Mihalcea Nicolae-Leonard 935/1

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 comp
oundStatement ;

compoundStatement : LEFT_CURLY_BRACKETS statement RIGHT_CURLY_BRACKETS | LEFT_CUR
LY_BRACKETS statement statements RIGHT_CURLY_BRACKETS ;
    statement : declarationStatement | assignmentStatement | ifStatement | whileS
tatement | ioStatement | compoundStatement | returnStatement;
    statements : statement | statement statements ;

declarationStatement : type IDENTIFIER SEMICOLON | type LEFT_SQUARE_PARENTHESIS R
IGHT_SQUARE_PARENTHESIS identifierList SEMICOLON | type assignmentStatement;
    identifierList: IDENTIFIER LEFT_SQUARE_PARENTHESIS INTEGER RIGHT_SQUARE_PAREN
THESIS ;
    listIndex : IDENTIFIER LEFT_SQUARE_PARENTHESIS INTEGER RIGHT_SQUARE_PARENTHES
IS ;

assignmentStatement : IDENTIFIER EQUAL expression SEMICOLON | listIndex EQUAL exp
ression SEMICOLON ;
    expression : INTEGER | FLOAT | STRING | IDENTIFIER | term operator term ;
    term: INTEGER | FLOAT | STRING | IDENTIFIER | listIndex ;
    operator : PLUS | MINUS | MOD | MULTIPLICATION | EQUAL | LESS | GREATER | LES
S_OR_EQUAL | GREATER_OR_EQUAL | NOT_EQUAL | INCREMENT | DECREMENT ;

ifStatement : IF condition compoundStatement | IF condition compoundStatement ELS
E 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_EQ
UAL ;

whileStatement : WHILE condition compoundStatement ;

ioStatement : READ LEFT_ROUND_PARENTHESIS IDENTIFIER RIGHT_ROUND_PARENTHESIS SEMI
COLON | 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
```