AST RULES

Group Number: 4

Group Members:

Deepam P Desai : 2020A7PS0971P
Rachit Agrawal : 2020A7PS0033P
Jaysheel Shah : 2020A7PS0083P
Aryan Desai : 2020A7PS0321P
Ramakant Talankar : 2020A7PS0979P

```
1) PROGRAM MODULEDECLARATIONS OTHERMODULES DRIVERMODULE
OTHERMODULES{
 PROGRAM.syn = make_node( “MAIN_PROG”,
make_node(“AST_MODULEDECLARATIONS”, MODULEDECLARATIONS.list_syn),
make_node(“AST_OTHERMODULES”, OTHERMODULES.list_syn),
DRIVERMODULE.syn,          make_node(“AST_OTHERMODULES”,
OTHERMODULES1.list_syn) )
        free(MODULEDECLARATIONS)
        free(OTHERMODULES)
        free(DRIVERMODULE)
        free(OTHERMODULES1)
}

2) MODULEDECLARATIONS MODULEDECLARATION MODULEDECLARATIONS{
MODULEDECLARATIONS.list_syn = insert_head(MODULEDECLARATIONS1.list_syn,
MODULEDECLARATION.syn)
free(MODULEDECLARATION)
free(MODULEDECLARATIONS1)
}

3) MODULEDECLARATIONS epsilon{
MODULEDECLARATIONS.list_syn = NULL
}

4) MODULEDECLARATION declare module id semicol{
MODULEDECLARATION.syn = id
free(declare)
free(module)
free(semicol)
}

5) OTHERMODULES MODULE OTHERMODULES{
OTHERMODULES.list_syn = insert_head(OTHERMODULES1.list_syn, MODULE.syn)
free(MODULE)
free(OTHERMODULES1)
```

```
}

6) OTHERMODULES epsilon{
OTHERMODULES.list_syn = NULL
}

7) DRIVERMODULE driverdef driver program driverenddef MODULEDEF{
DRIVERMODULE.syn = make_node(“AST_DRIVERDEF”, MODULEDEF.syn)
free(driverdef)
free(driver)
free(program)
free(driverenddef)
free(MODULEDEF)
}

8) MODULE def module id enddef takes input sqbo INPUT_PLIST sqbc semicol RET
MODULEDEF{
MODULE.syn = make_node(“AST_MODULE”, id, make_node(“INPUT_PARAMETERS”,
INPUT_PLIST.list_syn), RET.syn, MODULEDEF.syn)
        free(def)
        free(module)
        free(enddef)
        free(takes)
        free(input)
        free(sqbo)

        free(INPUT_LIST)
        free(sqbc)
        free(semicol)
        free(RET)
        free(MODULEDEF)
}

9) RET returns sqbo OUTPUT_PLIST sqbc semicol{
RET.syn = make_node(“OUTPUT_PARAMETERS”, OUTPUT_PLIST.list_syn)
        free(returns)
        free(sqbo)
        free(OUTPUT_PLIST)
        free(sqbc)
        free(semicol)
}

10) RET epsilon{
RET.syn = make_node(“OUTPUT_PARAMETERS”, NULL)
```

```
}

11) INPUT_PLIST id colon DATATYPE INPUT_PLIST_ONE{
        INPUT_PLIST_ONE.list_inh = make_list(DATATYPE.syn)
        INPUT_PLIST_ONE.list_inh = insert_end(INPUT_PLIST_ONE.list_inh, id)
        INPUT_PLIST.list_syn = INPUT_PLIST_ONE.list_syn
        free(colon)
        free(DATATYPE)
        free(INPUT_PLIST_ONE)
}

12) INPUT_PLIST_ONE comma id colon DATATYPE INPUT_PLIST_ONE1{
        INPUT_PLIST_ONE.list_syn = INPUT_PLIST_ONE1.list_syn
        INPUT_PLIST_ONE1.list_inh = insert_end(INPUT_PLIST_ONE.list_inh, DATATYPE.syn)
        INPUT_PLIST_ONE1.list_inh = insert_end(INPUT_PLIST_ONE.list_inh, id)
        free(comma)
        free(colon)
        free(DATATYPE)
        free(INPUT_PLIST_ONE)
}

13) INPUT_PLIST_ONE epsilon{
INPUT_PLIST_ONE.list_syn = INPUT_PLIST_ONE.list_inh
        free(epsilon)
}

14) OUTPUT_PLIST id colon TYPE OUTPUT_PLIST_ONE{
        OUTPUT_PLIST_ONE.list_inh = make_list(TYPE.syn)
        OUTPUT_PLIST_ONE.list_inh = insert_end(OUTPUT_PLIST_ONE.list_inh, id)
        OUTPUT_PLIST.list_syn = OUTPUT_PLIST_ONE.list_syn
        free(colon)
        free(TYPE)
        free(OUTPUT_PLIST_ONE)
}

15) OUTPUT_PLIST_ONE comma id colon TYPE OUTPUT_PLIST_ONE{
        OUTPUT_PLIST_ONE.list_syn = OUTPUT_PLIST_ONE1.list_syn
        OUTPUT_PLIST_ONE1.list_inh = insert_end(OUTPUT_PLIST_ONE.list_inh, TYPE.syn)
        OUTPUT_PLIST_ONE1.list_inh = insert_end(OUTPUT_PLIST_ONE.list_inh, id)
        free(comma)
        free(colon)
        free(TYPE)
        free(OUTPUT_PLIST_DASH1)
}

16) OUTPUT_PLIST_ONE epsilon(
        OUTPUT_PLIST_ONE.list_syn = OUTPUT_PLIST_ONE.list_inh
        free(epsilon)
)

17) DATATYPE integer {
   DATATYPE.syn=integer;
}

18) DATATYPE real {
   DATATYPE.syn=real;
}

19) DATATYPE boolean {
   DATATYPE.syn=boolean;
}

20) DATATYPE array sqbo RANGE_ARRAYS sqbc of TYPE {
DATATYPE.syn=make_node("ARRAY_DATATYPE", TYPE.syn, RANGE_ARRAYS.syn)
   free(array)
   free(sqbo)
   free(RANGE_ARRAYS)
   free(sqbc)
   free(of)
   free(TYPE)
}

21) RANGE_ARRAYS INDEX_ARR1 rangeop INDEX_ARR2 {
   RANGE_ARRAYS.syn=make_node("..",INDEX_ARR1.syn,INDEX_ARR2.syn)
   free(rangeop)
   free(INDEX_ARR1)
   free(INDEX_ARR2)
}

22) TYPE integer {
   TYPE.syn=integer
}

23) TYPE real {
   TYPE.syn=real
}
```

```
24) TYPE boolean {
   TYPE.syn=boolean
}

25) MODULEDEF start STATEMENTS end{
MODULEDEF.syn = make_node(“STMT_LIST”, STATEMENTS.list_syn)
        free(start)
        free(STATEMENTS)
        free(end)
}

26) STATEMENTS STATEMENT STATEMENTS{
        STATEMENTS.list_syn = insert_head(STATEMENTS1.list_syn,
        STATEMENT.syn)
        Free(STATEMENT)
        Free(STATEMENTS1)
}

27) STATEMENTS epsilon{
        STATEMENTS.list_syn = NULL
}

28) STATEMENT IOSTMT{

        STATEMENT.syn = IOSTMT.syn
        free(IOSTMT)

}

29) STATEMENT SIMPLESTMT{
        STATEMENT.syn = SIMPLESTMT.syn
        free(SIMPLESTMT)

}

30) STATEMENT DECLARESTMT{
        STATEMENT.syn = DECLARESTMT.syn
        free(DECLARESTMT)

}

31) STATEMENT CONDITIONALSTMT{
        STATEMENT.syn = CONDITIONALSTMT.syn
        free(CONDITIONALSTMT)
}

32) STATEMENT ITERATIVESTMT{
        STATEMENT.syn = ITERATIVESTMT.syn
        free(ITERATIVESTMT)
}

33) IOSTMT get_value bo id bc semicol{
        IOSTMT.syn = make_node(“SCANSTMT”, id)
        free(get_value)
        free(bo)
        free(bc)
        free(semicol)
}

34) IOSTMT print bo VAR_PRINT bc semicol{
        IOSTMT.syn = make_node(“PRINTSTMT”, VAR_PRINT.syn)
        free(print)
        free(bo)
        free(VAR_PRINT)
        free(bc)
        free(semicol)
}

35) BOOLCONSTT true{
        BOOLCONSTT.syn = true
}

36) BOOLCONSTT false{
        BOOLCONSTT.syn = false
}

37) VAR_PRINT id VAR_PRINT_ONE(
        VAR_PRINT_ONE.inh = id
        VAR_PRINT.syn = VAR_PRINT_ONE.syn
        free(VAR_PRINT_ONE)

)

38) VAR_PRINT num(
        VAR_PRINT.syn = num
)
```

```
39) VAR_PRINT rnum(
        VAR_PRINT.syn = rnum
)

40) VAR_PRINT BOOLCONSTT(
        VAR_PRINT.syn = BOOLCONSTT.syn
        free(BOOLCONSTT)

)

41) VAR_PRINT_ONE sqbo SIGN NEW_INDEX sqbc(

        VAR_PRINT_ONE.syn = make_node(“ARRAY_ADDR”, make_node(
        “INDEX_EXPR”, SIGN.syn, NEW_INDEX.syn))
        free(sqbo)
        free(SIGN)
        free(NEW_INDEX)
        free(sqbc)
)

42) VAR_PRINT_ONE epsilon(
        VAR_PRINT_ONE.syn = VAR_PRINT_ONE.inh
        free(epsilon)
)

43) SIMPLESTMT ASSIGNMENTSTMT {
   SIMPLESTMT.syn = ASSIGNMENTSTMT.syn;
   free(ASSIGNMENTSTMT);
}

44) SIMPLESTMT MODULEREUSESTMT {
   SIMPLESTMT.syn = MODULEREUSESTMT.syn;
   free(MODULEREUSESTMT);
}

45) ASSIGNMENTSTMT id WHICHSTMT {
   WHICHSTMT.inh = id;
   ASSIGNMENTSTMT.syn = WHICHSTMT.syn;
   free(WHICHSTMT);
}

46) WHICHSTMT ONEVALUEIDSTMT {
   ONEVALUEIDSTMT.inh = WHICHSTMT.inh;
   WHICHSTMT.syn = ONEVALUEIDSTMT.syn;
   free(ONEVALUEIDSTMT);
}

47) WHICHSTMT ONEVALUEARRSTMT {
   ONEVALUEARRSTMT.inh = WHICHSTMT.inh;
   WHICHSTMT.syn = ONEVALUEARRSTMT.syn;
   free(ONEVALUEARRSTMT);
}

48) ONEVALUEIDSTMT assignop EXPRESSION semicol
   ONEVALUEIDSTMT.syn = make_node("ASSIGN", ONEVALUEIDSTMT.inh, EXPRESSION.syn);
   free(assignop);
   free(EXPRESSION);
   free(semicol);
}

49) ONEVALUEARRSTMT sqbo ELEMENT_INDEX_WITH_EXPRESSIONS sqbc assignop EXPRESSION semicol {
   ONEVALUEARRSTMT.syn = make_node("ASSIGN", make_node("ARRAY_ADDR", ONEVALUEARRSTMT.inh, ELEMENT_INDEX_WITH_EXPRESSIONS.syn), EXPRESSION.syn);
   free(sqbo);
   free(ELEMENT_INDEX_WITH_EXPRESSIONS);
   free(sqbc);
   free(assignop);
   free(EXPRESSION);
   free(semicol);
}

50) INDEX_ARR SIGN NEW_INDEX {
   INDEX_ARR.syn=make_node("INDEX_EXPR",SIGN.syn,NEW_INDEX.syn)
   free(SIGN)
   free(NEW_INDEX)

}

51) NEW_INDEX num {
   NEW_INDEX.syn=num
}

52) NEW_INDEX id {
   NEW_INDEX.syn=id
}
```

```
53) SIGN plus {
   SIGN.syn=plus
}


54) SIGN minus {
   SIGN.syn=minus
}

55) SIGN epsilon {
         SIGN.syn=NULL
         free(epsilon)
}

56) MODULEREUSESTMT OPTIONAL use module id with parameters
ACTUAL_PARA_LIST semicol {
   MODULEREUSESTMT.syn = make_node("INVOKE_FUNCTION", id,
make_node("OPTIONAL_PARAMETERS", OPTIONAL.list_syn),
make_node("OPTIONAL_PARAMETERS", ACTUAL_PARA_LIST.list_syn));
   free(OPTIONAL);
   free(use);
   free(module);
   free(with);
   free(parameters);
   free(ACTUAL_PARA_LIST);
   free(semicol);
}

57) ACTUAL_PARA_LIST SIGN K_OLD ACTUAL_PARA_LIST_TWO {
   ACTUAL_PARA_LIST_TWO.list_inh =
make_list(make_node(“PARAM_EXPR”,SIGN.syn, K_OLD.syn));
   ACTUAL_PARA_LIST.list_syn = ACTUAL_PARA_LIST_TWO.list_syn;
   free(SIGN);
   free(K_OLD);
   free(ACTUAL_PARA_LIST_TWO);
}

58) ACTUAL_PARA_LIST_TWO comma SIGN K_OLD ACTUAL_PARA_LIST_TWO1
{ACTUAL_PARA_LIST_TWO1list_.inh=insert_end(ACTUAL_PARA_LIST_TWO.list_inh,
make_node(“PARAM_EXPR”, SIGN.syn, K_OLD.syn));
   ACTUAL_PARA_LIST_TWO.list_syn=ACTUAL_PARA_LIST_TWO1.list_syn;
   free(comma);
   free(SIGN);
```

```
    free(K_OLD);
    free(ACTUAL_PARA_LIST_TWO1);
}

59) ACTUAL_PARA_LIST_TWO epsilon {
    ACTUAL_PARA_LIST_TWO.lsit_syn = ACTUAL_PARA_LIST_TWO.list_inh;
            free(epsilon)
}

60) K_OLD num {
    K_OLD.syn = num;
}

61) K_OLD rnum {
    K_OLD.syn = rnum;
}

62) K_OLD BOOLCONSTT {
    K_OLD.syn = BOOLCONSTT.syn;
    free(BOOLCONSTT);
}

63) K_OLD id N_ELEVEN {
    N_ELEVEN.inh = id;
    K_OLD.syn = N_ELEVEN.syn;
    free(N_ELEVEN);
}

64) OPTIONAL sqbo IDLIST sqbc assignop {
    OPTIONAL.list_syn = IDLIST.list_syn;
    free(sqbo);
    free(IDLIST);
    free(sqbc);
    free(assignop);
}

65) OPTIONAL epsilon {
    OPTIONAL.list_syn = NULL;
}

66) IDLIST id IDLIST_ONE{

    IDLIST_ONE.list_inh=make_list(id)
    IDLIST.list_syn=IDLIST_ONE.list_syn
```

```
   free(IDLIST_ONE)
}

67) IDLIST_ONE comma id IDLIST_ONE1{
   IDLIST_ONE1.list_inh=insert_end(id)
   IDLIST_ONE.list_syn=IDLIST_ONE1.list_syn
   free(comma)
   free(INLIST_ONE1)
}

68) IDLIST_ONE epsilon{
   IDLIST_ONE.list_syn=IDLIST_ONE.list_inh
         free(epsilon)
}

69) EXPRESSION ARITHMETICORBOOLEANEXPR {
   EXPRESSION.syn = ARITHMETICORBOOLEANEXPR.syn;
   free(ARITHMETICORBOOLEANEXPR);
}

70) EXPRESSION UNARY {
   EXPRESSION.syn = UNARY.syn;
   free(UNARY);
}

71) UNARY UNARY_OP NEW_NT {
   NEW_NT.inh = UNARY_OP.syn;
   UNARY.syn = NEW_NT.syn;
   free(UNARY_OP);
   free(NEW_NT);
}

72) NEW_NT bo ARITHMETICEXPR bc {
   NEW_NT.syn = make_node("UNARYEXPR", NEW_NT.inh, ARITHMETICEXPR.syn);
   free(bo);
free(ARITHMETICEXPR);
   free(bc);
}

73) NEW_NT VAR_ID_NUM {
   NEW_NT.syn = VAR_ID_NUM.syn;
   free(VAR_ID_NUM);
}
```

```
}

74) VAR_ID_NUM id {
   VAR_ID_NUM.syn = id;
}

75) VAR_ID_NUM num {
   VAR_ID_NUM.syn = num;
}

76) VAR_ID_NUM rnum {
   VAR_ID_NUM.syn = rnum;
}

77) UNARY_OP plus {
   UNARY_OP.syn = plus;
}

78) UNARY_OP minus {
   UNARY_OP.syn = minus;
}

79) ARITHMETICORBOOLEANEXPR ANYTERM
ARITHMETICORBOOLEANEXPR_ONE {
   ARITHMETICORBOOLEANEXPR_ONE.inh = ANYTERM.syn;
   ARITHMETICORBOOLEANEXPR.syn = ARITHMETICORBOOLEANEXPR_ONE.syn;
   free(ANYTERM);
   free(ARITHMETICORBOOLEANEXPR_ONE);
}

80) ARITHMETICORBOOLEANEXPR_ONE LOGICALOP ANYTERM
ARITHMETICORBOOLEANEXPR_ONE1 {
   ARITHMETICORBOOLEANEXPR_ONE1.inh = make_node(LOGICALOP.syn,
ARITHMETICORBOOLEANEXPR_ONE.inh, ANYTERM.syn);
   ARITHMETICORBOOLEANEXPR_ONE.syn =
ARITHMETICORBOOLEANEXPR_ONE1.syn;
   free(LOGICALOP);
   free(ANYTERM);
   free(ARITHMETICORBOOLEANEXPR_ONE1);
}

81) ARITHMETICORBOOLEANEXPR_ONE epsilon {
   ARITHMETICORBOOLEANEXPR_ONE.syn =
ARITHMETICORBOOLEANEXPR_ONE.inh;
         free(epsilon)
}
```

```
82) ANYTERM ARITHMETICEXPR ANYTERM_ONE {
   ANYTERM_ONE.inh = ARITHMETICEXPR.syn;
   ANYTERM.syn = ANYTERM_ONE.syn;
   free(ARITHMETICEXPR);
   free(ANYTERM_ONE);
}


83) ANYTERM BOOLCONSTT {
   ANYTERM.syn = BOOLCONSTT.syn;
   free(BOOLCONSTT);
}

84) ANYTERM_ONE RELATIONALOP ARITHMETICEXPR {
   ANYTERM_ONE.syn = make_node(RELATIONALOP.syn, ANYTERM_ONE.inh,
ARITHMETICEXPR.syn);
   free(RELATIONALOP);
   free(ARITHMETICEXPR);
}

85) ANYTERM_ONE epsilon {
   ANYTERM_ONE.syn = ANYTERM_ONE.inh;
}

86) ARITHMETICEXPR TERM ARITHMETICEXPR_ONE {
   ARITHMETICEXPR_ONE.inh = TERM.syn;
   ARITHMETICEXPR.syn = ARITHMETICEXPR_ONE.syn;
   free(TERM);
   free(ARITHMETICEXPR_ONE);
}

87) ARITHMETICEXPR_ONE OP_ONE TERM ARITHMETICEXPR_ONE1 {
   ARITHMETICEXPR_ONE1.inh = make_node(OP_ONE.syn,
ARITHMETICEXPR_ONE.inh, TERM.syn);
   ARITHMETICEXPR_ONE.syn = ARITHMETICEXPR_ONE1.syn;
   free(OP_ONE);
   free(TERM);
   free(ARITHMETICEXPR_ONE1);
}

88) ARITHMETICEXPR_ONE epsilon {
   ARITHMETICEXPR_ONE.syn = ARITHMETICEXPR_ONE.inh;
}
```

```
89) TERM FACTOR TERM_ONE {
    TERM_ONE.inh = FACTOR.syn;
    TERM.syn = TERM_ONE.syn;
    free(FACTOR);
    free(TERM_ONE);
}


90) TERM_ONE OP_TWO FACTOR TERM_ONE1  {
    TERM_ONE.inh = make_node(OP_TW0.syn, TERM_ONE.inh, FACTOR.syn);
    TERM_ONE.syn = TERM_ONE1.syn;
    free(OP_TWO);
    free(FACTOR);
    free(TERM_ONE1);
}

91) TERM_ONE epsilon {
    TERM_ONE.syn = TERM_ONE.inh;
}

92) FACTOR bo ARITHMETICORBOOLEANEXPR bc {
    FACTOR.syn = ARITHMETICORBOOLEANEXPR.syn;
    free(bo);
    free(bc);
    free(ARITHMETICORBOOLEANEXPR);
}

93) FACTOR num {
    FACTOR.syn = num;
}

94) FACTOR rnum {
    FACTOR.syn = rnum;
}

95) FACTOR BOOLCONSTT {
    FACTOR.syn = BOOLCONSTT.syn;
    free(BOOLCONSTT);
}

96) FACTOR id N_ELEVEN {
    N_ELEVEN.inh = id;
    FACTOR.syn = N_ELEVEN.syn;
```

```
   free(N_ELEVEN);
}

97) N_ELEVEN sqbo ELEMENT_INDEX_WITH_EXPRESSIONS sqbc {
   N_ELEVEN.syn = make_node("ARRAY_ADDR", N_ELEVEN.inh,
ELEMENT_INDEX_WITH_EXPRESSIONS.syn);
        free(sqbo);
        free(ELEMENT_INDEX_WITH_EXPRESSIONS);
        free(sqbc);
}

98) N_ELEVEN epsilon {
        N_ELEVEN.syn = N_ELEVEN.inh;
        free(epsilon);
}

99) ELEMENT_INDEX_WITH_EXPRESSIONS SIGN N_TEN {
            N_TEN.inh = SIGN.syn;
            ELEMENT_INDEX_WITH_EXPRESSIONS.syn = N_TEN.syn;
            free(SIGN);
            free(N_TEN);
}

100) ELEMENT_INDEX_WITH_EXPRESSIONS ARREXPR {
   ELEMENT_INDEX_WITH_EXPRESSIONS.syn = ARREXPR.syn;
   free(ARREXPR);
}

101) N_TEN NEW_INDEX {
   N_TEN.syn = NEW_INDEX.syn;
   free(NEW_INDEX);
}

102) N_TEN bo ARREXPR bc {
        N_TEN.syn = make_node("UNARYARREXPR", N_TEN.inh, ARREXPR.syn);
        free(bo);
        free(ARREXPR);
        free(bc);
}

103) ARREXPR ARRTERM ARR_N_FOUR {
   ARR_N_FOUR.inh = ARRTERM.syn;
   ARREXPR.syn = ARR_N_FOUR.syn;
   free(ARRTERM);
   free(ARR_N_FOUR);
}

104) ARR_N_FOUR OP_ONE ARRTERM ARR_N_FOUR1 {
   ARR_N_FOUR1.inh = make_node(OP_ONE.syn, ARR_N_FOUR.inh,
ARRTERM.syn);
   ARR_N_FOUR.syn = ARR_N_FOUR1.syn;
   free(OP_ONE);
   free(ARRTERM);
   free(ARR_N_FOUR1);
}

105) ARR_N_FOUR epsilon {
        ARR_N_FOUR.syn = ARR_N_FOUR.inh;
        free(epsilon);
}

106) ARRTERM ARRFACTOR ARR_N_FIVE {
   ARR_N_FIVE.inh = ARRFACTOR.syn;
   ARRTERM.syn = ARR_N_FIVE.syn;
   free(ARRFACTOR);
   free(ARR_N_FIVE);
}

107) ARR_N_FIVE OP_TWO ARRFACTOR ARR_N_FIVE1 {
   ARR_N_FIVE1.inh = make_node(OP_TW0.syn, ARR_N_FIVE.inh,
ARRFACTOR.syn);
   ARR_N_FIVE.syn = ARR_N_FIVE1.syn;
   free(OP_TW0);
   free(ARRFACTOR);
   free(ARR_N_FIVE1);
}

108) ARR_N_FIVE epsilon {
        ARR_N_FIVE.syn = ARR_N_FIVE.inh;
        free(epsilon);
}

109) ARRFACTOR id {
   ARRFACTOR.syn = id;
}

110) ARRFACTOR num {
   ARRFACTOR.syn = num;
```

```
}

111) ARRFACTOR BOOLCONSTT {
   ARRFACTOR.syn = BOOLCONSTT.syn;
         free(BOOLCONSTT);
}

112) ARRFACTOR bo ARREXPR bc {
   ARRFACTOR.syn = ARREXPR.syn;
   free(bo);
         free(ARREXPR);
   free(bc);
}

113) OP_ONE plus {
   OP_ONE.syn = plus;
}

114) OP_ONE minus {
   OP_ONE.syn = minus;
}

115) OP_TWO mul {
   OP_TWO.syn = mul;
}

116) OP_TWO div {
   OP_TWO.syn = div;
}

117) LOGICALOP and {
   LOGICALOP.syn = and;
}

118) LOGICALOP or {
   LOGICALOP.syn = or;
}

119) RELATIONALOP lt {
   RELATIONALOP.syn = lt;
}

120) RELATIONALOP le {
   RELATIONALOP.syn = le;
```

```
}

121) RELATIONALOP gt {
   RELATIONALOP.syn = gt;
}

122) RELATIONALOP ge {
   RELATIONALOP.syn = ge;
}

123) RELATIONALOP eq {
   RELATIONALOP.syn = eq;
}

124) RELATIONALOP ne {
   RELATIONALOP.syn = ne;
}

125) DECLARESTMT declare IDLIST colon DATATYPE semicol {
DECLARESTMT.syn=make_node("AST_DECLARESTATEMENT",IDLIST.list_syn,DATA
TYPE.syn);
   free(declare);
   free(colon);
   free(semicol);
   free(IDLIST);
   free(DATATYPE);
}

126) CONDITIONALSTMT switch bo id bc start CASESTMT DEFAULT end {
   CONDITIONALSTMT.syn=make_node("SWITCH_CASE",
id,make_node(“CASE_NUM”, CASESTMT.list_syn),DEFAULT.syn);
   free(switch);
   free(bo);
   free(bc);
   free(start);
   free(CASESTMT);
   free(DEFAULT);
   free(end);
}

127) CASESTMT case VALUE colon STATEMENTS break semicol CASESTMT_ONE {
   CASESTMT.list_syn = insert_head(
CASESTMT_ONE.list_syn, make_node(“AST_CASE”, VALUE.syn,
make_node(“STMT_LIST”, STATEMENTS.list_syn))
```

```
);
   free(case)
   free(VALUE)
   free(colon)
   free(STATEMENTS)
   free(break)
   free(semicol)
   free(CASESTMT_ONE)
}

128) CASESTMT_ONE case VALUE colon STATEMENTS break semicol
CASESTMT_ONE1 {
   CASESTMT_ONE.list_syn = insert_head(
CASESTMT_ONE1.list_syn,
make_node(“AST_CASE”, VALUE.syn, make_node(“STMT_LIST”,
STATEMENTS.list_syn))
);
   free(case)
   free(VALUE)
   free(colon)
   free(STATEMENTS)
   free(break)
   free(semicol)
   free(CASESTMT_ONE1)
}

129) CASESTMT_ONE epsilon {
   CASESTMT_ONE.list_syn=NULL;
free(epsilon);
}

130) VALUE num {
   VALUE.syn=num
}

131) VALUE true {
   VALUE.syn=true
}

132) VALUE false {
   VALUE.syn=false
}

133) DEFAULT default colon STATEMENTS break semicol {
   DEFAULT.syn=make_node("AST_DEFAULT", make_node(“STMT_LIST”, STATEMENTS.list_syn));
   free(default)
   free(colon)
   free(break)
   free(semicol)
   free(STATEMENTS)
}

134) DEFAULT epsilon {
   DEFAULT.syn=NULL
         free(epsilon)
}

135) ITERATIVESTMT for bo id in RANGE_FOR_LOOP bc start STATEMENTS end {
   ITERATIVESTMT.syn=make_node("FOR_STMT", id, RANGE_FOR_LOOP.syn, make_node(“STMT_LIST”, STATEMENTS.list_syn));
   free(for)
   free(bo)
   free(in)
   free(RANGE_FOR_LOOP)
   free(bc)
   free(start)
   free(STATEMENTS)
   free(end)
}

136) ITERATIVESTMT while bo ARITHMETICORBOOLEANEXPR bc start STATEMENTS end {
   ITERATIVESTMT.syn=make_node("WHILE_STMT", ARITHMETICORBOOLEANEXPR.syn, make_node(“STMT_LIST”, STATEMENTS.list_syn));
   free(while)
   free(bo)
   free(ARITHMETICORBOOLEANEXPR)
   free(bc)
   free(start)
   free(STATEMENTS)
   free(end)
}

137) RANGE_FOR_LOOP INDEX_FOR_LOOP1 rangeop INDEX_FOR_LOOP2 {
RANGE_FOR_LOOP.syn=make_node("..",INDEX_FOR_LOOP1.syn,INDEX_FOR_LOOP2.syn)
```

```
   free(rangeop)
   free(INDEX_FOR_LOOP1)
   free(INDEX_FOR_LOOP2)
}

138) INDEX_FOR_LOOP SIGN_FOR_LOOP NEW_INDEX_FOR_LOOP {
INDEX_FOR_LOOP.syn=make_node("INDEX_EXPR",SIGN_FOR_LOOP.syn,NEW_INDEX_FOR_LOOP1.syn)
   free(SIGN_FOR_LOOP)
   free(NEW_INDEX_FOR_LOOP)
}

139) NEW_INDEX_FOR_LOOP num {
   NEW_INDEX_FOR_LOOP.syn=num
}

140) SIGN_FOR_LOOP plus {
   SIGN_FOR_LOOP.syn=plus
}

141) SIGN_FOR_LOOP minus {
   SIGN_FOR_LOOP.syn=minus
}

142) SIGN_FOR_LOOP epsilon {
   SIGN_FOR_LOOP.syn=NULL
         free(epsilon);
}
```