//I. Declaration Statement

//a.

int first;

//Leftmost -------------------------------------------------

begin => program\_stmt

=>dec\_stmt program\_stmt

=>dtype VAR\_ID dec\_expprogram\_stmt

=> INT VAR\_ID dec\_expprogram\_stmt

=>int VAR\_ID dec\_expprogram\_stmt

=>int first dec\_expprogram\_stmt

=>int first dec\_exp2program\_stmt

=>int first SEMIprogram\_stmt

=>int first;program\_stmt

=>int first; empty

=>int first;

//Rightmost -----------------------------------------------

begin => program\_stmt

=>dec\_stmt program\_stmt

=>dec\_stmt empty

=> dec\_stmt

=>dtype VAR\_ID dec\_exp

=>dtype VAR\_ID dec\_exp2

=>dtype VAR\_ID SEMI

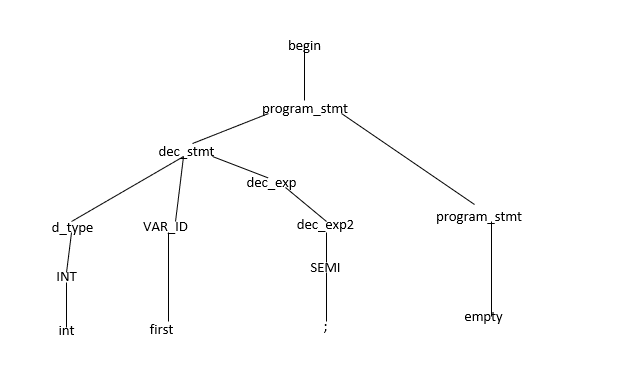
=>dtype VAR\_ID;

=>dtype first;

=> INT first;

=>int first;

**Parse Tree**

****

//b.

bool booexp = true;

//Leftmost -----------------------------------------------

start=> program\_stmt

=>dec\_stmt program\_stmt

=>dtype VAR\_ID dec\_expprogram\_stmt

=> BOOL VAR\_ID dec\_expprogram\_stmt

=>boolVAR\_ID dec\_expprogram\_stmt

=>bool booexp dec\_expprogram\_stmt

=>bool booexp EQUALS ass\_argsprogram\_stmt

=>bool booexp= ass\_argsprogram\_stmt

=>bool booexp= literal ass\_args2program\_stmt

=>bool booexp= bool\_val ass\_args2 program\_stmt

=>bool booexp= TRUE ass\_args2 program\_stmt

=>bool booexp=true ass\_args2 program\_stmt

=>bool booexp=true SEMI program\_stmt

=>bool booexp=true; program\_stmt

=>bool booexp=true; empty

=>bool booexp=true;

//Rightmost ----------------------------------------------

start=>program\_stmt

=>dec\_stmt program\_stmt

=>dec\_stmt empty

=>dec\_stmt

=>dtype VAR\_ID dec\_exp

=>dtype VAR\_ID EQUALS ass\_args

=>dtype VAR\_ID EQUALS literal ass\_args2

=>dtype VAR\_ID EQUALS literal SEMI

=>dtype VAR\_ID EQUALS literal ;

=>dtype VAR\_ID EQUALS bool\_val;

=>dtype VAR\_ID EQUALS TRUE;

=>dtype VAR\_ID EQUALS true;

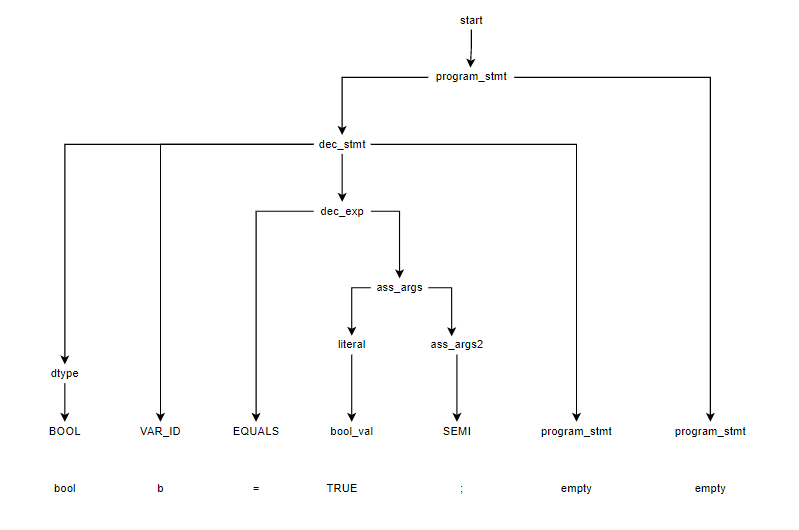
=>dtypeVAR\_ID=true;

=>dtype booexp = true;

=>BOOL\_VAL booexp = true;

=>bool booexp = true;

**Parse Tree**



//II. Input Statement

//a.

input(variable);

//Leftmost ---------------------------------------------

start => program\_stmt

=>input\_stmt program\_stmt

=>INPUT LPRN VAR\_ID RPRN SEMIprogram\_stmt

=>input LPRN VAR\_ID RPRN SEMIprogram\_stmt

=>input( VAR\_ID RPRN SEMIprogram\_stmt

=>input(variable RPRN SEMIprogram\_stmt

=>input(variable) SEMIprogram\_stmt

=>input(variable);program\_stmt

=>input(variable); empty

=>input(variable);

//Rightmost --------------------------------------------

start=>program\_stmt

=>input\_stmt program\_stmt

=>input\_stmt empty

=>input\_stmt

=>INPUT LPRN VAR\_ID RPRN SEMI

=> INPUT LPRN VAR\_ID RPRN ;

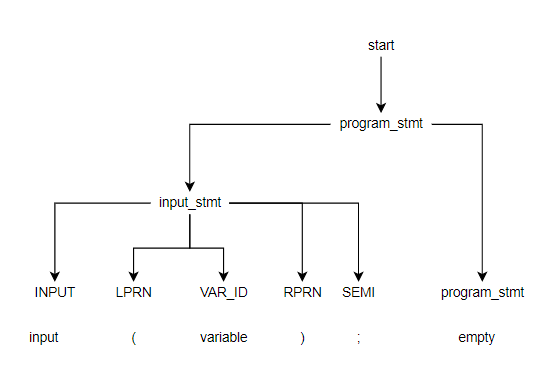
=> INPUT LPRN VAR\_ID ) ;

=> INPUT LPRN variable) ;

=> INPUT (variable) ;

=>input (variable) ;

**Parse Tree**



//III. Output Statement

//a.

output (“Hi”);

//Leftmost --------------------------------------------

start=>program\_stmt

=>output\_stmt program\_stmt

=>OUTPUT LPRN output\_exp RPRN SEMI program\_stmt

=>output LPRN output\_exp RPRN SEMI program\_stmt

=>output( output\_exp RPRN SEMI program\_stmt

=>output( output\_ext output\_concat RPRN SEMI program\_stmt

=>output( STR\_VAL output\_ext output\_concat RPRN SEMI program\_stmt

=>output(“Hi” output\_ext output\_concat RPRN SEMI program\_stmt

=>output(“Hi” empty output\_concat RPRN SEMI program\_stmt

=>output(“Hi” empty RPRN SEMI program\_stmt

=>output(“Hi”) SEMI program\_stmt

=>output(“Hi”); program\_stmt

=>output(“Hi”); empty

=>output(“Hi”);

//Rightmost -------------------------------------------

start=>program\_stmt

=>output\_stmtprogram\_stmt

=>output\_stmtempty

=>output\_stmt

=>OUTPUT LPRN output\_exp RPRN SEMI

=> OUTPUT LPRN output\_exp RPRN ;

=> OUTPUT LPRN output\_exp ) ;

=> OUTPUT LPRN output\_ext output\_concat ) ;

=> OUTPUT LPRN output\_ext empty) ;

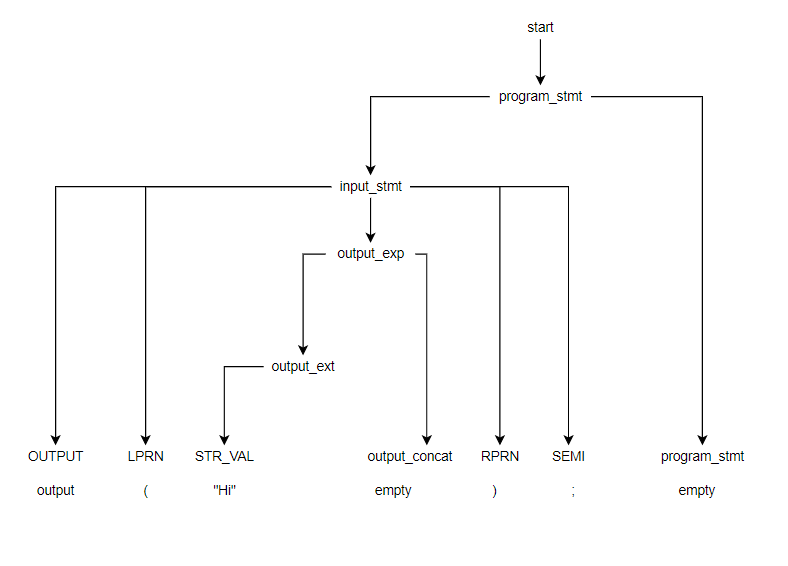
=> OUTPUT LPRN STR\_VAL output\_ext) ;

=> OUTPUT LPRN STR\_VAL exmpty) ;

=> OUTPUT LPRN “Hi”) ;

=> OUTPUT (“Hi”) ;

=>output(“Hi”) ;

**Parse Tree**

//b.

output(variable1);

//Leftmost --------------------------------------------

start=>program\_stmt

=>output\_stmt program\_stmt

=>OUTPUT LPRN output\_exp RPRN SEMI program\_stmt

=>output LPRN output\_exp RPRN SEMI program\_stmt

=>output( output\_exp RPRN SEMI program\_stmt

=>output(VAR\_ID output\_concat RPRN SEMI program\_stmt

=>output(variable1 output\_concat RPRN SEMI program\_stmt

=>output(variable1 empty RPRN SEMI program\_stmt

=>output(variable1) SEMI program\_stmt

=>output(variable1); program\_stmt

=>output(variable1); empty

=>output(variable1);

//Rightmost -------------------------------------------

start=>program\_stmt

=>output\_stmt program\_stmt

=>output\_stmt empty

=>output\_stmt

=>OUTPUT LPRN output\_exp RPRN SEMI

=> OUTPUT LPRN output\_exp RPRN;

=> OUTPUT LPRN output\_exp );

=> OUTPUT LPRN VAR\_ID output\_concat );

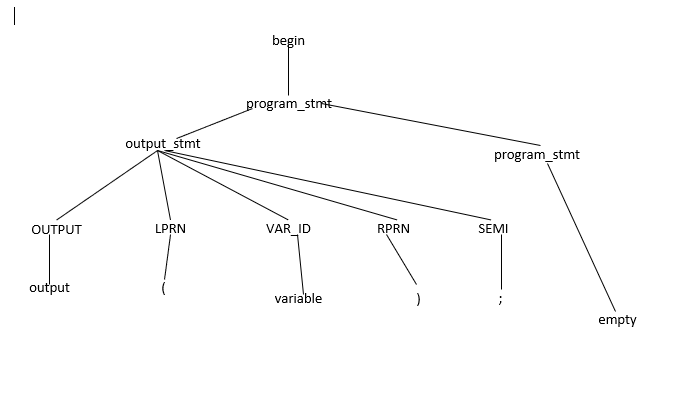
=> OUTPUT LPRN VAR\_ID empty );

=> OUTPUT LPRN variable1);

=> OUTPUT (variable1);

=>output (variable1);

**Parse Tree**



//c.

output(“Hi” + variable1);

//Leftmost --------------------------------------------

start=>program\_stmt

=>output\_stmt program\_stmt

=>OUTPUT LPRN output\_exp RPRN SEMI program\_stmt

=>output LPRN output\_exp RPRN SEMI program\_stmt

=>output( output\_exp RPRN SEMI program\_stmt

=>output( output\_ext output\_concat RPRN SEMI program\_stmt

=>output( STR\_VAL output\_ext output\_concat RPRN SEMI program\_stmt

=>output(“Hi” output\_ext output\_concat RPRN SEMI program\_stmt

=>output(“Hi” empty output\_concat RPRN SEMI program\_stmt

=>output(“Hi” PLUS output\_exp RPRN SEMI program\_stmt

=>output(“Hi” + output\_exp RPRN SEMI program\_stmt

=>output(“Hi” + VAR\_ID output\_concat RPRN SEMI program\_stmt

=>output(“Hi” + variable1 output\_concat RPRN SEMI program\_stmt

=>output(“Hi” + variable1 empty RPRN SEMI program\_stmt

=>output(“Hi” + variable1) SEMI program\_stmt

=>output(“Hi” + variable1); program\_stmt

=>output(“Hi” + variable1); empty

=>output(“Hi” + variable1);

//Rightmost -------------------------------------------

start=>program\_stmt

=>output\_stmt program\_stmt

=>output\_stmt empty

=>output\_stmt

=>OUTPUT LPRN output\_exp RPRN SEMI

=> OUTPUT LPRN output\_exp RPRN ;

=> OUTPUT LPRN output\_exp );

=> OUTPUT LPRN output\_ext output\_concat );

=> OUTPUT LPRN output\_ext PLUS output\_exp);

=> OUTPUT LPRN output\_ext PLUS VAR\_ID empty);

=> OUTPUT LPRN output\_ext PLUS variable1);

=> OUTPUT LPRN output\_ext + variable1);

=> OUTPUT LPRN STR\_VAL output\_ext + variable1);

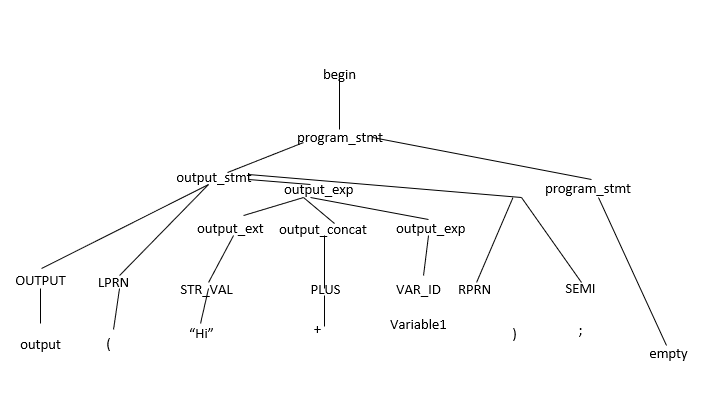
=> OUTPUT LPRN STR\_VAL empty + variable1);

=> OUTPUT LPRN “Hi” + variable1);

=> OUTPUT (“Hi” + variable1);

=>output (“Hi” + variable1);

**Parse Tree**



//VI. Assignment Statement

//a.

c = d;

//Leftmost --------------------------------------------

start=>program\_stmt

=>ass\_stmt program\_stmt

=>VAR\_ID ass\_exp program\_stmt

=>c ass\_exp program\_stmt

=>c EQUALS ass\_args program\_stmt

=> c = ass\_args program\_stmt

=> c = VAR\_ID ass\_args2 program\_stmt

=> c = d ass\_args2 program\_stmt

=> c = d SEMIprogram\_stmt

=> c = d;program\_stmt

=> c = d;empty

=> c = d;

//Rightmost -------------------------------------------

start=>program\_stmt

=>ass\_stmt program\_stmt

=>ass\_stmt empty

=>ass\_stmt

=>VAR\_ID ass\_exp

=> VAR\_ID EQUALS ass\_args

=> VAR\_ID EQUALS VAR\_ID ass\_args2

=> VAR\_ID EQUALS VAR\_ID SEMI

=> VAR\_ID EQUALS VAR\_ID ;

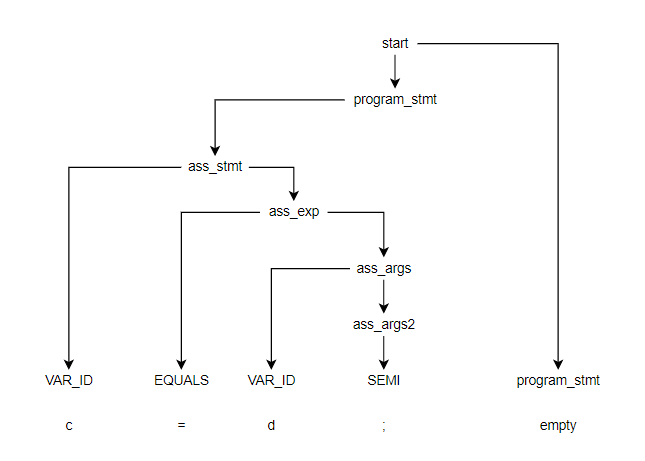
=> VAR\_ID EQUALS VAR\_ID ;

=> VAR\_ID EQUALS d ;

=> VAR\_ID = d;

=> c = d;

**Parse Tree**



//b.

number = 5;

//Leftmost --------------------------------------------

start=>program\_stmt

=>ass\_stmt program\_stmt

=>VAR\_ID ass\_exp program\_stmt

=>number ass\_exp program\_stmt

=>number EQUALS ass\_args program\_stmt

=> number = ass\_argsprogram\_stmt

=> number = literal ass\_args2 program\_stmt

=> number = INT\_VAL ass\_args2 program\_stmt

=> number = 5 ass\_args2 program\_stmt

=> number = 5 SEMIprogram\_stmt

=> number = 5; program\_stmt

=> number = 5; empty

=> number = 5;

//Rightmost -------------------------------------------

start=>program\_stmt

=>ass\_stmt program\_stmt

=>ass\_stmt empty

=>ass\_stmt

=>VAR\_ID ass\_exp

=> VAR\_ID EQUALS ass\_args

=> VAR\_ID EQUALS literal ass\_args2

=> VAR\_ID EQUALS literal SEMI

=> VAR\_ID EQUALS literal ;

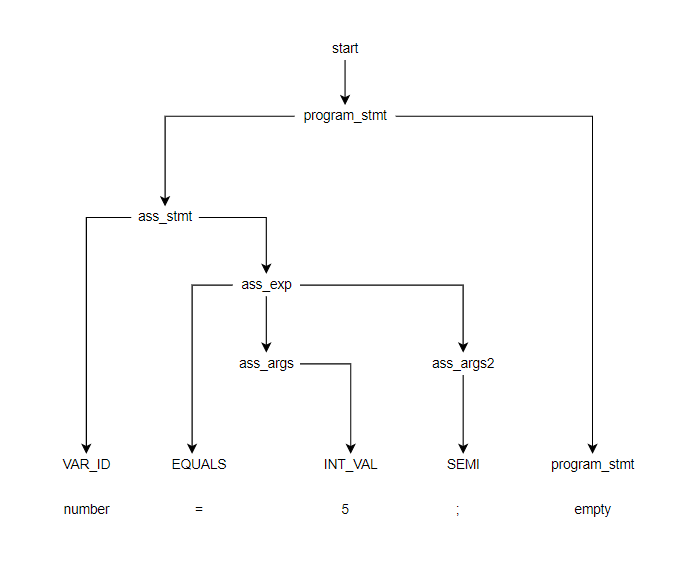
=> VAR\_ID EQUALS INT\_VAL ;

=> VAR\_ID EQUALS 5;

=> VAR\_ID = 5;

=> number = 5;

**Parse Tree**



//c.

multiply = (3\*5);

//Leftmost --------------------------------------------

start=> program\_stmt

=> ass\_stmt program\_stmt

=> VAR\_ID ass\_exp program\_stmt

=>multiply ass\_exp program\_stmt

=>multiply EQUALS ass\_args program\_stmt

=> multiply = ass\_args program\_stmt

=> multiply = LPRN ass\_args RPRN ass\_args2 program\_stmt

=> multiply = ( ass\_args RPRN ass\_args2 program\_stmt

=> multiply = (literal ass\_args2 RPRN ass\_args2 program\_stmt

=> multiply = (INT\_VAL ass\_args2 RPRN ass\_args2 program\_stmt

=> multiply = (3 ass\_args2 RPRN ass\_args2 program\_stmt

=> multiply = (3 ar\_op ass\_args RPRN ass\_args2 program\_stmt

=> multiply = (3 MULTI ass\_args RPRN ass\_args2 program\_stmt

=> multiply = (3\* ass\_args RPRN ass\_args2 program\_stmt

=> multiply = (3\* literal ass\_args2 RPRN ass\_args2 program\_stmt

=> multiply = (3\* INT\_VAL ass\_args2 RPRN ass\_args2 program\_stmt

=> multiply = (3\*5 ass\_args2 RPRN ass\_args2 program\_stmt

=> multiply = (3\*5 empty RPRN ass\_args2 program\_stmt

=> multiply = (3\*5 RPRN ass\_args2 program\_stmt

=> multiply = (3\*5) ass\_args2 program\_stmt

=> multiply = (3\*5) SEMI program\_stmt

=> multiply = (3\*5); program\_stmt

=> multiply = (3\*5); empty

=> multiply = (3\*5);

//Rightmost -------------------------------------------

start=> program\_stmt

=> ass\_stmt program\_stmt

=> ass\_stmt program\_stmt

=> ass\_stmt

=> VAR\_ID ass\_exp program\_stmt

=> VAR\_ID ass\_exp empty

=> VAR\_ID ass\_exp

=> VAR\_ID EQUALS ass\_args

=> VAR\_ID EQUALS LPRN ass\_args RPRN ass\_args2

=> VAR\_ID EQUALS LPRN ass\_args RPRN SEMI

=> VAR\_ID EQUALS LPRN ass\_args RPRN ;

=> VAR\_ID EQUALS LPRN ass\_args );

=> VAR\_ID EQUALS LPRN literal ass\_args2);

=> VAR\_ID EQUALS LPRN literal ar\_op ass\_args);

=> VAR\_ID EQUALS LPRN literal ar\_op literal ass\_args2);

=> VAR\_ID EQUALS LPRN literal ar\_op literal empty);

=> VAR\_ID EQUALS LPRN literal ar\_op INT\_VAL);

=> VAR\_ID EQUALS LPRN literal ar\_op 5);

=> VAR\_ID EQUALS LPRN literal MULTI 5);

=> VAR\_ID EQUALS LPRN literal \* 5);

=> VAR\_ID EQUALS LPRN INT\_VAL \* 5);

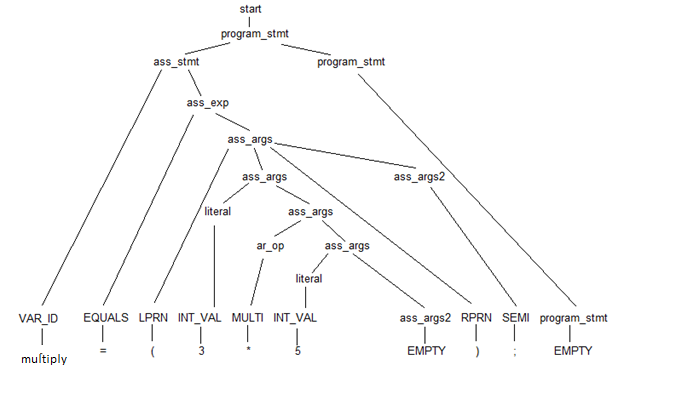
=> VAR\_ID EQUALS LPRN 3 \* 5);

=> VAR\_ID EQUALS (3 \* 5);

=> VAR\_ID = (3 \* 5);

=> multiply= (3 \* 5);

**Parse Tree**



V. Iterative

a. while(a==0)

{

Des = 100;

}

• LEFTMOST

start=>program\_stmt

=>loop\_stmt program\_stmt

=>WHILE cond\_exp

LBRC

loop\_prog

RBRC program\_stmt

=>while cond\_exp

LBRC

loop\_prog

RBRC program\_stmt

=>while LPRN cond\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(VAR\_ID rel\_op cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a rel\_op cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a EQ cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a== cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a== literal log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a== INT\_VAL log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0 log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0 empty RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0) LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0) empty program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0) empty RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0) empty

LBRC

loop\_prog

RBRC program\_stmt

=>while(a==0)

{

loop\_prog

RBRC program\_stmt

=>while(a==0)

{

program\_stmt

RBRC program\_stmt

=>while(a==0)

{

ass\_stmt program\_stmt

RBRC program\_stmt

=>while(a==0)

{

VAR\_ID ass\_exp program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des ass\_exp program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des EQUALS ass\_args program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = ass\_args program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = literal ass\_args2 program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = INT\_VAL ass\_args2 program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = 100 ass\_args2 program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = 100 SEMI program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = 100; program\_stmt

RBRC program\_stmt

=>while(a==0)

{

Des = 100; empty

RBRC program\_stmt

=>while(a==0)

{

Des = 100;

} program\_stmt

=>while(a==0)

{

Des = 100;

} empty

=>while(a==0)

{

Des = 100;

}

• RIGHTMOST

start=>program\_stmt

=>loop\_stmt program\_stmt

=>loop\_stmt empty

=>loop\_stmt

=>WHILE cond\_exp

LBRC

loop\_prog

RBRC

start=> WHILE cond\_exp

LBRC

loop\_prog

}

start=> WHILE cond\_exp

LBRC

program\_stmt

}

start=> WHILE cond\_exp

LBRC

ass\_stmt program\_stmt

}

start=> WHILE cond\_exp

LBRC

ass\_stmt empty

}

start=> WHILE cond\_exp

LBRC

VAR\_ID ass\_exp

}

start=> WHILE cond\_exp

LBRC

VAR\_ID EQUALS ass\_args

}

start=> WHILE cond\_exp

LBRC

VAR\_ID EQUALS literal ass\_args2

}

start=> WHILE cond\_exp

LBRC

VAR\_ID EQUALS literal SEMI

}

start=> WHILE cond\_exp

LBRC

VAR\_ID EQUALS literal ;

}

start=> WHILE cond\_exp

LBRC

VAR\_ID EQUALS INT\_VAL ;

}

start=> WHILE cond\_exp

LBRC

VAR\_ID EQUALS 100;

}

start=> WHILE cond\_exp

LBRC

VAR\_ID = 100;

}

start=> WHILE cond\_exp

LBRC

Des = 100;

}

start=> WHILE cond\_exp

{

Des = 100;

}

start=> WHILE LPRN cond\_args RPRN LBRC program\_stmt RBRC

{

Des = 100;

}

start=> WHILE LPRN cond\_args RPRN LBRC program\_stmt empty

{

Des = 100;

}

start=> WHILE LPRN cond\_args RPRN LBRC empty

{

Des = 100;

}

start=> WHILE LPRN cond\_args RPRN empty

{

Des = 100;

}

start=> WHILE LPRN cond\_args ) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID rel\_op cond\_args2) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID rel\_op literal log\_args) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID rel\_op literal empty) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID rel\_op INT\_VAL) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID rel\_op 0) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID EQ 0) empty

{

Des = 100;

}

start=> WHILE LPRN VAR\_ID ==0) empty

{

Des = 100;

}

start=> WHILE LPRN a==0) empty

{

Des = 100;

}

start=> WHILE (a==0) empty

{

Des = 100;

}

start=> while(a==0) empty

{

Des = 100;

}

• PARSE TREE

b. while(cond==12)

{

while(cond1==5)

{

Comp = 100;

}

}

• LEFTMOST

start=>program\_stmt

=>loop\_stmt program\_stmt

=>WHILE cond\_exp

LBRC

loop\_prog

RBRC program\_stmt

=>while cond\_exp

LBRC

loop\_prog

RBRC program\_stmt

=>while LPRN cond\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while( cond\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(VAR\_ID rel\_op cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond rel\_op cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond EQ cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond== cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond== literal log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond== INT\_VAL log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC

=>while(cond==12 log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond==12 empty RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond==12) LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond==12) empty program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond==12) empty RBRC

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond==12) empty

LBRC

loop\_prog

RBRC program\_stmt

=>while(cond==12)

{

loop\_prog

RBRC

=>while(cond==12)

{

program\_stmt

RBRC

=>while(cond==12)

{

loop\_stmt program\_stmt

RBRC

=>while(cond==12)

{

WHILE cond\_exp

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while cond\_exp

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while LPRN cond\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While( cond\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(VAR\_ID rel\_op cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1 rel\_op cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1 EQ cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1== cond\_args2 RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1== literal log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1== INT\_VAL log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1==5 log\_args RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

While(cond1==5 empty RPRN LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5) LBRC program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5) empty program\_stmt RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5) empty RBRC

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5) empty

LBRC

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

loop\_prog

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

ass\_stmt program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

VAR\_ID ass\_exp program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp ass\_exp program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp EQUALS ass\_args program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = ass\_args program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = literal ass\_args2 program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = INT\_VAL ass\_args2 program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100 ass\_args2 program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100 SEMI program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100; program\_stmt

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100; empty

RBRC program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100;

} program\_stmt

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100;

} empty

RBRC

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100;

} empty

}

• RIGHTMOST

start=> WHILE cond\_exp

LBRC

loop\_prog

RBRC

=> WHILE cond\_exp

LBRC

loop\_prog

}

=> WHILE cond\_exp

LBRC

program\_stmt

}

=> WHILE cond\_exp

LBRC

loop\_stmt program\_stmt

}

=> WHILE cond\_exp

LBRC

loop\_stmt empty

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

loop\_prog

RBRC

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

loop\_prog

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

program\_stmt

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

ass\_stmt program\_stmt

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

ass\_stmt empty

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID ass\_exp

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID EQUALS ass\_args

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID EQUALS literal ass\_args2

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID EQUALS literal SEMI

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID EQUALS literal ;

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID EQUALS INT\_VAL ;

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID EQUALS 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

VAR\_ID = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

LBRC

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE cond\_exp

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN cond\_args RPRN LBRC program\_stmt RBRC

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN cond\_args RPRN LBRC program\_stmt empty

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN cond\_args RPRN LBRC empty

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN cond\_args RPRN empty

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN cond\_args )

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID rel\_op cond\_args2)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID rel\_op literal log\_args)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID rel\_op literal empty)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID rel\_op INT\_VAL)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID rel\_op 5)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID EQ 5)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN VAR\_ID ==5)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE LPRN cond1==5)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

WHILE (cond1==5)

{

Comp = 100;

}

}

=> WHILE cond\_exp

LBRC

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE cond\_exp

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN cond\_args RPRN LBRC program\_stmt RBRC

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN cond\_args RPRN LBRC program\_stmt empty

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN cond\_args RPRN LBRC empty

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN cond\_args RPRN empty

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN cond\_args )

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID rel\_op cond\_args2)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID rel\_op literal log\_args)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID rel\_op literal empty)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID rel\_op INT\_VAL)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID rel\_op 12)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID EQ 12)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN VAR\_ID ==12)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE LPRN cond==12)

{

while(cond1==5)

{

Comp = 100;

}

}

=> WHILE (cond==12)

{

while(cond1==5)

{

Comp = 100;

}

}

=>while(cond==12)

{

while(cond1==5)

{

Comp = 100;

}

}

• PARSE TREE

VI. Condition Statement

a. if(control==‘a’)

{

Sum = 2;

}

• LEFTMOST

start=> IF cond\_exp cond\_list

=>if cond\_exp cond\_list

=>if LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if( cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(VAR\_ID rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control EQ cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control== cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control== literal log\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control== CHAR\_VAL log\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control==’a’ log\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control==’a’ empty RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(control==’a’)

LBRC

program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

ass\_stmt program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

VAR\_ID ass\_exp program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum ass\_exp program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum EQUALS ass\_args program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = ass\_args program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = literal ass\_args2 program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = INT\_VAL ass\_args2 program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = 2 ass\_args2 program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = 2 SEMI program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = 2; program\_stmt

RBRC cond\_list

=>if(control==’a’)

{

Sum = 2; empty

RBRC cond\_list

=>if(control==’a’)

{

Sum = 2;

} cond\_list

=>if(control==’a’)

{

Sum = 2;

} empty

=>if(control==’a’)

{

Sum = 2;

}

• RIGHTMOST

start=> IF cond\_exp cond\_list

=> IF cond\_exp empty

=> IF LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC

=> IF LPRN cond\_args RPRN

LBRC

program\_stmt

}

=> IF LPRN cond\_args RPRN

LBRC

ass\_stmt program\_stmt

}

=> IF LPRN cond\_args RPRN

LBRC

ass\_stmt empty

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID ass\_exp

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS ass\_args

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ass\_args2

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal SEMI

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS INT\_VAL ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS 2 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID = 2 ;

}

=> IF LPRN cond\_args RPRN

LBRC

Sum = 2 ;

}

=> IF LPRN cond\_args RPRN

{

Sum = 2 ;

}

=> IF LPRN cond\_args)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID rel\_op cond\_args2)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID rel\_op literal log\_args)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID rel\_op literal empty)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID rel\_op CHAR\_VAL)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID rel\_op ‘a’)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID EQ ‘a’)

{

Sum = 2 ;

}

=> IF LPRN VAR\_ID ==‘a’)

{

Sum = 2 ;

}

=> IF LPRN control==‘a’)

{

Sum = 2 ;

}

=> IF (control==‘a’)

{

Sum = 2 ;

}

=>if(control==‘a’)

{

Sum = 2 ;

}

• PARSE TREE

b. if(initial==‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

• LEFTMOST

start=> IF cond\_exp cond\_list

=>if cond\_exp cond\_list

=>if LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if( cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(VAR\_ID rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital EQ cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital == cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital == literal log\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital == CHAR\_VAL log\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital ==’B’ log\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital ==’B’ empty RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

LBRC

program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

ass\_stmt program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

VAR\_ID ass\_exp program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff ass\_exp program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff EQUALS ass\_args program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = ass\_args program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = literal ass\_args2 program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = INT\_VAL ass\_args2 program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = 0 ass\_args2 program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = 0 SEMI program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = 0; program\_stmt

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = 0; empty

RBRC cond\_list

=>if(Inital ==’B’)

{

Diff = 0;

} cond\_list

=>if(Inital ==’B’)

{

Diff = 0;

}

ELSE

LBRC

program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

LBRC

program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

ass\_stmt program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

VAR\_ID ass\_exp program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff ass\_exp program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff EQUALS ass\_args program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = ass\_args program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = literal ass\_args2 program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = INT\_VAL ass\_args2 program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = 100 ass\_args2 program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = 100 SEMI program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = 100; program\_stmt

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = 100; empty

RBRC

=>if(Inital ==’B’)

{

Diff = 0;

}

else

{

Diff = 100;

}

• RIGHTMOST

start=> IF cond\_exp cond\_list

=> IF cond\_exp

ELSE

LBRC

program\_stmt

RBRC

=> IF cond\_exp

ELSE

LBRC

program\_stmt

}

=> IF cond\_exp

ELSE

LBRC

ass\_stmt program\_stmt

}

=> IF cond\_exp

ELSE

LBRC

ass\_stmt empty

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID ass\_exp

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS ass\_args

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS literal ass\_args2

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS literal SEMI

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS literal ;

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS INT\_VAL;

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS 10;

}

=> IF cond\_exp

ELSE

LBRC

VAR\_ID = 10;

}

=> IF cond\_exp

ELSE

LBRC

Diff = 10;

}

=> IF cond\_exp

ELSE

{

Diff = 10;

}

=> IF cond\_exp

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

program\_stmt

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

ass\_stmt program\_stmt

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

ass\_stmt empty

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID ass\_exp

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS ass\_args

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ass\_args2

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal SEMI

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ;

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS INT\_VAL ;

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS 0;

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID = 0;

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

LBRC

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args RPRN

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN cond\_args )

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID rel\_op cond\_args2)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID rel\_op literal log\_args)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID rel\_op literal empty)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID rel\_op CHAR\_VAL)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID rel\_op ‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID EQ ‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN VAR\_ID ==‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF LPRN Initial==‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

=> IF (Initial==‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

=>if(Initial==‘B’)

{

Diff = 0;

}

else

{

Diff = 10;

}

• PARSE TREE

c. if( a == 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5;

}

else

{

Quo = 7;

}

• LEFTMOST

start => program\_stmt

=>cond\_stmt program stmt

=> IF cond\_exp cond\_list program\_stmt

=>if cond\_exp cond\_list program\_stmt

=>if LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_list program\_stmt

=>if ( cond args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram\_stmt

=>if (VAR\_ID rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list program\_stmt

=>if(a rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if (a EQ cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(a==literal log\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==INT\_VAL log\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5 log\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5 empty RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

ass\_stmt program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

VAR\_ID ass\_exp program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo ass\_exp program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo EQUALS ass\_args program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = ass\_args program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = literal ass\_args2 program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = INT\_VAL ass\_args2 program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3 ass\_args2 program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3 SEMI program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3; program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3; program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3; empty

RBRC cond\_list

=>if(a==5)

{

Quo = 3;

} cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

ELSIF cond\_exp cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif cond\_exp cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif ( cond\_args RPRN

LBRC

program\_stmt

RBRC cond\_list

=>if(a==5)

{

Quo = 3;

}

elsif (VAR\_ID rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a rel\_op cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a NEQ cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != cond\_args2 RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != literal log\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != INT\_VAL log\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1 log\_args RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1 empty RPRN

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

LBRC

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

ass\_stmt program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

VAR\_ID ass\_exp program\_stmt

RBRC cond\_list

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo ass\_exp program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo EQUALS ass\_args program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = ass\_args program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = literal ass\_args2 program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = INT\_VAL ass\_args2 program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5 SEMI program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5; program\_stmt

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5; empty

RBRC cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

} cond\_listprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

ELSE

LBRC

program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

LBRC

program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

ass\_stmt program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

VAR\_ID ass\_exp program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo ass\_exp program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo EQUALS ass\_args program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = ass\_args program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = literal ass\_args2 program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = INT\_VAL ass\_args2 program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7 ass\_args2 program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7 SEMI program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7; program\_stmt

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7; empty

RBRCprogram stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7;

}program stmt

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7;

}empty

=>if(a==5)

{

Quo = 3;

}

elsif (a != 1)

{

Quo = 5;

}

else

{

Quo = 7;

}

• RIGHTMOST

start => program\_stmt

=> IF cond\_exp cond\_list

=> IF cond\_exp cond\_list ELSIF cond\_exp cond\_list

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

program\_stmt

RBRC

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

program\_stmt

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

ass\_stmt program\_stmt

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

ass\_stmt empty

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID ass\_exp

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS ass\_args

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS literal ass\_args2

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS literal SEMI

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS literal ;

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS INT\_VAL ;

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID EQUALS 7 ;

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

VAR\_ID = 7 ;

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

LBRC

Quo = 7 ;

}

=> IF cond\_exp

ELSIF cond\_exp

ELSE

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF cond\_exp

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

program\_stmt

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

ass\_stmt program\_stmt

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

ass\_stmt empty

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID ass\_exp

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS ass\_args

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ass\_args2

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal SEMI

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS INT\_VAL ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

VAR\_ID = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

LBRC

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args RPRN

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN cond\_args )

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID rel\_op cond\_args2)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID rel\_op literal log\_args)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID rel\_op literal empty)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID rel\_op INT\_VAL)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID rel\_op 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID NEQ 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN VAR\_ID != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF LPRN a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

ELSIF (a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF cond\_exp

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

program\_stmt

RBRC

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

program\_stmt

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

ass\_stmt program\_stmt

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

ass\_stmt empty

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID ass\_exp

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS ass\_args

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ass\_args2

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal SEMI

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS literal ;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS INT\_VAL;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID EQUALS 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

VAR\_ID = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

LBRC

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args RPRN

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN cond\_args )

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID rel\_op cond\_args2)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID rel\_op literal log\_args)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID rel\_op literal empty)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID rel\_op INT\_VAL)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID rel\_op 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID EQ 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN VAR\_ID == 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF LPRN a == 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=> IF (a == 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

=>if(a == 1)

{

Quo = 3;

}

elsif(a != 1)

{

Quo = 5 ;

}

else

{

Quo = 7 ;

}

