1)startprogram,program,DOLLAR
= {
	//bottom-up
	a)free(DOLLAR);
	b)startprogram.addr = program.addr;
	c)free(program);
  }

2)program,moduleDeclarations,otherModules1,driverModule,otherModules2
={tmt
	//bottom-up
	a)Program.addr =
	make_new_node(“PROGRAM”,moduleDeclarations.syn,otherModules1.syn,driverModul
	e.addr,otherModules2.syn);
	b)free(moduleDeclarations);
	c)free(otherModules1);
	d)free(driverModule);
	e)free(otherModules2);
}
3)moduleDeclarations,moduleDeclaration,moduleDeclarations1
={
	//bottom-up
	a)moduleDeclarations.syn =
	insert_at_beginning(moduleDeclarations1.syn,moduleDeclaration.addr);
	b)free(moduleDeclaration);
	c)free(moduleDeclarations1);
 }
4)moduleDeclarations,EPSILON
={
	//bottom-up
	a)moduleDeclarations.syn = NULL;
	b)free(EPSILON);
}
5)moduleDeclaration,DECLARE,MODULE,ID,SEMICOL
={
	//bottom-up
	a)moduleDeclaration.addr = ID.addr;
	b)free(DECLARE);
	c)free(MODULE);
	d)free(SEMICOL);
 }
6)otherModules,module,otherModules1
={
	//bottom-up

a)otherModules.syn = insert_at_beginning(otherModules1.syn,module.addr);
b)free(otherModules1);
c)free(module);
}

7)otherModules,EPSILON

={

//bottom-up

a)otherModules.syn = NULL;

b)free(EPSILON);

}

8)driverModule,DRIVERDEF,DRIVER,PROGRAM,DRIVERENDDEF,moduleDef

={

//bottom-up

a)driverModule.addr = moduleDef.syn;

b)free(DRIVERDEF);

c)free(DRIVER);

d)free(PROGRAM);

e)free(DRIVERENDDEF);

f)free(moduleDef);

}

9)module,DEF,MODULE,ID,ENDDEF,TAKES,INPUT,SQBO,input_plist,SQBC,SEMICOL,ret,moduleDef

={

//bottom-up

a)free(DEF);

b)free(MODULE);

c)free(ENDDEF);

d)free(TAKES);

e)free(INPUT);

f)free(SQBO);

g)free(SQBC);

h)free(SEMICOL);

i) module.addr =

make_new_node(“MODULE_DEF”,ID.addr,input_plist.syn,ret.syn,moduleDef.syn);

j)free(input_plist);

k)free(ret);

l)free(moduleDef);

}

10)ret,RETURNS,SQBO,output_plist,SQBC,SEMICOL

={

//bottom-up

a)free(RETURNS);

b)free(SQBO);

c)ret.syn = output_plist.syn;

d)free(output_plist);

```
        e)free(SQBC);
        f)free(SEMICOL);
 }
11)ret,EPSILON
={
        //bottom-up
        a)ret.syn = NULL;
        b)free(EPSILON);
   }
12)input_plist,ID,COLON,dataType,moreList
=
{
        //bottom-up
        a)input_plist.syn = insert_at_beginning(moreList.syn,make_pair(ID.addr,dataType.addr))
        b)free(COLON)
        c)free(dataType)
        d)free(moreList)
}
13)moreList,COMMA,ID,COLON,dataType,moreList1
= {
        //bottom up
        a)moreList.syn = insert_at_beginning(morelist1.syn,make_pair(Id.addr,dataType.addr))
        b)free(COMMA)
        c)free(COLON)
        d)free(dataType)
        e)free(moreList1)
   }
14)moreList,EPSILON
={
        //bottom up
        a)moreList.syn = NULL
}
15)output_plist,ID,COLON,type,moreOutput
=
{
        //bottom-up
        a)output_plist.syn = insert_at_beginning(moreOutput.syn,make_pair(ID.addr,type.addr))
        b)free(COLON)
        c)free(type)
        d)free(moreOutput)
}
16)output_plist,EPSILON
= {
        //bottom-up
```

a)output_plist.syn = NULL
b)free(EPSILON)
}
17)moreOutput,COMMA,ID,COLON,type,moreOutput1
= {
//bottom up
a)moreOutput.syn = insert_at_beginning(moreOutput1.syn,make_pair(ID.addr,type.addr))
b)free(COMMA)
c)free(COLON)
d)free(type)
e)free(moreOutput1)
}

18)moreOutput,EPSILON
={
//bottom up
a)moreOutput.syn = NULL
}

19)dataType,INTEGER
= {
//bottom up
a) dataType.addr=INTEGER.addr;
}
20)dataType,REAL
= {//bottom up
a)dataType.addr=REAL.addr;
}
21)dataType,BOOLEAN
= {//bottom up
a)dataType.addr=BOOLEAN.addr;
}
22)dataType,ARRAY,SQBO,range,SQBC,OF,type
= {
//bottom-up
a)free(ARRAY);
b)free(SQBO);
c)dataType.addr = make_new_node(“ARRAY-DCL”,range.addr,type.addr);
d)free(range);
e)free(SQBC);
f)free(OF);
g)free(type);

23)type,INTEGER

= {//bottom up
a)type.addr=INTEGER.addr;
}
24)type,REAL
= {//bottom up
a)type.addr=REAL.addr;
}
25)type,BOOLEAN
= {//bottom up
a)type.addr=BOOLEAN.addr;
}
26)moduleDef,START,statements,END
= {
//bottom up
a)free(START)
b)free(END)
c)moduleDef.syn = statements.syn
d)free(statements)
}
27)statements,statement,statements1
={
//bottom up
a)statements.syn = insert_at_beginning(statements1.syn,statement.addr)
b)free(statement)
c)free(statements1)
}
28)statements,EPSILON
={
//bottom up
a)statements.syn = NULL
}

29)statement,ioStmt
= {//bottom up
a)statement,addr = ioStmt.addr;
b)free(ioStmt);
}
30)statement,simpleStmt
= {//bottom up
a)statement.addr=simpleStmt.addr;
b)free(simpleStmt);
}
31)statement,declareStmt
=

{
	//bottom up
	a)statement.addr = declareStmt.addr
	b)free(declareStmt)
}
32)statement,conditionalStmt
= {//bottom up
	a)statement.addr=conditionalStmt.addr;
	b)free(conditionalStmt);
  }
33)statement,iterativeStmt
={//bottom up
	a)statement.addr = iterativeStmt.addr;
	b)free(iterativeStmt);
}
34)ioStmt,GET_VALUE,BO,ID,BC,SEMICOL
={//bottom up
	a)free(GET_VALUE);
	b)free(BO)
	c)ioStmt.addr = make_new_node(“GET-VALUE”,ID.addr);
	d)free(BC);
	e)free(SEMICOL);
}
35)ioStmt,PRINT,BO,print_var,BC,SEMICOL
={//bottom up
	 a)free(PRINT);
	 b)free(BO)
	 c)ioStmt.addr = make_new_node(“PRINT”,print_var.addr);
	 d)free(print_var);
	 e) free(BC);
	 f)free(SEMICOL);
}
36)print_var,ID,whichId2
={//bottom up
	a)print_var.syn = ID.addr;
	b)whichId2.inh = print_var.syn; //top-down
	c)print_var.addr = whichId2.addr;
	d)free(whichId2);
}
37)print_var,boolvar
= { //bottom up
	 a)print_var.addr = boolvar.addr;
	 b)free(boolvar)
  }

38)print_var,NUM
= {//bottom up
a)print_var.addr=NUM.addr;
}
39)print_var,RNUM
= {//bottom up
a)print_var.addr=RNUM.addr;
}
40)whichId2,SQBO,sign,index,SQBC
={//bottom up
a)free(SQBO);
b)free(SQBC);
c)whichId2.addr = make_new_node(“ARR-PRINT”,
whichId2.inh,make_new_node(“INDEX”,sign.addr,index.addr));
d)free(sign);
e)free(index);
}
41)whichId2,EPSILON
= {//bottom up
a)whichId2.addr = whichId2.inh;
b)free(EPSILON);
}
42)boolvar,TRUE
={//bottom up
a)boolvar.addr = TRUE.addr;
}
43)boolvar,FALSE
={//bottom up
a)boolvar.addr = FALSE.addr;

}
44)whichId,SQBO,newArithmeticExpr,SQBC
={//bottom up
a)whichId.addr
=make_new_node(“ARRAY_ACCESS”,whichId.inh,newArithmeticExpr.addr)
b)free(newArithmeticExpr)
c)free(SQBO)
d)free(SQBC)
}
45)whichId,EPSILON
= {//bottom up
a)free(EPSILON)
b) whichId.addr = whichId.inh

}
46)index,NUM
= { //bottom up
a)index.addr = NUM.addr;
}
47)index,ID
= { //bottom up
a)index.addr = ID.addr;
}
48)sign,PLUS
= {//bottom up
a)sign.addr = PLUS.addr;
}
49)sign,MINUS
= { //bottom up
a)sign.addr = MINUS.addr;
}
50)sign,EPSILON
= {//bottom up
a)sign.addr = NULL;
}
51)aVar,ID,whichId
={
a)aVar.syn = ID.addr //bottom up
b)whichId.inh= aVar.syn //top-down
c) aVar.addr = whichId.addr //bottom up
d)free(whichID)
}
52)aVar,NUM
= {//bottom up
a)aVar.addr = NUM.addr;
}
53)aVar,RNUM
= {//bottom up
a)aVar.addr = RNUM.addr;
}
54)var_id_num,NUM
= {//bottom up
a)var_id_num.addr = NUM.addr;
}
55)var_id_num,RNUM
= {//bottom up
a)var_id_num.addr = RNUM.addr;
}

56)var_id_num,ID
= {//bottom up
a)var_id_num.addr = ID.addr;
}
57)newArithmeticExpr,u1
={
a)newArithmeticExpr.addr = u1.addr // bottom up
b)free(u1) //bottom up
}
58)newArithmeticExpr,startExpr
={
a)newArithmeticExpr.addr = startExpr.addr //bottom up
b)free(startExpr) //bottom up
}
59)startExpr,newTerm,newA1
= {
a)startExpr.syn = newTerm.addr; //bottom-up
b)newA1.inh = startExpr.syn; //top-down
c)startExpr.addr = newA1.syn; //bottom-up
d)free(newTerm); //bottom up
e)free(newA1); //bottom up
}

60)newA1,op1,newTerm,newA11
= {
a)newA1.addr = make_new_node(op1.addr.value,newA1.inh,newTerm.addr) // bottom up
b)newA11.inh = newA1.addr; //top-down
c)newA1.syn = newA11.syn; //bottom-up
d)free(newTerm);
e)free(newA11);
f)free(op1);
}

61)newA1,EPSILON
= {
a)newA1.syn = newA1.inh; //bottom-up
b)free(EPSILON); //bottom up
}
62)newTerm,newNextTerm,newA2
= {
a)newA2.inh = newTerm.syn; //top-down
b)newTerm.syn = newNextTerm.addr; //bottom-up
c)newTerm.addr = newA2.syn; //bottom-up
d)free(newA2); //bottom up

```
        e)free(newNextTerm); //bottom up
 }

63)newA2,op2,newNextTerm,newA21
= {
        //bottom-up
        a)newA2.addr = make_new_node(op2.addr.value,newA2.inh,newNextTerm.addr);
        b)newA21.inh = newA2.addr; //top-down
        c)newA2.syn = newA21.syn; //bottom up
        d)free(op2); //bottom up
        e)free(newNextTerm); //bottom up
        f)free(newA21); //bottom up
 }

64)newA2,EPSILON
= {//bottom up
        a)free(EPSILON);
        b)newA2.syn = newA2.inh;
 }
65)newNextTerm,BO,startExpr,BC
=
{//bottom up
        a)newNextTerm.addr = startExpr.addr
        b)free(startExpr)
        c)free(BO)
        d)free(BC)
}
66)newNextTerm,var_id_num2
=
{//bottom up
        a)newNextTerm.addr = var_id_num2.addr
        b)free(var_id_num2)
}
67)u1,MINUS,after_u1
= {     //bottom up
        a)u1.addr = make_new_node("U1_MINUS",MINUS.addr,after_u1.addr);
        b)free(after_u1);
  }

68)u1,PLUS,after_u1
= {     //bottom up
        a)u1.addr = make_new_node("U1_PLUS",PLUS.addr,after_u1.addr);
        b)free(after_u1);
  }
```

69)after_u1,BO,startExpr,BC
={
	//bottom-up
	a)after_u1.addr = startExpr.addr
	b)free(BO)
	c)free(BC)
	d)free(startExpr)
 }
70)after_u1,var_id_num2
={//bottom up
	a)after_u1.addr = var_id_num2.addr
	b)free(var_id_num2)
  }
71)var_id_num2,NUM
={//bottom up
	a)var_id_num2.addr = NUM.addr
}
72)var_id_num2,ID
={//bottom up
	a)var_id_num2.addr = ID.addr
}
73)simpleStmt,assignmentStmt
= {//bottom up
	a)simpleStmt.addr = assignmentStmt.addr
	b) free(assignmentStmt)
  }
74)simpleStmt,moduleReuseStmt
 = {//bottom up
	a)simpleStmt.addr = moduleReuseStmt.addr
	b) free(moduleReuseStmt)
  }

75)assignmentStmt,ID,whichStmt
= {
	a)assignmentStmt.syn = ID.addr// bottom up
	b) whichStmt.inh = assignmentStmt.syn//top down
	c) assignmentStmt.addr = whichStmt.addr //bottom up
	d) free(whichStmt) //bottom up
 }
76)whichStmt,lvalueIDStmt
={	a)lvalueIDStmt.inh = whichStmt.inh //top down
	b)whichStmt.addr = lvalueIDStmt.addr //bottom up
	c)free(lvalueIDStmt) //bottom up
}

77)whichStmt,lvalueARRStmt
= {
a)lvalueARRStmt.inh = whichStmt.inh //top-down
b)whichStmt.addr = lvalueARRStmt.addr //bottom up
c)free(lvalueARRStmt)//bottom up
}
78)lvalueIDStmt,ASSIGNOP,expression,SEMICOL
= {
a)lvalueIDStmt.addr = make_new_node(“LVALUEID”,lvalueIDStmt.inh,expression.addr)
//bottom up
b) free(expression) //bottom up
c)free(ASSIGNOP) //bottom up
d)free(SEMICOL) //bottom up
}

79)lvalueARRStmt,SQBO,newArithmeticExpr,SQBC,ASSIGNOP,expression,SEMICOL
= {
a)lvalueARRStmt.addr =
make_new_node(“LVALUEARRAY”,make_new_node(“ARRAY_ACCESS”,lvalueARRStmt.inh,newArithmeti
cExpr.addr),expression.addr) //bottom up
b)free(SQBO) //bottom up
c)free(newArithmeticExpr) //bottom up
d)free(SQBC) //bottom up
e)free(ASSIGNOP) //bottom up
f)free(expression) //bottom up
g)free(SEMICOL) //bottom up
}
80)moduleReuseStmt,optional,USE,MODULE,ID,WITH,PARAMETERS,idList,SEMICOL
=
{
a)moduleReuseStmt=make_new_node(“MODULE-INVOKE”,ID.addr,(make_new_node(“ARGUME
NTS”,optional.syn,idList.syn)); //bottom up
b)free(USE); //bottom up
c)free(MODULE); //bottom up
d)free(WITH); //bottom up
e)free(PARAMETERS); //bottom up
f)free(SEMICOL); //bottom up
g)free(optional);
h)free(idList);
}
81)optional,SQBO,idList2,SQBC,ASSIGNOP
={//bottom up
a)optional.syn = idList2.syn; //bottom up
b)free(SQBO);

c)free(SQBC);
d)free(ASSIGNOP);
e)free(idList2);
}
82)optional,EPSILON
={//bottom up
a)optional.syn = NULL;
b)free(EPSILON);
}
83)idList,actualParameter,moreId
=
{ //bottom up
a)idList.syn = insert_at_beginning(moreId.syn,actualParameter.addr);
b)free(actualParameter);
c)free(moreId);
}
84)moreId,COMMA,actualParameter,moreId1
={ //bottom up
a)free(COMMA);
b)moreId.syn = insert_at_beginning(moreId1.syn,actualParameter.addr);
c)free(actualparameter);
d)free(moreId1);
}
85)moreId,EPSILON
={ //bottom up
a)moreId.syn = NULL;
b)free(EPSILON);
}
86)actualParameter,sign,aVar
={ //bottom up
a)actualParameter.addr = make_new_node(“PARAMETER_NUM”,sign.addr,aVar.addr);
b)free(sign);
c)free(aVar);
}
87)actualParameter,boolvar
={ //bottom up
a)actualParameter.addr = boolVar.addr;
b)free(boolVar);
}
88)expression,arithmeticOrBooleanExpr
= { //bottom up
a)expression.addr = arithmeticOrBooleanExpr.addr;
b)free(arithmeticOrBooleanExpr);
}

89)expression,u
= {//bottom up
a)expression.addr = u.addr;
b)free(u);
}
90)arithmeticOrBooleanExpr,anyTerm,ab1
= {
a)arithmeticOrBooleanExpr.syn = anyTerm.addr; //bottom-up
b)ab1.inh = arithmeticOrBooleanExpr.syn; //top-down
c)arithmeticOrBooleanExpr.addr = ab1.syn; //bottom-up
d)free(ab1); //bottom up
e)free(anyTerm); //bottom up
}
91)ab1,bop,anyTerm,ab11

= {
a)ab1.addr = make_new_node(bop.addr,ab1.inh,anyTerm.addr); //bottom-up
b)ab11.inh = ab1.addr; //top-down
c)ab1.syn = ab11.syn; //bottom-up
d)free(anyTerm); //bottom up
e)free(ab11); //bottom up

}
92)ab1,EPSILON
= {
a)ab1.syn = ab1.inh; //bottom up
b)free(EPSILON); //bottom up
}
93)anyTerm,arithmeticExpr,ab2
= {
a)anyTerm.syn = arithmeticExpr.addr; //bottom-up
b)ab2.inh = anyTerm.syn; //top-down
c)anyTerm.addr =ab2.addr; //bottom-up
d)free(arithmeticExpr); //bottom up
e)free(ab2); //bottom up
}
94)anyTerm,boolvar
= {
a)anyTerm.addr = boolvar.addr //bottom up
b)free(boolvar); //bottom up
}
95)ab2,relationalOp,arithmeticExpr
= { //bottom up

a)ab2.addr = make_new_node(relationalOp.addr,ab2.inh,arithmeticExpr.addr);
b)free(relationalOp);
c)free(arithmeticExpr);
}
96)ab2,EPSILON
= {//bottom up
a)ab2.addr = ab2.inh;
b)free(EPSILON);
}
97)u,MINUS,after_unary
= {//bottom up
a)u.addr = make_new_node("U_MINUS",MINUS.addr,after_unary.addr);
b)free(after_unary);
}
98)u,PLUS,after_unary
= {//bottom up
a)u.addr = make_new_node("U_PLUS",PLUS.addr,after_unary.addr);
b)free(after_unary);
}
99)after_unary,BO,arithmeticExpr,BC
= {//bottom up
a)after_unary.addr = arithmeticExpr.addr;
b)free(BO);
c)free(BC);
d)free(arithmeticExpr)
}
100)after_unary,var_id_num
= {
a)after_unary.addr = var_id_num.addr; //bottom-up
b)free(var_id_num);
}
101)arithmeticExpr,term,a1
= {
a)arithmeticExpr.syn = term.addr; //bottom-up
b)a1.inh = arithmeticExpr.syn; //top-down
c)arithmeticExpr.addr = a1.syn; //bottom-up
d)free(term);
e)free(a1);
}
102)a1,op1,term,a11
= {
a)a1.addr = make_new_node(op1.addr.value,a1.inh,term.addr); //bottom-up,.
b)a11.inh = a1.addr; //top-down
c)a1.syn = a11.syn; //bottom-up

d)free(term);
e)free(a11);
f)free(op1);
}
103)a1,EPSILON
= {//bottom up
a)a1.syn = a1.inh;
b)free(EPSILON);
}
104)term,nextTerm,a2
= {
a)a2.inh = term.syn; //top-down
b)term.syn = nextTerm.addr; //bottom-up
c)term.addr = a2.syn; //bottom-up
d)free(a2);
e)free(nextTerm);
}
105)a2,op2,nextTerm,a21
= {
a)a2.addr = make_new_node(op2.addr.value,a2.inh,nextTerm.addr); //bottom-up
b)a21.inh = a2.addr; //top-down
c)a2.syn = a21.syn; //bottom up
d)free(op2); //bottom up
e)free(nextTerm);
f)free(a21);
}
106)a2,EPSILON
= {//bottom up
a)free(EPSILON);
b)a2.syn = a2.inh;
}
107)nextTerm,BO,arithmeticOrBooleanExpr,BC
= {//bottom up
a)free(BO);
b)nextTerm.addr = arithmeticOrBooleanExpr.addr; //bottom-up
c)free(BC);
d)free(arithmeticOrBooleanExpr);
}
108)nextTerm,aVar
= {
a)nextTerm.addr = aVar.addr; //bottom-up
b)free(aVar);
}
109)op1,PLUS

= {//bottom up
	a)op1.addr = PLUS.addr;
 }
110)op1,MINUS
= {//bottom up
	a)op1.addr = MINUS.addr;
 }
111)op2,MUL
= {//bottom up
	a)op2.addr = MUL.addr;
 }
112)op2,DIV
= {//bottom up
	a)op2.addr = DIV.addr;
 }
113)bop,AND
= {//bottom up
	a)bop.addr = AND.addr;
 }
114)bop,OR
= {//bottom up
	a)bop.addr = OR.addr;
 }
115)relationalOp,LT
= {//bottom up
	a)relationalOp.addr = LT.addr;
 }
116)relationalOp,LE
= {//bottom up
	a)relationalOp.addr = LE.addr;
 }
117)relationalOp,GT
= {
	a)relationalOp.addr = GT.addr;
 }
118)relationalOp,GE
= {
	a)relationalOp.addr = GE.addr;
 }
119)relationalOp,EQ
= {
	a)relationalOp.addr = EQ.addr;
 }
120)relationalOp,NE

= {
a)relationalOp.addr = NE.addr;
}
121)declareStmt,DECLARE,idList2,COLON,dataType,SEMICOL
= {//bottom up
a)declareStmt.addr = make_new_node(“DECLARE”,dataType.addr,idList2.syn)
b)free(COLON)
c)free(DECLARE)
d)free(SEMICOL)
e)free(idList2)
f)free(dataType)
}
122)conditionalStmt,SWITCH,BO,ID,BC,START,caseStmt,default,END
= {//bottom up
a)free(SWITCH);
b)free(BO);
c)free(BC);
d)free(START);
e)conditionalStmt.syn = make_new_node(“SWITCH”,ID.addr,caseStmt.syn,default.syn);
f)free(END);
g)free(caseStmt);
h)free(default);
}
123)caseStmt,CASE,value,COLON,statements,BREAK,SEMICOL,post
= {//bottom up
a)free(CASE);
b)free(COLON);
c)free(BREAK);
d)free(SEMICOL);
e)caseStmt.syn = insert_at_beginning(post.syn,make_pair(value.addr,statements.syn));
f)free(post);
g)free(statements);
h)free(value);
}
124)post,caseStmt
= {//bottom up
a)post.syn = caseStmt.syn;
b)free(caseStmt);
}
125)post,EPSILON
= {//bottom up
a)post.syn = NULL;
b)free(EPSILON);
}

126)value,NUM
= {//bottom up
	a)value.addr = NUM.addr;
 }
127)value,TRUE
= {//bottom up
	a)value.addr = TRUE.addr;
 }
128)value,FALSE
= {//bottom up
	a)value.addr = FALSE.addr;
 }
129)default,DEFAULT,COLON,statements,BREAK,SEMICOL
= {//bottom up
	a)free(DEFAULT);
	b)free(COLON);
	c)default.syn = statements.syn;
	d)free(BREAK);
	e)free(SEMICOL);
	f)free(statements);
 }
130)default,EPSILON
= {//bottom up
	a)default.syn = NULL;
 }
131)iterativeStmt,FOR,BO,ID,IN,range_for,BC,START,statements,END
= {//bottom up
	a)free(FOR);
	b)free(BO);
	c)free(IN);
	d)free(BC);
	e)free(START);
	f)free(END);
	g)iterativeStmt.addr = make_new_node("FOR",ID.addr ,range_for.addr,statements.syn);
	h)free(statements);
	i)free(range_x);
 }
132)iterativeStmt,WHILE,BO,arithmeticOrBooleanExpr,BC,START,statements,END
= {//bottom up
	a)free(WHILE);
	b)free(BO);
	c)free(BC);
	d)free(START);
	e)free(END);

```
        f)iterativeStmt.addr =
make_new_node("WHILE",arithmeticOrBooleanExpr.addr,statements.syn);
        g)free(arithmeticOrBooleanExpr);
        h)free(statements);
 }
133)range_for,sign1,NUM1,RANGEOP,sign2,NUM2
 = {//bottom up
        a)free(RANGEOP);
        b)range_for.addr =
make_new_node("RANGE-FOR",make_new_node("LEFT-FOR",sign1.addr,NUM1.addr),make_
new_node("RIGHT-FOR",sign2.addr,NUM2.addr));
        c)free(sign1);
        d)free(sign2);
   }
134)range,sign1,index1,RANGEOP,sign2,index2
 = {//bottom up
         a)free(RANGEOP);
         b)range.addr =
make_new_node("RANGE",make_new_node("LEFT-INDEX",sign1.addr,index1.addr),make_ne
w_node("RIGHT-INDEX",sign2.addr,index2.addr));
         c)free(sign1);
         d)free(index1);
         e)free(sign2);
         f)free(index2);
   }
135)idList2,ID,moreId2
={ //bottom up
        a)idList2.syn = insert_at_beginning(moreId2.syn,ID.addr);
        b)free(moreId2);
}
136)moreId2,COMMA,ID,moreId21
={ //bottom up
        a)free(COMMA);
        b)moreId2.syn =  insert_at_beginning(moreId21.syn,ID.addr);
        c)free(moreId21);
 }
137)moreId2,EPSILON
={ //bottom up
        a)moreId2.syn = NULL;
        b)free(EPSILON);
 }
```