Dumbravean Bogdan, 933

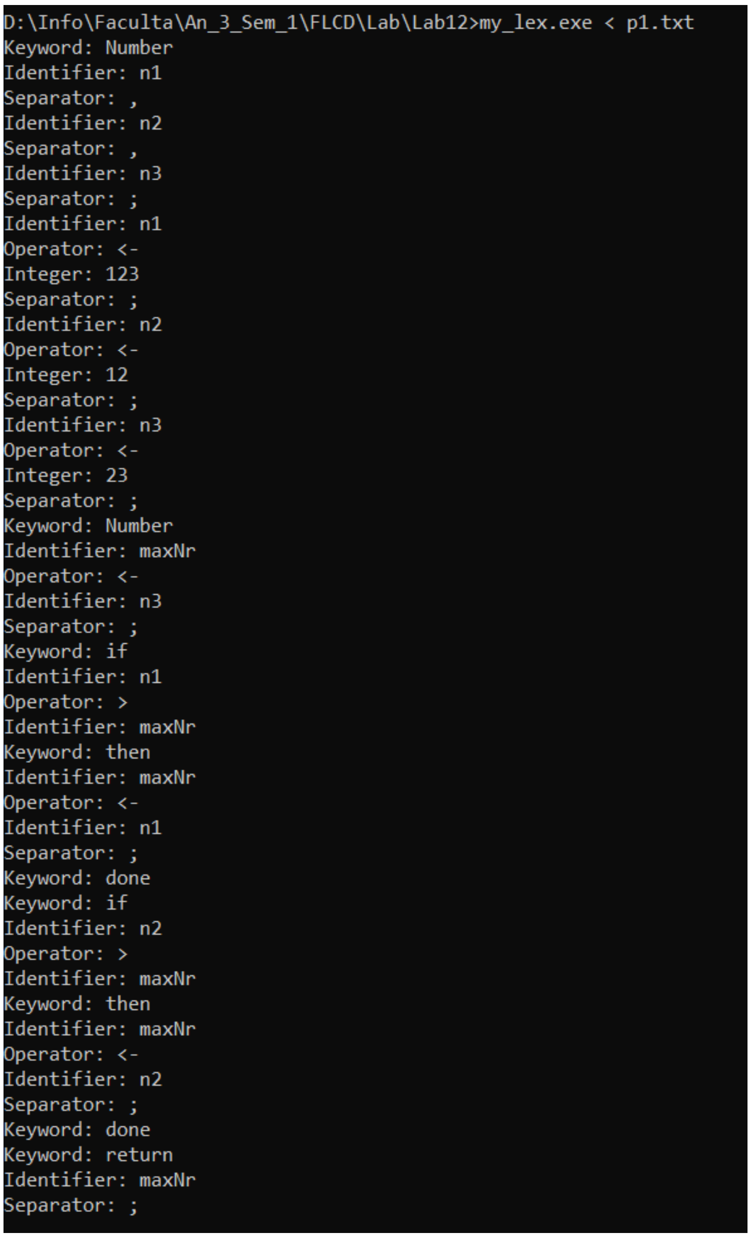
Lang.lxi:

```
%{                                        /* need this for the call to atof() below */
#include <math.h>
%}

%option noyywrap

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


%%


(\+0)|(\-0)                                       printf("! Lexical error: %s\n", yytext);
{CONSTINT}{ID}                                    printf("! Lexical error: %s\n", yytext);
0{CONSTINT}                                       printf("! Lexical error: %s\n", yytext);

{CONSTINT}                                        printf( "Integer: %s\n", yytext);
{CONSTCHAR}                                       printf( "String: %s\n", yytext);
Number|Boolean|String|List|Dict|const|if|then|else|done|while|in|do|read|write|return|and|not|or  {
   printf( "Keyword: %s\n", yytext );
}
{ID}                                              printf( "Identifier: %s\n", yytext );
"+"|"-"|"*"|"/"|"%"|"<-"|"<"|"<="|"="|">="|">"|"<>"        printf( "Operator: %s\n", yytext );
"["|"]"|"("|")"|","|";"                           printf( "Separator: %s\n", yytext );

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


%%
int main( argc, argv )
int argc;
char **argv;
{
   ++argv, --argc; /* skip over program name */
   if ( argc > 0 )
      yyin = fopen( argv[0], "r" );
   else
      yyin = stdin;
   yylex();
}
```

P1:

```
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;
```

Result:

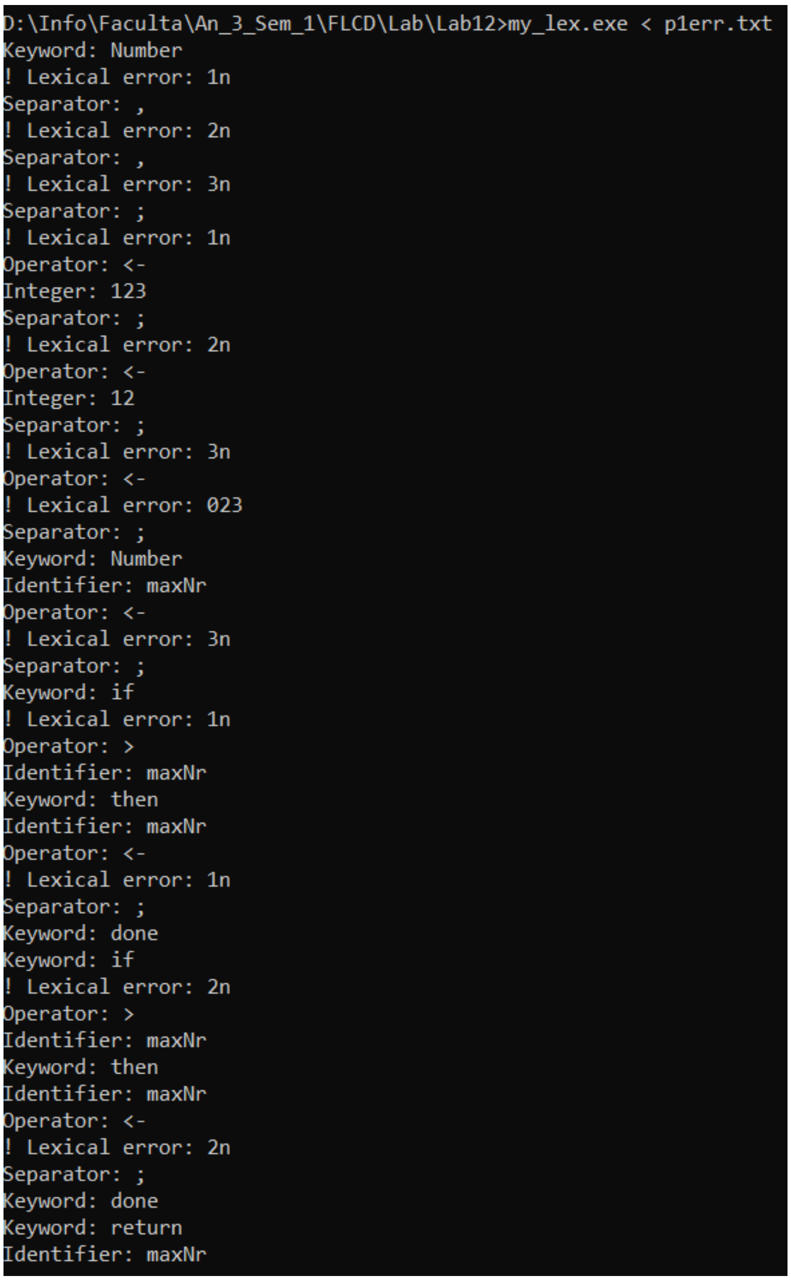


```
D:\Info\Faculta\An_3_Sem_1\FLCD\Lab\Lab12>my_lex.exe < p1.txt
Keyword: Number
Identifier: n1
Separator: ,
Identifier: n2
Separator: ,
Identifier: n3
Separator: ;
Identifier: n1
Operator: <-
Integer: 123
Separator: ;
Identifier: n2
Operator: <-
Integer: 12
Separator: ;
Identifier: n3
Operator: <-
Integer: 23
Separator: ;
Keyword: Number
Identifier: maxNr
Operator: <-
Identifier: n3
Separator: ;
Keyword: if
Identifier: n1
Operator: >
Identifier: maxNr
Keyword: then
Identifier: maxNr
Operator: <-
Identifier: n1
Separator: ;
Keyword: done
Keyword: if
Identifier: n2
Operator: >
Identifier: maxNr
Keyword: then
Identifier: maxNr
Operator: <-
Identifier: n2
Separator: ;
Keyword: done
Keyword: return
Identifier: maxNr
Separator: ;
```

P1err:

```
Number 1n, 2n, 3n;                          // Lexical error -> a variable shouldn't start with letter

1n <- 123;
2n <- 12;
3n <- 023;                                  // Lexical error -> number shouldn't start with 0

Number maxNr <- 3n;

if 1n > maxNr then
        maxNr <- 1n;
done

if 2n > maxNr then
        maxNr <- 2n;
done

return maxNr
```

Result:



```
D:\Info\Faculta\An_3_Sem_1\FLCD\Lab\Lab12>my_lex.exe < p1err.txt
Keyword: Number
! Lexical error: 1n
Separator: ,
! Lexical error: 2n
Separator: ,
! Lexical error: 3n
Separator: ;
! Lexical error: 1n
Operator: <-
Integer: 123
Separator: ;
! Lexical error: 2n
Operator: <-
Integer: 12
Separator: ;
! Lexical error: 3n
Operator: <-
! Lexical error: 023
Separator: ;
Keyword: Number
Identifier: maxNr
Operator: <-
! Lexical error: 3n
Separator: ;
Keyword: if
! Lexical error: 1n
Operator: >
Identifier: maxNr
Keyword: then
Identifier: maxNr
Operator: <-
! Lexical error: 1n
Separator: ;
Keyword: done
Keyword: if
! Lexical error: 2n
Operator: >
Identifier: maxNr
Keyword: then
Identifier: maxNr
Operator: <-
! Lexical error: 2n
Separator: ;
Keyword: done
Keyword: return
Identifier: maxNr
```