WHILE\_LOOP: while ( EXPRESSION ) Statement

if M\_expr(Expression, s) == Error

            return Error

if M\_expr(Expression, s) == False

            return s

else if M\_expr(Expression, s) == Error

            if M\_stmt(Statement, s) == Error

                        return Error

            return s = M\_stmt(Statement,s)

MeaningOf|IF\_STMT(if ( <bool\_expr> )  <stmt1> else <stmt2>, s)

if(MeaningOF|BOOL\_EXPR(<bool\_expr>, s) == error)

            return error

else if(MeaningOf|BOOL\_EXPR(<bool\_expr>, s) == true){

            if( MeaningOf|STMT(<stmt1>, s) == error)

                        return error

            return s = MeaningOf|STMT(<stmt1>, s)

} else if(MeaningOf|BOOL\_EXPR(<bool\_expr>, s) == false)

            if( MeaningOF|STMT<stmt2>, s) == error)

                        return error

            return s = MeaningOf|STMT(<stmt2>, s)

Meaningof | BOOL\_AND (<BOR1> && <BOR2>, s)

if (MeaningOf| BOOL\_OR (<BOR1>, s) == error)

return error

else if( MeaningOf | BOOL\_OR (<BOR1>, s) == TRUE) {

if (MeaningOf | BOOL\_OR (<BOR2>, s) == error)

return error

return Meaningof | BOOL\_OR (<BOR1>, s)

} else {

if (Meaningof | BOOL\_OR (<BOR2>, s) == error)

return error

return FALSE

Meaningof | BOOL\_OR (<EXPR1> && <EXPR2>, s)

if (MeaningOf|EXPR(<EXPR1>, s) == error)

return error

else if( MeaningOf|EXPR(<EXPR1>, s) == TRUE) {

if (MeaningOf|EXPR(<EXPR2>, s) == error)

return error

return Meaningof|EXPR(<EXPR1>, s)

} else {

if (Meaningof|EXPR(<EXPR2>, s) == error)

return error

return FALSE

MeaningOf|IF\_STMT (if ( <bool\_expr> ) <stmtl>, s) ===>  if (MeaningOf|BOOL\_EXPR (<bool\_expr>, s) == error)  return error

else if (Meaningof|BOOL\_EXPR (<bool\_expr>, s) == true) {  if ( MeaningOf|STMT (<stmt1>, s) == error)  return error

return s = MeaningOf|STMT(<stmt1>, s)  }else if (Meaningof|BOOL\_EXPR(<bool\_expr>, s) == false)  return s

M\_expr (E + T, s)

if M\_expr (E, s) == Error

return error

if M\_term(T, s) == Error

return Error

else return E + T

M\_expr (E - T, s)

if M\_expr (E, s) == Error

return error

if M\_term(T, s) == Error

return Error

else return E – T

M\_expr (T, s)

if M\_term(T, s) == Error

return Error

else return T

M\_term(T \* F, s)

if M\_term(T, s) == Error

return Error

if M\_fact(F, s) == Error

return Error

else return T \* F

M\_term(T / F, s)

if M\_term(T, s) == Error

return Error

if M\_fact(F, s) == Error

return Error

if M\_fact(F, s) == 0

return DivideByZeroError

else return T / F

M\_fact((E), s)

if M\_expr(E, s) == Error

return Error

else return M\_expr(E, s)

M\_fact(ID, s)

if VARMAP[ID] == undefined

return Error

else VARMAP[ID]

M\_fact(INT\_LIT, s)

return M\_int(INT\_LIT, s)

M\_fact(FLOAT\_LIT, s)

return M\_float(FLOAT\_LIT, s)