Lab 9

%{

#include <stdio.h>

#include <stdlib.h>

#define YYDEBUG 1

%}

%token AND

%token OR

%token ANDAND

%token OROR

%token NOT

%token BREAK

%token CONTINUE

%token PASS

%token DO

%token IF

%token ELSE

%token WHILE

%token RETURN

%token START

%token PRINT

%token READ

%token PLUS

%token MINUS

%token MULTIPLICATION

%token MOD

%token EQUAL

%token LESS

%token GREATER

%token LESS\_OR\_EQUAL

%token GREATER\_OR\_EQUAL

%token NOT\_EQUAL

%token INCREMENT

%token DECREMENT

%token LEFT\_CURLY\_BRACKETS

%token RIGHT\_CURLY\_BRACKETS

%token LEFT\_ROUND\_PARENTHESIS

%token RIGHT\_ROUND\_PARENTHESIS

%token LEFT\_SQUARE\_PARENTHESIS

%token RIGHT\_SQUARE\_PARENTHESIS

%token SEMICOLON

%token COLON

%token COMMA

%token INTEGER

%token STRING

%token CHARACTER

%token FLOAT

%token IDENTIFIER

%start program

%%

program : START LEFT\_ROUND\_PARENTHESIS RIGHT\_ROUND\_PARENTHESIS COLON INTEGER compoundStatement ;

compoundStatement : LEFT\_CURLY\_BRACKETS statement RIGHT\_CURLY\_BRACKETS | LEFT\_CURLY\_BRACKETS statement statements RIGHT\_CURLY\_BRACKETS ;

    statement : declarationStatement | assignmentStatement | ifStatement | whileStatement | ioStatement | compoundStatement | returnStatement;

    statements : statement | statement statements ;

declarationStatement : type IDENTIFIER SEMICOLON | type LEFT\_SQUARE\_PARENTHESIS RIGHT\_SQUARE\_PARENTHESIS identifierList SEMICOLON | type assignmentStatement;

    identifierList: IDENTIFIER LEFT\_SQUARE\_PARENTHESIS INTEGER RIGHT\_SQUARE\_PARENTHESIS ;

    listIndex : IDENTIFIER LEFT\_SQUARE\_PARENTHESIS INTEGER RIGHT\_SQUARE\_PARENTHESIS ;

assignmentStatement : IDENTIFIER EQUAL expression SEMICOLON | listIndex EQUAL expression SEMICOLON ;

    expression : INTEGER | FLOAT | STRING | IDENTIFIER | term operator term ;

    term: INTEGER | FLOAT | STRING | IDENTIFIER | listIndex ;

    operator : PLUS | MINUS | MOD | MULTIPLICATION | EQUAL | LESS | GREATER | LESS\_OR\_EQUAL | GREATER\_OR\_EQUAL | NOT\_EQUAL | INCREMENT | DECREMENT ;

ifStatement : IF condition compoundStatement | IF condition compoundStatement ELSE compoundStatement | IF condition compoundStatement ELSE ifStatement;

    condition : LEFT\_ROUND\_PARENTHESIS evaluation RIGHT\_ROUND\_PARENTHESIS | LEFT\_ROUND\_PARENTHESIS evaluation continuation RIGHT\_ROUND\_PARENTHESIS;

    continuation: ANDAND evaluation | OROR evaluation;

    evaluation: expression relation expression;

    relation : GREATER | LESS | GREATER\_OR\_EQUAL | LESS\_OR\_EQUAL | EQUAL | NOT\_EQUAL ;

whileStatement : WHILE condition compoundStatement ;

ioStatement : READ LEFT\_ROUND\_PARENTHESIS IDENTIFIER RIGHT\_ROUND\_PARENTHESIS SEMICOLON | PRINT LEFT\_ROUND\_PARENTHESIS IDENTIFIER RIGHT\_ROUND\_PARENTHESIS SEMICOLON

    | PRINT LEFT\_ROUND\_PARENTHESIS STRING RIGHT\_ROUND\_PARENTHESIS SEMICOLON;

returnStatement: RETURN IDENTIFIER SEMICOLON | RETURN INTEGER SEMICOLON;

type: INTEGER | FLOAT | STRING

%%

yyerror(char \*s)

{

  printf("%s\n", s);

}

extern FILE \*yyin;

int 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");

}

# Run 1:

start(): Integer{

Integer number\_1 = 5;

Integer number\_2 = 20;

Integer number\_3 = 30;

if (number\_1 > number\_2 && number\_1 > number\_3){

print(number\_1);

}

else if ( number\_2 > number\_1 && number\_2 > number\_3 ){

print(number\_2);

}

else if ( number\_3 >= number\_1 && number\_3 <= number\_2 ){

print(number\_3);

}

else{

print("Values are not unique");

}

return 0;

}

Reserved: start

(

)

:

Reserved: Integer

{

Reserved: Integer

Identifier: number\_1

Reserved: =

Integer: 5

;

Reserved: Integer

Identifier: number\_2

Reserved: =

Integer: 20

;

Reserved: Integer

Identifier: number\_3

Reserved: =

Integer: 30

;

Reserved: if

(

Identifier: number\_1

Reserved: >

Identifier: number\_2

Reserved: &&

Identifier: number\_1

Reserved: >

Identifier: number\_3

)

{

Reserved: print

(

Identifier: number\_1

)

;

}

Reserved: else

Reserved: if

(

Identifier: number\_2

Reserved: >

Identifier: number\_1

Reserved: &&

Identifier: number\_2

Reserved: >

Identifier: number\_3

)

{

Reserved: print

(

Identifier: number\_2

)

;

}

Reserved: else

Reserved: if

(

Identifier: number\_3

Reserved: >=

Identifier: number\_1

Reserved: &&

Identifier: number\_3

Reserved: <=

Identifier: number\_2

)

{

Reserved: print

(

Identifier: number\_3

)

;

}

Reserved: else

{

Reserved: print

(

String: "Values are not unique"

)

;

}

Reserved: return

Integer: 0

;

}

O.K.

# Run 2:

start(): Integer{

Integer[] my@array = [1, 2, 3, 4, 5];

Integer array\_length = 5, i = 0;

Integer sum = 0;

Integer b;

for(i=0; i < array\_length; i++ ){

sum = sum + my\_array[ĂĂĂĂĂĂĂ];

}

print(sum);

}

Reserved: start

(

)

:

Reserved: Integer

{

Reserved: Integer

[

[

syntax error