list
		
		  FreeNode CASE, COLON, BREAK, SEMICOL

129	<N9> epsilon	<N9>.syn_list = <N9>.inh_list FreeNode epsilon
130	<value> NUM	<value>.addr = NUM.addr
131	<value> TRUE	<value>.addr = TRUE.addr
132	<value> FALSE	<value>.addr = FALSE.addr
133	<default> DEFAULT COLON <statements> BREAK SEMICOL	<default>.addr = <statements>.addr
134	<default> epsilon	<default>.addr = NULL FreeNode(epsilon)
135	<iterativeStmt> FOR BO ID IN <range_for_loop> BC START_	//bottom up <iterativeStmt>.addr = createNode("iterativeStmt",FOR.addr,ID.addr,<range_for_loop>.addr,<statements>.addr) freeNode(<range_for_loop>)
136	<iterativeStmt> WHILE BO <arithmeticOrBooleanExpr> BC	//bottom up <iterativeStmt>.addr = createNode("iterativeStmt",WHILE.addr,<arithmeticOrBooleanExpr>.addr,<statements>.addr)

137	<range_for_loop> <index_for_loop>_1 RANGEOP <index_	//bottom up
		<range_for_loop>.addr = createNode("range_for_loop",<index_for_loop>_1.addr,<index_for_loop>_2.addr)
		freeNode(<index_for_loop>_1)
		freeNode(<index_for_loop>_2)

138	<index_for_loop> <sign_for_loop> <new_index_for_loop>	//bottom up
		<index_for_loop>.addr = createNode("indexForLoop",<sign_for_loop>.addr,<new_index_for_loop>.addr)
		freeNode(<sign_for_loop>)
		freeNode(<new_index_for_loop>)

139	<new_index_for_loop> NUM	//bottom up <new_index_for_loop>.addr = NUM.addr
140	<sign_for_loop> PLUS	//bottom up <sign_for_loop>.addr = PLUS.addr
141	<sign_for_loop> MINUS	//bottom up <sign_for_loop>.addr = MINUS.addr

142	<sign_for_loop> epsilon	//bottom up
		freeNode(epsilon)