Ammar Khan

Moazzan Sabir

Umer Abdul Khaliq

Song Young Wong

Taimoor Azam

Group 7

Module 2

**RETURN:**

**RETURN\_STMT ->** return ( EXP ) ; { RETURN\_STMT.value = return ( EXP.value ) ; }

**EXP ->** CONST { EXP.value = CONST.value }

**EXP ->** OPR OP OPR { EXP.value = OPR.value OP.value OPR.value}

**EXP ->** ID { EXP.value = ID.value }

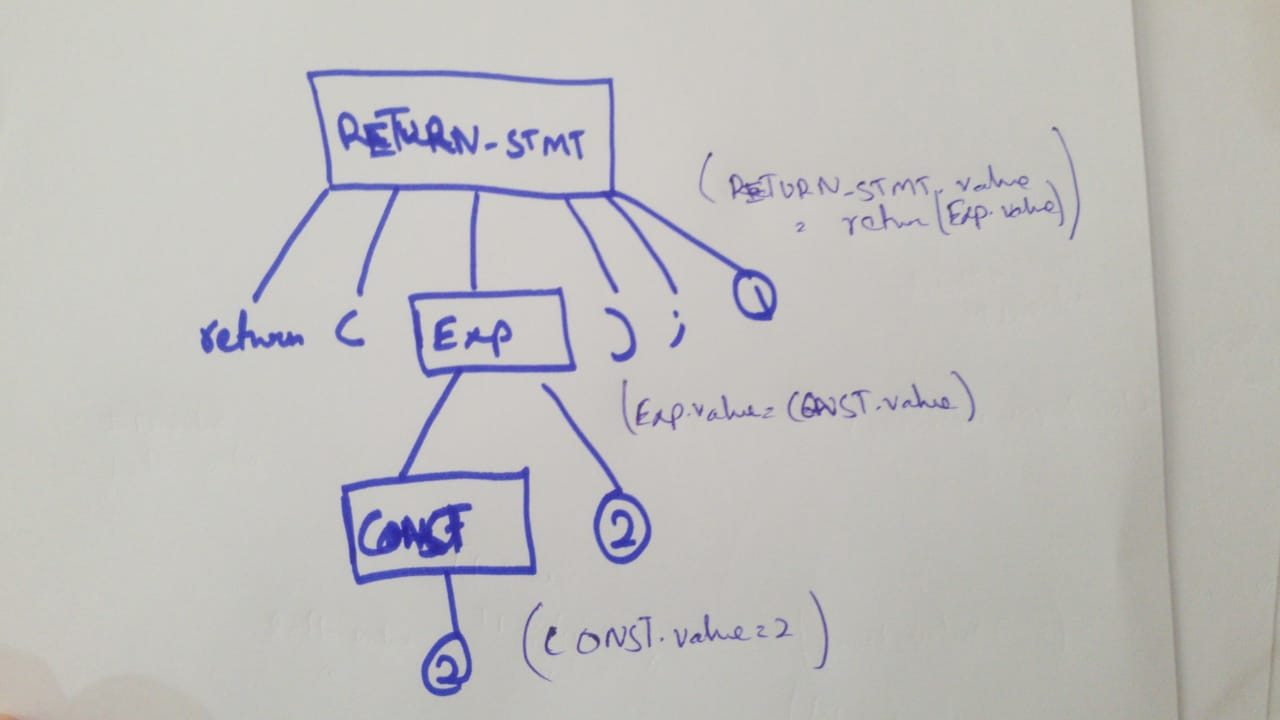
**OPR ->** CONST { OPR.value = CONST.value }

**OPR ->** ID { OPR.value = ID.value }

**OP ->** + | - | / { OP.value }

**CONST ->** 0 |…..|9|0 CONST | ….|9 CONST {CONST. value }

**Passing Value:-** return ( 2 ) ;



**EXPRESSION:**

**EXP ->** OPR OP OPR { EXP.value = OPR.value OP.value OPR.value

**EXP ->** CONST { EXP.value = CONST.value }

**EXP ->** ID { EXP.value = ID.value }

**OPR ->** CONST { OPR.value = CONST.value }

**OPR ->** ID { OPR.value = ID.value }

**OP ->** + { OP.value }

**CONST ->** 0 |..|9|0 CONST | ….|9 CONST {CONST. value}

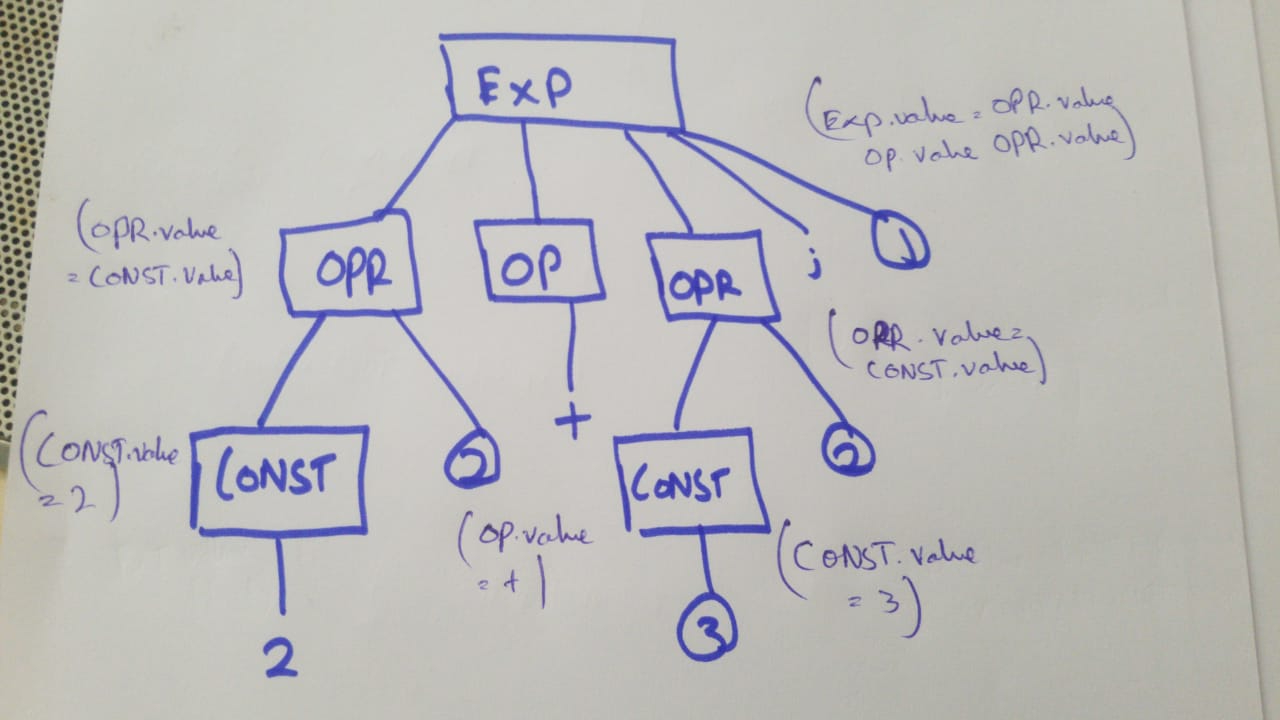
**ID->** LETTER ALPHANUM {ID.value = LETTER.value ALPHANUM.value}

**LETTER. ->** a | …| z |A |…| Z {LETTER.VALUE }

**ALPHANUM ->** a ALPHANUM | … | z ALPHANUM | A ALPHANUM | .. | Z ALPHANUM | .. | - | ⅄

**{ ALPHANUM.value }**

**Passing Value:-** 2 + 3 ;



**ASSIGNMENT STATEMENT:**

**ASSIGN\_STMT ->** ID = OPR ; { ASSIGN\_STMT.value = ID.value = OPR.value ;}

**ID->** LETTER ALPHANUM {ID.value = LETTER.value ALPHANUM.value}

**LETTER ->** a | …| z |A |…| Z {LETTER.VALUE }

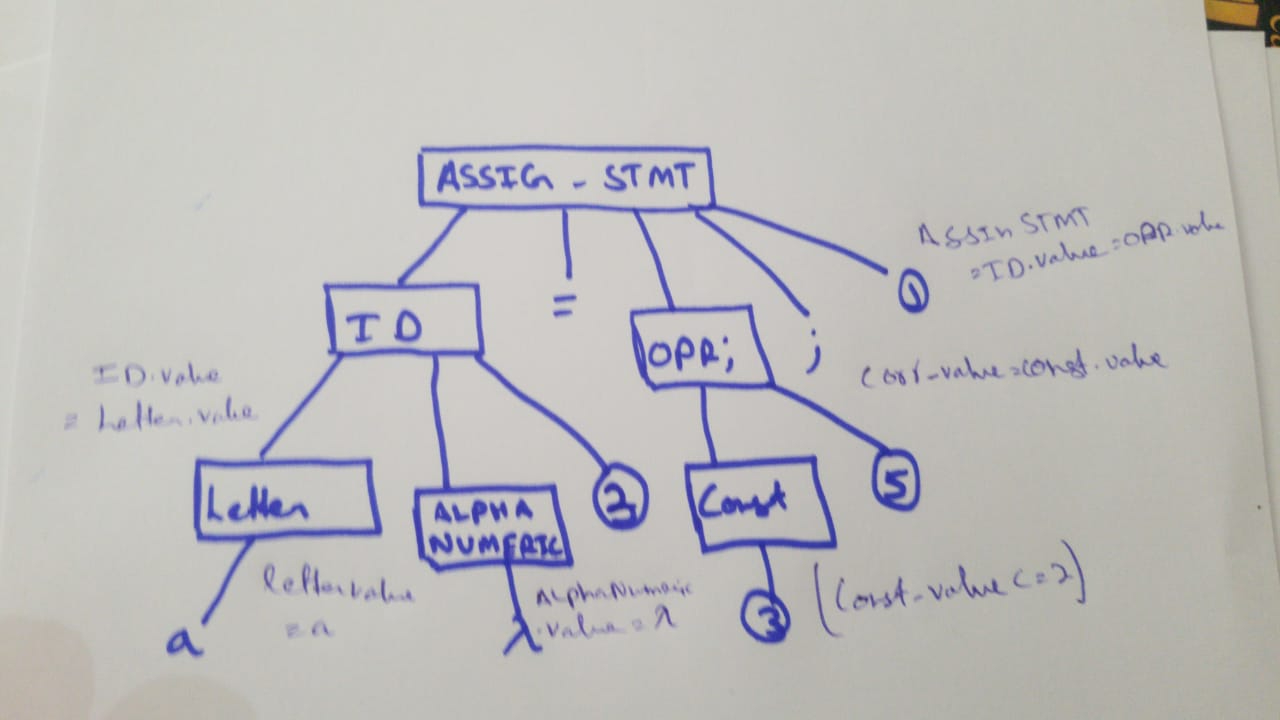
**ALPHANUM ->** a ALPHANUM | … | z ALPHANUM | A ALPHANUM | .. | Z ALPHANUM | …. | - | ⅄

**{ ALPHANUM.value }**

**OPR ->** CONST {OPR.value = CONST. value}

**CONST ->** 0 |..|9|0 CONST | ….|9 CONST {CONST. value }

**Passing Value:-** a = 2 ;



**FUNCTION:**

**FUNC ->** F\_N ( DT ID ) { STMT } { FUNC.value = F\_N.value ( DT.value ID.value ) { STMT.value }

**F\_N ->** ID { F\_N.value = ID.value }

**ID->** LETTER ALPHANUM {ID.value = LETTER.value ALPHANUM.value}

**LETTER. ->** a | …| z |A |…| Z {LETTER.VALUE }

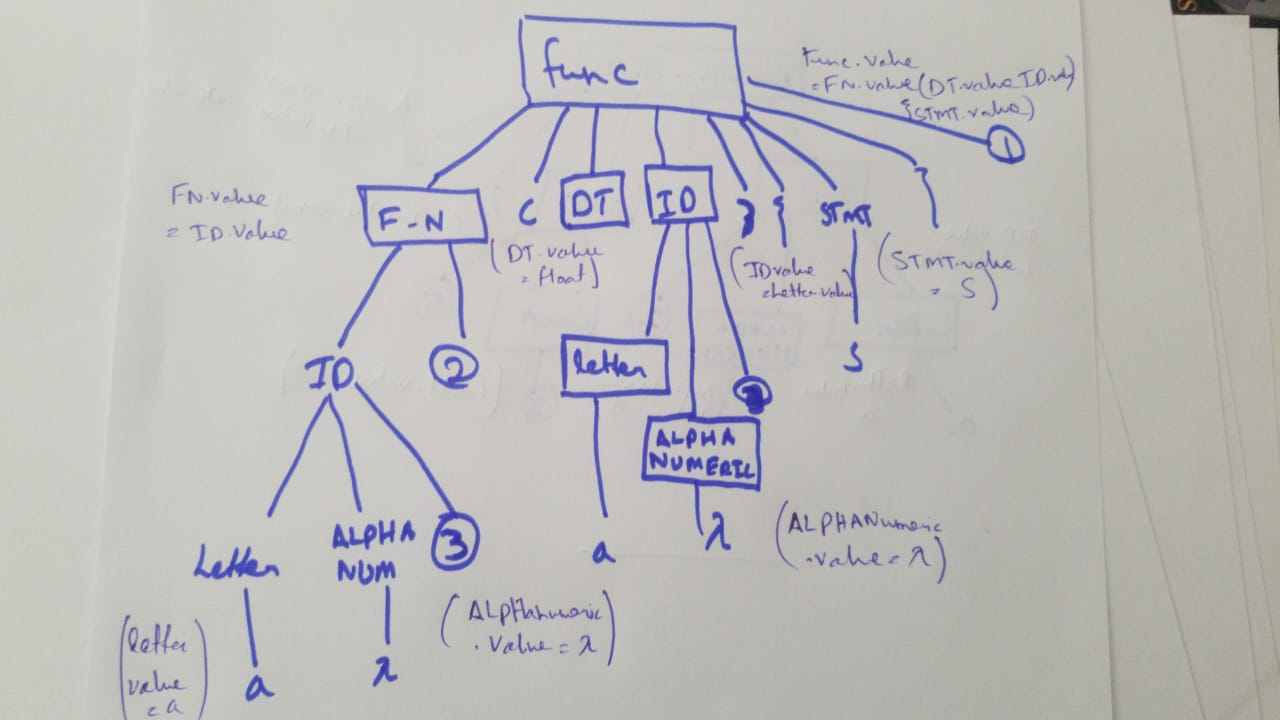
**ALPHANUM ->** a ALPHANUM | … | z ALPHANUM | A ALPHANUM | ….. | Z ALPHANUM | …. | - | ⅄

**{ ALPHANUM.value }**

**DT ->** float { DT.value }

**STMT ->** S1 | S2 | S3 { STMT.value }

**Passing Value:-** a ( float a) { s1 )



**VARIABLE DECLARATION:**

**V\_DEC ->** DT ID ;

**DT ->** float { DT.value }

**DT ->** int

**DT ->** Char

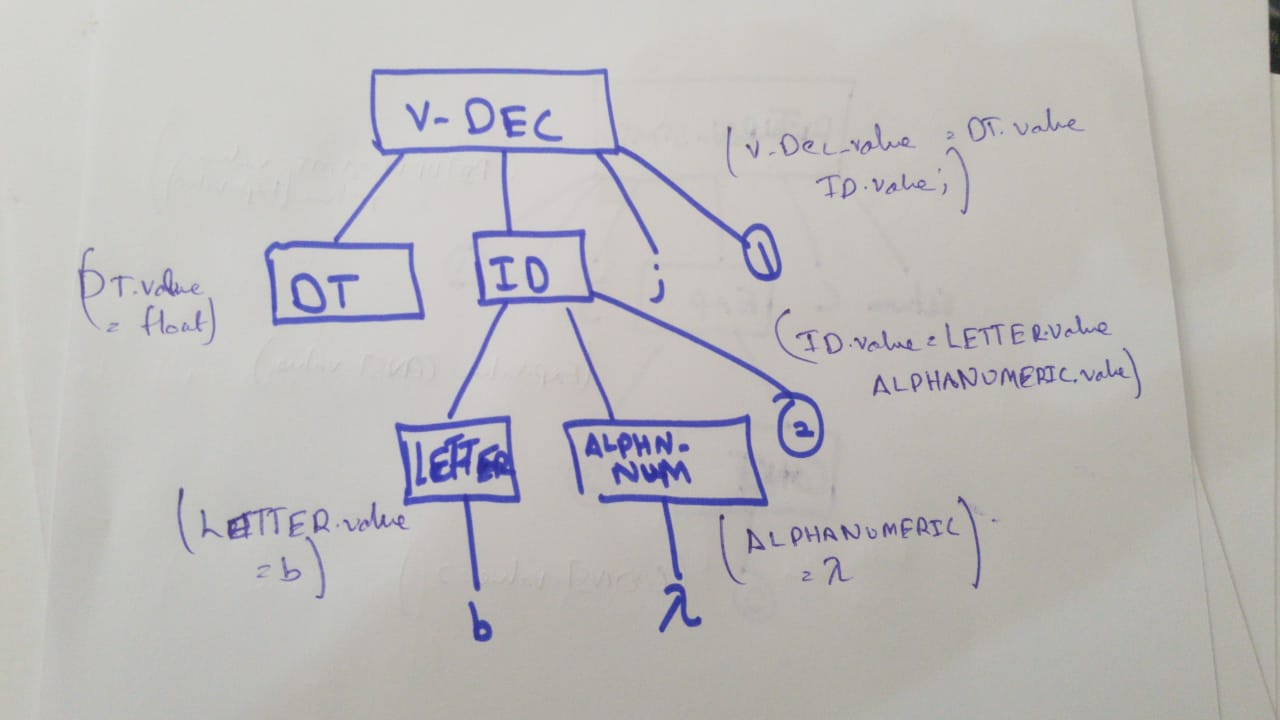
**ID->** LETTER ALPHANUM {ID.value = LETTER.value ALPHANUM.value}

**LETTER. ->** a | …| z |A |…| Z {LETTER.VALUE }

**ALPHANUM ->** a ALPHANUM | … | z ALPHANUM | A ALPHANUM |.. | Z ALPHANUM | …. | - | ⅄

**{ ALPHANUM.value }**

**Passing Value:-** float b ;



**NULL:**

**NULL-STMT->** ; {NULL\_STMT.value = ; }

**NULL\_STMT ->** ( EXP ) ; {NULL\_STMT.value = ( EXP.value ) }

**EXP->** OPR OP OPR { EXP.value = OPR.value OP.value OPR.value }

**EXP ->** CONST { EXP.value = CONST.value }

**EXP ->** ID { EXP.value = ID.value }

**OPR ->** CONST { OPR.value = CONST.value }

**OPR ->** ID { OPR.value = ID.value }

**OP ->** + { OP.value }

**CONST ->** 0 |..|9|0 CONST | ….|9 CONST {CONST. value }

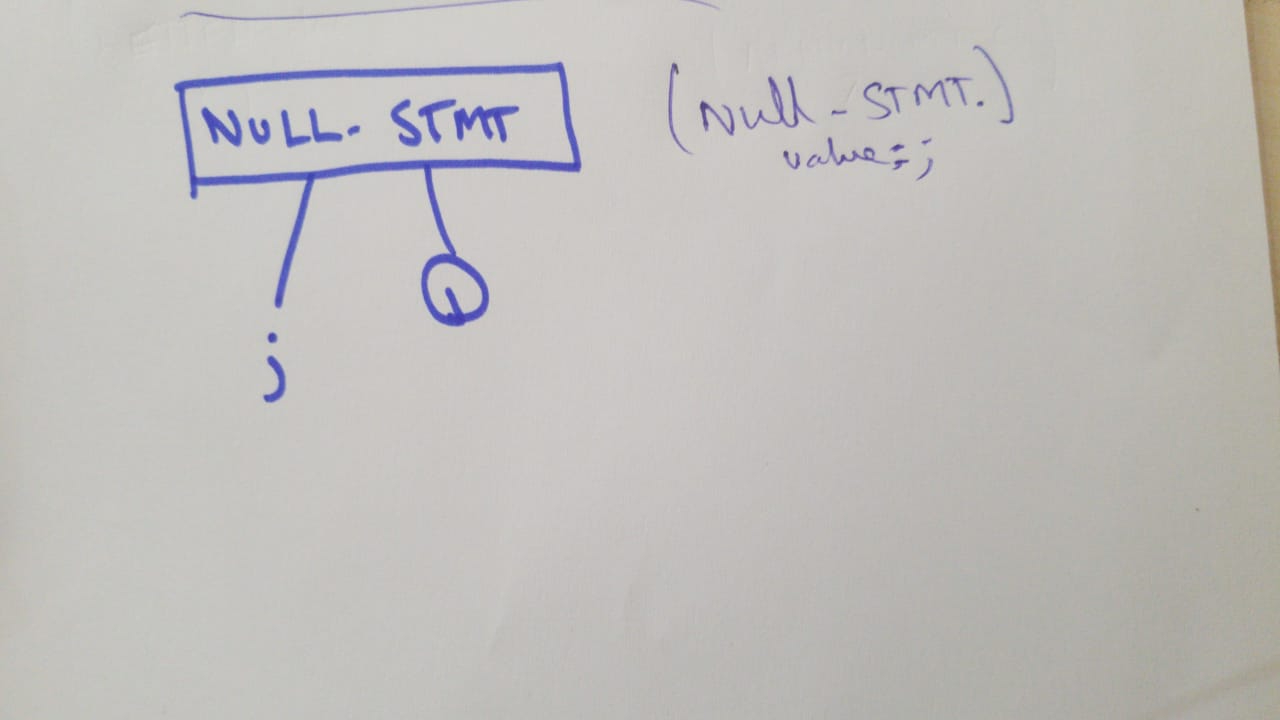
**ID->** LETTER ALPHANUM {ID.value = LETTER.value ALPHANUM.value}

**LETTER. ->** a | ..| z |A |…| Z {LETTER.VALUE = a}

ALPHANUM -> a ALPHANUM | … | z ALPHANUM | A ALPHANUM | ….. | Z ALPHANUM | …. | - | ⅄

{ ALPHANUM.value }

**Passing Value:-** ;



**WHILE STATEMENT:**

**WHILE\_STMT ->** while ( EXP ) { STMT} { WHILE\_STMT.value = while ( EXP.value ) { STMT.value } }

**EXP ->** e1{ EXP.value }

**EXP ->** e2

**EXP ->** INC\_DEC STMT ; { EXP.value = INC\_DEC STMT.value ; }

**INC\_DEC STMT ->** 1D ++ | ID -- | -- ID | ++ ID { INC\_DEC STMT.value = ID.value ++ }

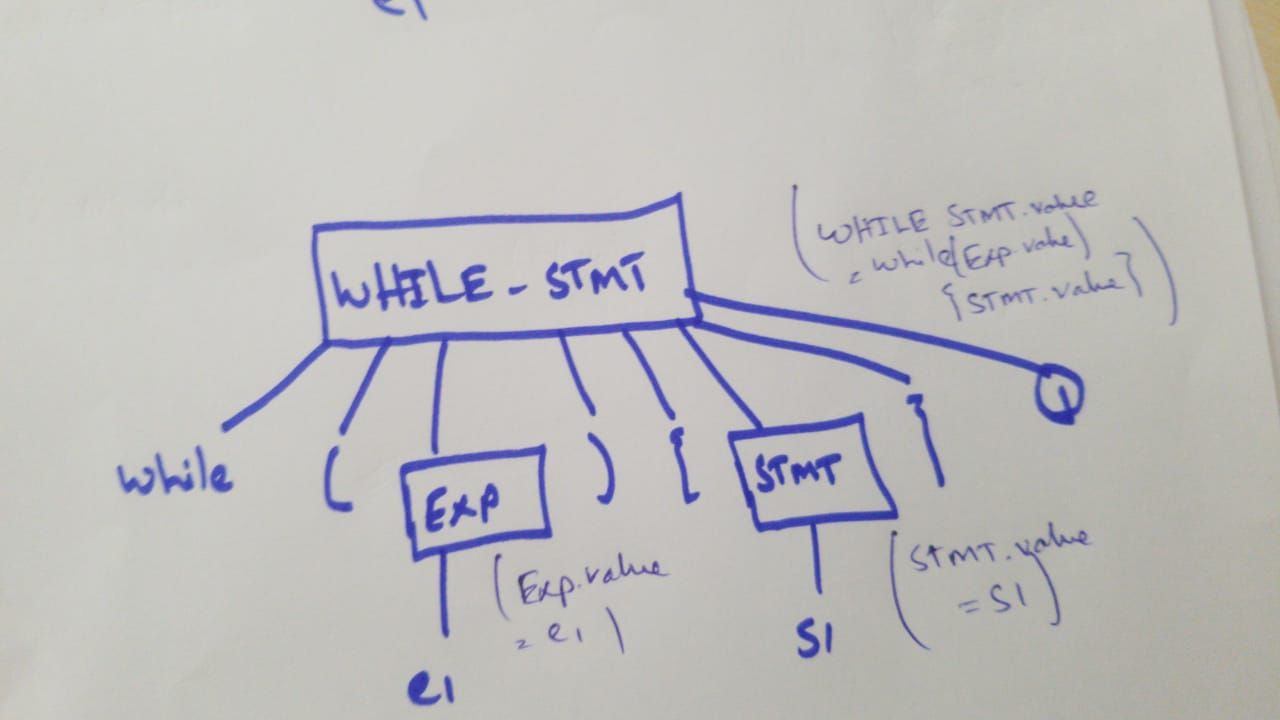
**ID ->** LETTER { ID.value = LETTER.value }

**LETTER ->** a | …| z |A |…| Z {LETTER.VALUE }

**STMT ->** S1 { STMT.value }

**STMT ->** S2

**Passing Value :-** while (e1) {s1}



**FOR STATEMENT:**

**FOR\_STMT ->** for ( EXP1 ; EXP ; EXP2 ) {STMT} {FOR\_STMT.value = For ( EXP1.value ; EXP.value ; EXP2.value ) }

**EXP1 ->** ASSIGN\_STMT { EXP1.value = ASSIGN\_STMT.value }

**EXP ->** OPR OP OPR { EXP.value = OPR.value}

**EXP2 ->** INC\_DEC { EXP2.value = INC\_DEC.value }

**STMT ->** s1 { STMT.value }

**ASSIGN\_STMT ->** 1D = OPR {ASSIGN\_STMT.value = ID.value = OPR.value }

**ID ->** LETTER ALPHANUM { ID.value = LETTER.value ALPHANUM.value }

**LETTER. ->** a | …| z |A |…| Z {LETTER.VALUE }

**ALPHANUM ->** a ALPHANUM | … | z ALPHANUM | A ALPHANUM | .. | Z ALPHANUM | …. | - | ⅄

**{ ALPHANUM.value }**

**OPR ->** CONST { OPR.value = CONST.value }

**OPR -> ID** { OPR.value = ID.value }

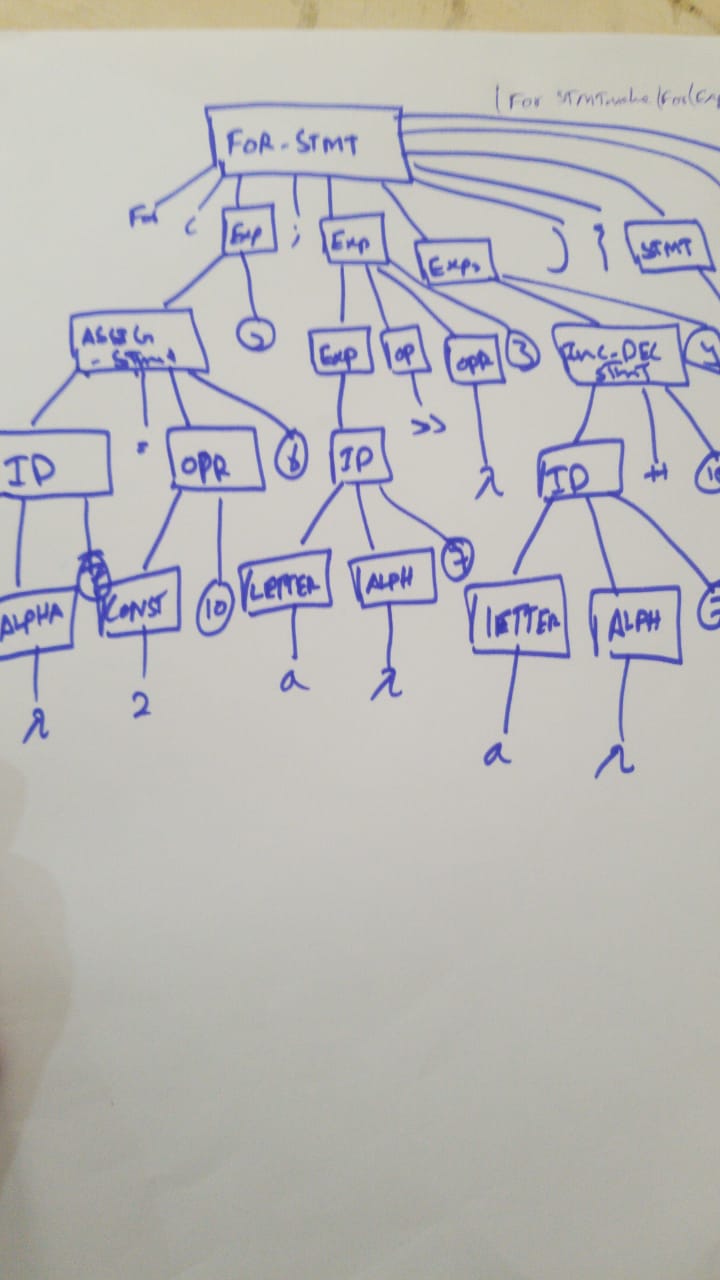
**OPR ->** ⅄ { OPR.value}

**OP ->** >>{ OP.value }

**CONST ->** 0 |..|9|0 CONST | ….|9 CONST {CONST. value }

**INC\_DEC STMT ->**1D ++ | ID -- | -- ID | ++ ID { INC\_DEC STMT.value = ID.value ++ }

**Passing Value:**- for ( a = 2 ; a >> ; a++ ) { s1}



**IF STATEMENT:**

**IF\_STMT ->** if ( EXP) then { STMT } STMT2 {IF\_STMT.value = if ( EXP.value ) then ( STMT.value ) STMT2 }

**EXP ->** e1 {EXP.value }

**EXP ->** e2

**STMT ->** s1 { STMT.value }

**STMT ->** s2

**STMT2 ->** ⅄ { STMT2.value }

**STMT2 –>** else { STMT }

**Passing Value :-** if ( e1 ) then { s1 }

