Second Phase

Negar Neda — 9331050 / Sare Soltani nejad — 9331039

PARSER.YAC

```
%{
      import java.io.*;
      %}

%token PROGRAM_KW IDENTIFIER VAR_KW INTEGER_KW REAL_KW PROCEDURE_KW END_KW
IF_KW THEN_KW ELSE_KW REPEAT_KW EXIT_KW FOR_KW INTEGER_CONSTANT TO_KW DO_KW
BEGIN_KW DOWNTO_KW REAL_CONSTANT DIV_KW MOD_KW AND_KW OR_KW NOT_KW EMPTY_KW
BOOLEAN_KW SEMICOLON_KW COLON_KW COMMA_KW SINGLE_QUOTE_KW ASS_KW LP_KW RP_KW
LB_KW RB_KW LCB_KW RCB_KW QUESTION_KW EQUALS_KW DOT_KW LE_KW LT_KW GT_KW
GE_KW EQ_KW NE_KW ADD_KW SUB_KW MUL_KW WHITE_SPACE NEWLINE BOOLEAN_CONSTANT
ERROR_NO_SHARP ERROR_ZERO COMMENTS

%code{
      static PrintStream writer;

public static void main(String args[]) throws IOException,
FileNotFoundException{
      YYParser yyparser;
final Yylex lexer;

writer = new PrintStream(new File("output.txt"));
lexer = new Yylex(new InputStreamReader(new
FileInputStream("java_code.txt")));

yyparser = new YYParser(new Lexer(){

      @Override
      public int yylex() {
      int yyl_return = -1;
      try {

            yyl_return = lexer.yylex();
      }
      catch (IOException e) {
            System.err.println("IO error :" + e);
      }
      return yyl_return;
}

@Override
public void yyerror(String error) {
      System.err.println("Error : " + error);
```

```
}

@Override
public Object getLVal() {
      return null;
}
});
yyparser.parse();

return;
}
}

%nonassoc IFX

%right "THEN_KW" "ELSE_KW"

%left OR_KW
%left AND_KW
%left EQ_KW NE_KW
%left LT_KW GT_KW LE_KW GE_KW
%left ADD_KW SUB_KW
%left MUL_KW DIV_KW MOD_KW
%right NOT_KW
%left ELSE_KW


%%

program:
PROGRAM_KW IDENTIFIER declarations procedure_list compound_statement{
      System.out.println("Rule 1: " +
      "program: PROGRAM_KW IDENTIFIER declarations procedure_list
compound_statement");
}
declarations:
VAR_KW declaration_list SEMICOLON_KW{
      System.out.println("Rule 2.1: " + "declarations: VAR_KW
declaration_list SEMICOLON_KW");
}
| {
      System.out.println("Rule 2.2: " + "declarations: empty");
}

declaration_list:
identifier_list COLON_KW type{
      System.out.println("Rule 3.1: " + "declaration_list: identifier_list
COLON_KW type");
}
|



1
```

```
declaration_list SEMICOLON_KW identifier_list COLON_KW type{
      System.out.println("Rule 3.2: " + "declaration_list: declaration_list
SEMICOLON_KW identifier_list COLON_KW type");
}

identifier_list:
IDENTIFIER{
      System.out.println("Rule 4.1: " + "identifier_list: IDENTIFIER");
}
|
identifier_list COMMA_KW IDENTIFIER{
      System.out.println("Rule 4.2: " + "identifier_list: identifier_list
COMMA_KW IDENTIFIER");
}

type:
INTEGER_KW{
      System.out.println("Rule 5.1: " + "type: INTEGER_KW");
}
|
REAL_KW{
      System.out.println("Rule 5.2: " + "type: REAL_KW");
}

procedure_list:
procedure_list procedure{
      System.out.println("Rule 6.1: " + "procedure_list: procedure_list
procedure");
}
| {
      System.out.println("Rule 6.2: " + "procedure_list: empty");
}

procedure:
PROCEDURE_KW IDENTIFIER parameters SEMICOLON_KW declarations
compound_statement SEMICOLON_KW{
      System.out.println("Rule 7: " + "procedure: PROCEDURE_KW IDENTIFIER
parameters SEMICOLON_KW declarations compound_statement SEMICOLON_KW");
}

parameters:
LP_KW declaration_list RP_KW{
      System.out.println("Rule 8.1: " +
      "parameters: LP_KW declaration_list RP_KW");
}
| {
      System.out.println("Rule 8.2: " +
            "arguments: empty");
}
```

```
compound_statement:
BEGIN_KW statement_list END_KW
{
      System.out.println("Rule 9: " + "compound_statement :BEGIN_KW
statement_list END_KW");
}


statement_list:
statement
{
      System.out.println("Rule 10.1: " + "statement_list: statement");
}
| statement_list SEMICOLON_KW statement
{
      System.out.println("Rule 10.2: " + "statement_list : statement_list
SEMICOLON_KW statement");
}


statement:
IDENTIFIER ASS_KW expression
{
      System.out.println("Rule 11.1" + "statement: IDENTIFIER ASS_KW
expression");
}
| IF_KW expression THEN_KW statement ELSE_KW statement
{
      System.out.println("Rule 11.2" + "statement : IF_KW expression THEN_KW
statement ELSE_KW statement END_KW");
}
| IF_KW expression THEN_KW statement %prec IFX
{
      System.out.println("Rule 11.3" + "statement : IF_KW expression THEN_KW
statement END_KW");
}
| REPEAT_KW statement_list EXIT_KW IF_KW expression statement_list END_KW
{
      System.out.println("Rule 11.4" + "statement : REPEAT_KW statement_list
EXIT_KW IF_KW expression statement_list END_KW");
}
| compound_statement
{
      System.out.println("Rule 11.5" + "statement : compound_statement");
}
| IDENTIFIER arguments
{
      System.out.println("Rule 11.6" + "statement : IDENTIFIER arguments");
}
```

```
| FOR_KW IDENTIFIER EQUALS_KW INTEGER_CONSTANT TO_KW INTEGER_CONSTANT DO_KW
BEGIN_KW statement_list END_KW
{
      System.out.println("Rule 11.7" + "statement : FOR_KW IDENTIFIER
EQUALS_KW INTEGER_CONSTANT TO_KW INTEGER_CONSTANT DO_KW BEGIN_KW
statement_list END_KW");
}
| FOR_KW IDENTIFIER EQUALS_KW INTEGER_CONSTANT DOWNTO_KW INTEGER_CONSTANT
DO_KW BEGIN_KW statement_list END_KW
{
      System.out.println("Rule 11.8" + "statement : FOR_KW IDENTIFIER
EQUALS_KW INTEGER_CONSTANT DOWNTO_KW INTEGER_CONSTANT DO_KW BEGIN_KW
statement_list END_KW");
}
| {
      System.out.println("Rule 11.9: " + "statement: empty");
}


arguments:
LP_KW actual_parameter_list RP_KW
{
      System.out.println("Rule 12.1: " +
      "arguments: LP_KW actual_parameter_list RP_KW");
}
| {
      System.out.println("Rule 12.2: " +
            "arguments: empty");
}


actual_parameter_list:
actual_parameter_list COMMA_KW actual_parameter
{
      System.out.println("Rule 13.1: " + "actual_parameter_list :
actual_parameter_list COMMA_KW actual_parameter");
}
| actual_parameter
{
      System.out.println("Rule 13.2: " + "actual_parameter_list :
actual_parameter");
}

actual_parameter:
expression
{
      System.out.println("Rule 14.1: " + "actual_parameter : expression");
}
expression:
INTEGER_CONSTANT{
```

```
      System.out.println("Rule 15.1: " + "expression: INTEGER_CONSTANT");
}
|
REAL_CONSTANT{
      System.out.println("Rule 15.2: " + "expression: REAL_CONSTANT");
}
|
IDENTIFIER{
      System.out.println("Rule 15.3: " + "expression: IDENTIFIER");
}
|
expression ADD_KW expression{
      System.out.println("Rule 15.4: " + "expression: expression ADD_KW
expression");
}
|
expression SUB_KW expression{
      System.out.println("Rule 15.5: " + "expression: expression SUB_KW
expression");
}
|
expression MUL_KW expression{
      System.out.println("Rule 15.6: " + "expression: expression MUL_KW
expression");
}
|
expression DIV_KW expression{
      System.out.println("Rule 15.7: " + "expression: expression DIV_KW
expression");
}
|
SUB_KW expression{
      System.out.println("Rule 15.8: " + "expression: SUB_KW expression");
}
|
expression MOD_KW expression{
      System.out.println("Rule 15.9: " + "expression: expression MOD_KW
expression");
}
| LP_KW expression RP_KW
{
      System.out.println("Rule 15.19: " + "expression : LP_KW expression
RP_KW");
}
| NOT_KW expression
{
      System.out.println("Rule 15.18: " + "expression :NOT_KW expression");
}
| expression OR_KW expression
{
```

```
      System.out.println("Rule 15.17: " + "expression :expression OR_KW
expression");
}
| expression AND_KW expression
{
      System.out.println("Rule 15.16: " + "expression :expression AND_KW
expression");
}
| expression GE_KW expression
{
      System.out.println("Rule 15.15: " + "expression :  expression GE_KW
expression");
}
| expression GT_KW expression
{
      System.out.println("Rule 15.14: " + "expression :  expression GT_KW
expression");
} | expression NE_KW expression
{
      System.out.println("Rule 15.13: " + "expression :  expression NE_KW
expression");
}
| expression EQ_KW expression
{
      System.out.println("Rule 15.12: " + "expression :  expression EQ_KW
expression");
}
| expression LE_KW expression
{
      System.out.println("Rule 15.11: " + "expression :  expression LE_KW
expression");
}
| expression LT_KW expression
{
      System.out.println("Rule 15.10: " + "expression :  expression LT_KW
expression");
}
```

FLEX.FLEX

تنها این قسمت از برنامه تغییر کرده

```
%%
{PROGRAM_KW} {
	//System.out.println(yytext() + "\t\t\t" + "PROGRAM_KW\t\t\t" + '-');
	return YYParser.PROGRAM_KW;
}

{EMPTY_KW} {
	//System.out.println(yytext() + "\t\t\t" + "EMPTY_KW\t\t\t" + '-');
	return YYParser.EMPTY_KW;
}
{VAR_KW} {
	//System.out.println(yytext() + "\t\t\t" + "VAR_KW\t\t\t" + '-');
	return YYParser.VAR_KW;
}
{INTEGER_KW} {
	//System.out.println(yytext() + "\t\t\t" + "INTEGER_KW\t\t\t" + '-');
	return YYParser.INTEGER_KW;
}
{REAL_KW} {
	//System.out.println(yytext() + "\t\t\t" + "REAL_KW\t\t\t" + '-');
	return YYParser.REAL_KW;
}

{PROCEDURE_KW} {
	//System.out.println(yytext() + "\t\t\t" + "PROCEDURE_KW\t\t\t" + '-');
	return YYParser.PROCEDURE_KW;
}
{IF_KW} {
	//System.out.println(yytext() + "\t\t\t" + "IF_KW\t\t\t" + '-');
	return YYParser.IF_KW;
}

{THEN_KW} {
	//System.out.println(yytext() + "\t\t\t" + "THEN_KW\t\t\t" + '-');
	return YYParser.THEN_KW;
}

{ELSE_KW} {
	//System.out.println(yytext() + "\t\t\t" + "ELSE_kw\t\t\t" + '-');
	return YYParser.ELSE_KW;
}
{REPEAT_KW} {
	//System.out.println(yytext() + "\t\t\t" + "REPEAT_KW\t\t\t" + '-');
	return YYParser.REPEAT_KW;
}
{EXIT_KW} {
```

```
      //System.out.println(yytext() + "\t\t\t" + "EXIT_KW\t\t\t" + '-');
      return YYParser.EXIT_KW;
}
{END_KW} {
      //System.out.println(yytext() + "\t\t\t" + "END_KW\t\t\t" + '-');
      return YYParser.END_KW;
}
{FOR_KW} {
      //System.out.println(yytext() + "\t\t\t" + "FOR_KW\t\t\t" + '-');
      return YYParser.FOR_KW;
}
{TO_KW} {
      //System.out.println(yytext() + "\t\t\t" + "TO_KW\t\t\t" + '-');
      return YYParser.TO_KW;
}
{DOT_KW} {
      //System.out.println(yytext() + "\t\t\t" + "DOT_KW\t\t\t" + '-');
      return YYParser.DOT_KW;
}
{BEGIN_KW} {
      //System.out.println(yytext() + "\t\t\t" + "BEGIN_KW\t\t\t" + '-');
      return YYParser.BEGIN_KW;
}
{DOWNTO_KW} {
      //System.out.println(yytext() + "\t\t\t" + "DOWNTO_KW\t\t\t" + '-');
      return YYParser.DOWNTO_KW;
}
{DIV_KW} {
      //System.out.println(yytext() + "\t\t\t" + "DIV_KW\t\t\t" + '-');
      return YYParser.DIV_KW;
}
{MOD_KW} {
      //System.out.println(yytext() + "\t\t\t" + "MOD_KW\t\t\t" + '-');
      return YYParser.MOD_KW;
}
{AND_KW} {
      //System.out.println(yytext() + "\t\t\t" + "AND_KW\t\t\t" + '-');
      return YYParser.AND_KW;
}
{OR_KW} {
      //System.out.println(yytext() + "\t\t\t" + "OR_KW}\t\t\t" + '-');
      return YYParser.OR_KW;
}
{NOT_KW} {
      //System.out.println(yytext() + "\t\t\t" + "NOT_KW\t\t\t" + '-');
      return YYParser.NOT_KW;
}
{SEMICOLON_KW} {
      //System.out.println(yytext() + "\t\t\t" + "SEMICOLON_KW\t\t\t" + '-');
      return YYParser.SEMICOLON_KW;
```

```
}
{COLON_KW} {
      //System.out.println(yytext() + "\t\t\t" + "COLON_KW\t\t\t" + '-');
      return YYParser.COLON_KW;
}
{COMMA_KW} {
      //System.out.println(yytext() + "\t\t\t" + "COMMA_KW\t\t\t" + '-');
      return YYParser.COMMA_KW;
}
{ASS_KW} {
      //System.out.println(yytext() + "\t\t\t" + "ASS_KW\t\t\t" + '-');
      return YYParser.ASS_KW;
}
{LP_KW} {
      //System.out.println(yytext() + "\t\t\t" + "LP_KW\t\t\t" + '-');
      return YYParser.LP_KW;
}
{RP_KW} {
      //System.out.println(yytext() + "\t\t\t" + "RP_KW\t\t\t" + '-');
      return YYParser.RP_KW;
}
{LB_KW} {
      //System.out.println(yytext() + "\t\t\t" + "LB_KW}\t\t\t" + '-');
      return YYParser.LB_KW;
}
{RB_KW} {
      //System.out.println(yytext() + "\t\t\t" + "RB_KW\t\t\t" + '-');
      return YYParser.RB_KW;
}
{LCB_KW} {
      //System.out.println(yytext() + "\t\t\t" + "LCB_KW\t\t\t" + '-');
      return YYParser.LCB_KW;
}
{RCB_KW} {
      //System.out.println(yytext() + "\t\t\t" + "RCB_KW\t\t\t" + '-');
      return YYParser.RCB_KW;
}
{QUESTION_KW} {
      //System.out.println(yytext() + "\t\t\t" + "QUESTION_KW\t\t\t" + '-');
      return YYParser.QUESTION_KW;
}
{EQUALS_KW} {
      //System.out.println(yytext() + "\t\t\t" + "EQUALS_KW\t\t\t" + '-');
      return YYParser.EQUALS_KW;
}
{LE_KW} {
      //System.out.println(yytext() + "\t\t\t" + "LE_KW\t\t\t" + '-');
      return YYParser.LE_KW;
}
{LT_KW} {
```

```
      //System.out.println(yytext() + "\t\t\t" + "LT_KW\t\t\t" + '-');
      return YYParser.LT_KW;
}
{GT_KW} {
      //System.out.println(yytext() + "\t\t\t" + "GT_KW\t\t\t" + '-');
      return YYParser.GT_KW;
}
{GE_KW} {
      //System.out.println(yytext() + "\t\t\t" + "GE_KW\t\t\t" + '-');
      return YYParser.GE_KW;
}
{EQ_KW} {
      //System.out.println(yytext() + "\t\t\t" + "EQ_KW\t\t\t" + '-');
      return YYParser.EQ_KW;
}
{NE_KW} {
      //System.out.println(yytext() + "\t\t\t" + "NE_KW\t\t\t" + '-');
      return YYParser.NE_KW;
}
{ADD_KW} {
      //System.out.println(yytext() + "\t\t\t" + "ADD_KW\t\t\t" + '-');
      return YYParser.ADD_KW;
}
{SUB_KW} {
      //System.out.println(yytext() + "\t\t\t" + "SUB_KW\t\t\t" + '-');
      return YYParser.SUB_KW;
}
{MUL_KW} {
      //System.out.println(yytext() + "\t\t\t" + "MUL_KW\t\t\t" + '-');
      return YYParser.MUL_KW;
}
{DIV_KW} {
      //System.out.println(yytext() + "\t\t\t" + "DIV_KW\t\t\t" + '-');
      return YYParser.DIV_KW;
}
{MOD_KW} {
      //System.out.println(yytext() + "\t\t\t" + "MOD_KW\t\t\t" + '-');
      return YYParser.MOD_KW;
}
{BOOLEAN_CONSTANT} {
      //System.out.println(yytext() + "\t\t\t" + "BOOLEAN_CONSTANT\t\t\t" +
'-');
      return YYParser.BOOLEAN_CONSTANT;
}
{BOOLEAN_KW} {
      //System.out.println(yytext() + "\t\t\t" + "BOOLEAN_KW\t\t\t" + '-');
      return YYParser.BOOLEAN_KW;
}
{IDENTIFIER} {
      //System.out.println(yytext() + "\t\t\t" + "IDENTIFIER\t\t\t" + '-');
```

```
      return YYParser.IDENTIFIER;
}
{INTEGER_CONSTANT} {
      //System.out.println(yytext() + "\t\t\t" + "INTEGER_CONSTANT\t\t\t" +
'-');
      return YYParser.INTEGER_CONSTANT;
}
{REAL_CONSTANT} {
      //System.out.println(yytext() + "\t\t\t" + "REAL_CONSTANT\t\t\t" + '-
');
      return YYParser.REAL_CONSTANT;
}
{ERROR_NO_SHARP} {
      //System.out.println(yytext() + "\t\t\t" + "ERROR_NO_SHARP\t\t\t" + '-
');
      return YYParser.ERROR_NO_SHARP;
}
{ERROR_ZERO} {
      //System.out.println(yytext() + "\t\t\t" + "ERROR_ZERO\t\t\t" + '-');
      return YYParser.ERROR_ZERO;
}
{COMMENTS}
{
//System.out.println(yytext() + "\t\t\t" + "COMMENTS\t\t\t" + '-');
      return YYParser.COMMENTS;
}
"\s" | "\n" | "\r" | "\t" {
}
. {
}
```

TESTCASE

```
program x
var
sare, negar : real;
num: integer;
procedure calc(x, y, z, a, b, c : integer; k: real);
var m : integer;
begin
x : = #0.1 + #12;
y: = #12 * #2;
z: = #51.0 % #39;
a: = #1;
b: = #1;
if (x.lt y) then
m : = x
else
```

```
m : = y / #2;
if (z.gt m) then
m : = z;

repeat
m : = m - #1
      exit if (m.le #1)
      m : = m + #1
      end;

c: = a and b;
if (c.eq #0) then
if (b.eq #1) then
c : = a or b
else
c: = a and b

      end;
begin
end
```

OUTPUT

```
Rule 4.1: identifier_list: IDENTIFIER
Rule 4.2 : identifier_list : identifier_list COMMA_KW IDENTIFIER
Rule 5.2 : type : REAL_KW
Rule 3.1 : declaration_list : identifier_list COLON_KW type
Rule 4.1 : identifier_list : IDENTIFIER
Rule 5.1 : type : INTEGER_KW
Rule 3.2 : declaration_list : declaration_list SEMICOLON_KW identifier_list
COLON_KW type
Rule 2.1 : declarations : VAR_KW declaration_list SEMICOLON_KW
Rule 6.2 : procedure_list : empty
Rule 4.1 : identifier_list : IDENTIFIER
Rule 4.2 : identifier_list : identifier_list COMMA_KW IDENTIFIER
Rule 4.2 : identifier_list : identifier_list COMMA_KW IDENTIFIER
Rule 4.2 : identifier_list : identifier_list COMMA_KW IDENTIFIER
Rule 4.2 : identifier_list : identifier_list COMMA_KW IDENTIFIER
Rule 4.2 : identifier_list : identifier_list COMMA_KW IDENTIFIER
Rule 5.1 : type : INTEGER_KW
Rule 3.1 : declaration_list : identifier_list COLON_KW type
Rule 4.1 : identifier_list : IDENTIFIER
Rule 5.2 : type : REAL_KW
Rule 3.2 : declaration_list : declaration_list SEMICOLON_KW identifier_list
COLON_KW type
Rule 8.1 : parameters : LP_KW declaration_list RP_KW
Rule 4.1 : identifier_list : IDENTIFIER
```

```
Rule 5.1 : type : INTEGER_KW
Rule 3.1 : declaration_list : identifier_list COLON_KW type
Rule 2.1 : declarations : VAR_KW declaration_list SEMICOLON_KW
Rule 15.2 : expression : REAL_CONSTANT
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.4 : expression : expression ADD_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.1 : statement_list : statement
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.6 : expression : expression MUL_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.2 : expression : REAL_CONSTANT
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.9 : expression : expression MOD_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.3 : expression : IDENTIFIER
Rule 15.3 : expression : IDENTIFIER
Rule 15.10 : expression : expression LT_KW expression
Rule 15.19 : expression : LP_KW expression RP_KW
Rule 15.3 : expression : IDENTIFIER
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 15.3 : expression : IDENTIFIER
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.7 : expression : expression DIV_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 11.2statement : IF_KW expression THEN_KW statement ELSE_KW statement
END_KW
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.3 : expression : IDENTIFIER
Rule 15.3 : expression : IDENTIFIER
Rule 15.14 : expression : expression GT_KW expression
Rule 15.19 : expression : LP_KW expression RP_KW
Rule 15.3 : expression : IDENTIFIER
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 11.3statement : IF_KW expression THEN_KW statement END_KW
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.3 : expression : IDENTIFIER
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.5 : expression : expression SUB_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.1 : statement_list : statement
```

```
Rule 15.3 : expression : IDENTIFIER
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.11 : expression : expression LE_KW expression
Rule 15.19 : expression : LP_KW expression RP_KW
Rule 15.3 : expression : IDENTIFIER
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.4 : expression : expression ADD_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.1 : statement_list : statement
Rule 11.4statement : REPEAT_KW statement_list EXIT_KW IF_KW expression
statement_list END_KW
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.3 : expression : IDENTIFIER
Rule 15.3 : expression : IDENTIFIER
Rule 15.16 : expression : expression AND_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 15.3 : expression : IDENTIFIER
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.12 : expression : expression EQ_KW expression
Rule 15.19 : expression : LP_KW expression RP_KW
Rule 15.3 : expression : IDENTIFIER
Rule 15.1 : expression : INTEGER_CONSTANT
Rule 15.12 : expression : expression EQ_KW expression
Rule 15.19 : expression : LP_KW expression RP_KW
Rule 15.3 : expression : IDENTIFIER
Rule 15.3 : expression : IDENTIFIER
Rule 15.17 : expression : expression OR_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 15.3 : expression : IDENTIFIER
Rule 15.3 : expression : IDENTIFIER
Rule 15.16 : expression : expression AND_KW expression
Rule 11.1statement : IDENTIFIER ASS_KW expression
Rule 11.2statement : IF_KW expression THEN_KW statement ELSE_KW statement
END_KW
Rule 11.3statement : IF_KW expression THEN_KW statement END_KW
Rule 10.2 : statement_list : statement_list SEMICOLON_KW statement
Rule 9 : compound_statement : BEGIN_KW statement_list END_KW
Rule 7 : procedure : PROCEDURE_KW IDENTIFIER parameters SEMICOLON_KW
declarations compound_statement SEMICOLON_KW
Rule 6.1 : procedure_list : procedure_list procedure
Rule 11.9 : statement : empty
Rule 10.1 : statement_list : statement
Rule 9 : compound_statement : BEGIN_KW statement_list END_KW
Rule 1 : program : PROGRAM_KW IDENTIFIER declarations procedure_list
compound_statement
```