1) <mainFunction> → MAIN SQO SQC <stmtAndFuncDefs> END
<mainFunction>.addr = <stmtAndFuncDefs>.addr

2) <stmtAndFuncDefs> → <stmtOrFunctionDef> <N1>
<stmtAndFuncDefs>.addr = new node (stmtAndFuncDef, <stmtOrFuncDef>.addr, <N1>.addr)

3) <N1> → <stmtAndFunctionDef>.addr
<N1>.addr = <stmtAndFunctionDef>.addr

4) <N1> → Epsilon
<N1>.addr = NULL

5) <stmtOrFuncDef> → <stmt>
<stmtOrFuncDef>.addr = <stmt>.addr

6) <stmtOrFuncDef> → <functionDef>
<stmtOrFuncDef>.addr = <functionDef>.addr

7) <stmt> → <declarationstmt>
<stmt>.addr = <declarationstmt>.addr

8) <stmt> → <assignment_type1>
<stmt>.addr = <assignment_type1>.addr

9) <stmt> → <assign_type2>
<stmt>.addr = <assign_type2>.addr

10) <stmt> → <ifstmt>
<stmt>.addr = <ifstmt>.addr

11) <stmt> → <iostmt>
<stmt>.addr = <iostmt>.addr

12) <stmt> → <funCallStmt>.SEMICOLON
<stmt>.addr = <funCallStmt>.addr

13) <functionDef> → FUNCTION SQO <parameter_list1> SQC ASSIGNOP
FUNID SQO <parameter_list2> SQC <stmtAndFuncDefs>
END SEMICOLON

<functionDef.addr> → new node (functionDef, <par_list1>.addr, FUNID.addr
, <par_list2>.addr, <stmtAndFuncDefs>.addr)

14) <par_list> → <type> ID <remainingList>

15) <type> → INT
<type>.addr = INT.addr

16) <type> → REAL
<type>.addr = REAL.addr

17) <type> → STRING
<type>.addr = STRING.addr

18) <type> → MATRIX
<type>.addr = MATRIX.addr

19) <remaining List) → COMMA <parameter_list)

20) <remainingList> → epsilon
<remainingList>.addr = NULL

21) <declaration stmt> → <type> <varlist> SEMICOLON
<declaration stmt>.addr = new node(declstmt, <type>.addr, <varlist>.addr)

22) <varlist> → ID <more_ids>
<varlist>.addr = ID.addr
ID.link = <more_ids>.addr

23) <more ids> → COMMA <varlist>
<more_ids>.addr = <varlist>.addr

24) <more ids> → epsilon
<more_ids>.addr = NULL

25) <assign_type1> → <lhs_singleVar> ASSIGNOP <right Hand side_type1>
SEMICOLON
<assign_type1>.addr = new node(assign_type1, <lhs_singlevar>.addr,
<rightHandside_type1>.addr)

26) <assign_type2> → <lhs_listvar> ASSIGNOP <rhs_type2> SEMICOLON
<assign_type2>.addr = new node(assign_type2, <lhs_listvar>.addr,
<rhs_type2>.addr)

27) <lhs_singlevar> → ID
<lhs_singlevar>.addr = ID.addr.

28) <lhs_listvar> → SQO <var_list> SQC
<lhs_listvar>.addr = <varlist>.addr

29) <rhs_type1> ⟶ <arithmetic Expression>
<rhs-type1>.addr = <arithmetic Expression>.addr

30) <rhs_type1> ⟶ <size Expression>
<rhs_type1>.addr = <size Expression>.addr

31) <rhs-type1> ⟶ <funCallStmt>
<rhs-type1>.addr = <funCallStmt>.addr

32) <rhs_type2> ⟶ <size Expression>
<rhs-type2>.addr = <size Expression>.addr

33) <rhs_type2> ⟶ <funCallStmt>
<rhs-type2>.addr = <funCallstmt>.addr

34) <size Expression> ⟶ SIZE ID
<size Expression>.addr = newnode(sizeExp, SIZE.addr, ID.addr)

35) <ifstmt> ⟶ IF OP <boolean Exp> CL <stmt> <otherStmts> <N2>
<ifstmt>.addr = new node(if, <boolean Exp>.addr, <stmt>.addr, <otherStmts>.addr, <N2>.addr)

36) <N2> ⟶ ELSE <stmt> <otherStmt> ENDIF SEMICOLON
<N2>.addr = new node(N2, <stmt>.addr, <otherStmts>.addr)

37) <N2> ⟶ ENDIF SEMICOLON
<N2>.addr = NULL

38) <otherStmts1> ⟶ <stmt> <otherStmts2> // | &2 to distinguish
<otherStmts1>.addr = <stmt>.addr
<stmt>.link = <otherStmts2>.addr

39) <otherStmts> ⟶ epsilon
<otherStmts>.addr = NULL

40) <ioStmt> ⟶ READ OP ID CL SEMICOLON
<ioStmt>.addr = new node(io, READ.addr, ID.addr)

41) <ioStmt> ⟶ PRINT OP ID CL SEMICOLON
<ioStmt>.addr = new node(io, PRINT.addr, ID.addr)

42) <funCallStmt> ⟶ FUNID OP <inputParameterList> CL
<funCallStmts>.addr = new node(funCall, FUNID.addr, <inputParameterList>.addr)

43) <inputParameterList> ⟶ <var> <listVar>
<inputParameterlist>.addr = <var>.addr.
<var>.link = <listVar>.addr.

44) <listVar> ⟶ COMMA <inputParameterList>
<listvar>.addr = <inputParameterList>.addr

45) <listVar> ⟶ epsilon
<listVar>.addr = NULL

46) <inputParameterlist> ⟶ epsilon
<inputParameterlist>.addr = NULL

47) <operator_lowprec> → PLUS
<operator_lowprec>.addr = PLUS.addr

48) <operator_lowprec> → MINUS
<operator_lowprec>.addr = MINUS.addr

49) <op_highprec> → MUL
<op_highprec>.addr = MUL.addr

50) <op_highprec> → DIV
<op_highprec>.addr = DIV.addr

51) <booleanExp> → op <booleanExp> cl <logicalOp> op <booleanExp> cl
<booleanExp>.addr = new node (boolExp, <booleanExp>.addr, <logicalOp>.addr, <booleanExp>.addr)

52) <booleanExp>.addr = new node (boolExp, <constrainedVars>.addr, <relationalOp>.addr, <constrainedVars>.addr).

53) <constrainedVars> → ID
<constrainedVars>.addr = ID.addr

54) <constrainedVars> → NUM
<constrainedVars>.addr = NUM.addr

55) <constrainedVars> → RNUM
<constrainedVars>.addr = RNUM.addr

56) <var> → ID <N5>
<var>.addr = new node (var, ID.addr, <N5>.addr)

57) <var> → NUM
<var>.addr = NUM.addr

58) <var> → RNUM
<var>.addr = RNUM.addr

59) <var> → STR
<var>.addr = STR.addr

60) <var> → <matrix>
<var>.addr = <matrix>.addr

61) <N5> → <matrixElement>
<N5>.addr = <matrixElement>.addr

62) <N5> → epsilon
<N5>.addr = NULL

63) <matrix> → SQO <rows> SQC
<matrix>.addr = <rows>.addr

64) <rows> → <row> <N6>
<rows>.addr = <row>.addr
<row>.link = <N6>.addr

65) <N6> → SEMICOLON <rows>
<N6>.addr = <rows>.addr

66) <N6> → epsilon
<N6>.addr = NULL

67) <row> → NUM <remaining colElements>
<row>.addr = NUM.addr
NUM.link = <remaining colElements>.addr

68) <remaining colElements1> → COMMA NUM <remaining colElements2>
<remaining colElements1>.addr = NUM.addr
NUM.link = <remaining colElements2>.addr

69) <remaining colElements> → epsilon
<remaining colElements>.addr = NULL

70) <matrix Element> → SQO NUM COMMA NUM SQC
<matrix Element>.addr = new node(matrixElement, NUM.addr, NUM.ad

71) <logical Op> → AND
<logical Op>.addr = AND.addr

72) <logical Op> → OR
<logical Op>.addr = OR.addr

73) <boolean Exp> → NOT OP <boolean Exp2> CL
<boolean Exp1>.addr = new node(boolean Exp, <boolean Exp2>.addr)

74) <relational Op> → LT
<relational Op>.addr = LT.addr

75) <relational Op> → LE
<relational Op>.addr = LE.addr

76) <relational Op> → EQ
<relational Op>.addr = EQ.addr

77) <relational Op> → GT
<relational Op>.addr = GT.addr

78) <relational Op> → GE
<relational Op>.addr = GE.addr

79) <relationalOp> → NE
<relationalOp>.addr = NE.addr

80) <arithmeticExpression> → <arithmeticTerm> <N3>
<arithmeticExpression>.addr =