Dumbravean Bogdan, 933

Lang.lxi:

```
%{                                  /* need this for the call to atof() below */

#include "y.tab.h"
%}

%option noyywrap

ID                          _*[a-zA-Z][a-zA-Z0-9_]*
CONSTINT        0|[+-]?[1-9][0-9]*
CONSTCHAR       ['][a-zA-Z0-9_]*[']


%%
if {return IF;}
then {return THEN;}
else {return ELSE;}
done {return DONE;}
while {return WHILE;}
in {return IN;}
do {return DO;}
read {return READ;}
write {return WRITE;}
return {return RETURN;}

{CONSTINT}                                  {return CONSTINT;}
{CONSTCHAR}                                {return CONSTCHAR;}
Number {
   return Number;
}
Boolean|String|List|Dict|and|not|or  {
   return yytext;
}
{ID}                                        {return IDENTIFIER;}
"<-" {return ASSIGN;}
"<=" {return LE;}
">=" {return GE;}
"<>" {return DIFF;}
"="|"+"|"-"|"*"|"/"|"%"|"<"|">"|"["|"]"|"("|")"|","                    {return yytext[0];}
";"                                     {return yytext[0];}

[ \t\n]+            /* eat up whitespace */
"//".*          /* eat up comments */
%%
```

Lang.y:

```
%{
#include <stdio.h>
#include <stdlib.h>
#define YYDEBUG 1
%}

%token IF
%token THEN
%token ELSE
%token DONE
%token WHILE
%token IN
%token DO
%token READ
%token WRITE
%token RETURN

%token IDENTIFIER
%token CONSTINT
%token CONSTCHAR

%token Number
%token Boolean
%token String
%token List
%token Dict

%left '+' '-'
%left '*' '/' '%'
%left or
%left and
%left not

%token ASSIGN
%token LE
%token DIFF
%token GE

%%
program: stmtlist
                    ;
stmtlist: stmt
                    | stmt stmtlist
                    ;
```

```
stmt: simplstmt ';'
                    | structstmt
                    ;
decllist: type declarationlist ';'
                    ;
declarationlist: declaration
                    | declaration ',' declarationlist
                    ;
declaration: IDENTIFIER
                    | assignstmt
                    ;
assignstmt: elem ASSIGN expression
                    ;
elem: IDENTIFIER
                    | arrayelem
                    ;
arrayelem: IDENTIFIER '[' IDENTIFIER ']'
                    | IDENTIFIER '[' CONSTINT ']'
                    ;
type: type1
                    | arraytype
                    ;
type1: Boolean
                    | Number
                    | String
                    ;
arraytype: List '(' type1 ')'
                    | Dict '(' type1 ')'
                    ;
simplstmt: assignstmt
                    | iostmt
                    | returnstmt
                    ;
iostmt: READ '(' elem ')'
                    | WRITE '(' elem ')'
                    | WRITE '(' CONSTCHAR ')'
                    ;
returnstmt: RETURN
                    | RETURN elem
                    | RETURN CONSTINT
                    ;
structstmt: decllist
                    | ifstmt
                    | whilestmt
                    ;
```

```
ifstmt: IF condition THEN stmtlist endifstmt
                    ;
endifstmt: DONE
                    | ELSE stmtlist DONE
                    | ELSE IF condition DO stmtlist endifstmt
                    ;
whilestmt: WHILE condition DO stmtlist DONE
                    | WHILE IDENTIFIER IN IDENTIFIER DO stmtlist DONE
                    | WHILE type IDENTIFIER IN IDENTIFIER DO stmtlist DONE
                    ;
condition: condition1
                    | not condition1
                    | condition1 logicalop condition1
                    ;
condition1: expression RELATION expression
                    | expression
                    ;
expression: factor arithmeticop factor
                    | factor
                    ;
factor: '(' condition ')'
                    | IDENTIFIER
                    | IDENTIFIER '[' IDENTIFIER ']'
                    | IDENTIFIER '[' CONSTINT ']'
                    | CONSTINT
                    | '[' factorList ']'
                    ;
factorList:  factor
                    | factor ',' factorList
                    ;
RELATION: '<'
                    | LE
                    | '='
                    | DIFF
                    | GE
                    | '>'
                    ;
arithmeticop: '+'
                    | '-'
                    | '*'
                    | '/'
                    | '%'
                    ;
logicalop: or
                    | and
```

```
                    ;

%%

yyerror(char *s)
{
  printf("%s\n", s);
}

extern FILE *yyin;

main(int argc, char **argv)
{
  if(argc>1) yyin = fopen(argv[1], "r");
  if((argc>2)&&(!strcmp(argv[2],"-d"))) yydebug = 1;
  if(!yyparse()) fprintf(stderr,"\tO.K.\n");
}
```

Examples:

- Correct:

p1.txt

```
Number n1, n2, n3;

n1 ← 123;
n2 ← 12;
n3 ← 23;

Number maxNr ← n3;

if n1 > maxNr then
    maxNr ← n1;
done

if n2 > maxNr then
    maxNr ← n2;
done

return maxNr;
```

```
D:\Info\Faculta\An_3_Sem_1\FLCD\Lab\Lab13>a.exe p1.txt
        O.K.
```

- Incorrect:

p1.txt

```
1   Number n1, n2, n3;
2
3   n1 ← 123;
4   n2 ← 12;
5   n3 ← 23;
6
7   Number maxNr ← n3;
8
9   if n1 > maxNr then
10      maxNr ← n1;
11                          // done is missing
12
13  if n2 > maxNr then
14      maxNr ← n2;
15  done
16
17  return maxNr;
18
```

```
D:\Info\Faculta\An_3_Sem_1\FLCD\Lab\Lab13>a.exe p1.txt
syntax error
```