LAB 8

LINK TO GIT: https://github.com/radutalaviniaelena/FLCD

Statement: Use lex

You may use any version (LEX or FLEX)

1) Write a LEX specification containing the regular expressions corresponding to your language specification - see lab 1

2) Use Lex in order to obtain a scanner. Test for the same input as in lab 1 (p1, p2).

Deliverables: pdf file containing lang.lxi (lex specification file) + demo

Content of scanner.lxi file:

```
%{
int no_of_lines = 0;
%}

%option noyywrap

DIGIT [0-9]
NZ_DIGIT [1-9]
LETTER [a-zA-Z]
INTEGER_CONSTANT [+-]?{NZ_DIGIT}{DIGIT}*|0
STRING_CONSTANT \"({LETTER}|{DIGIT})*\"
CHAR_CONSTANT \'({DIGIT}|{LETTER})\'
IDENTIFIER "_"{LETTER}({LETTER}|{DIGIT})*
CONSTANT {INTEGER_CONSTANT}|{STRING_CONSTANT}|{CHAR_CONSTANT}
```

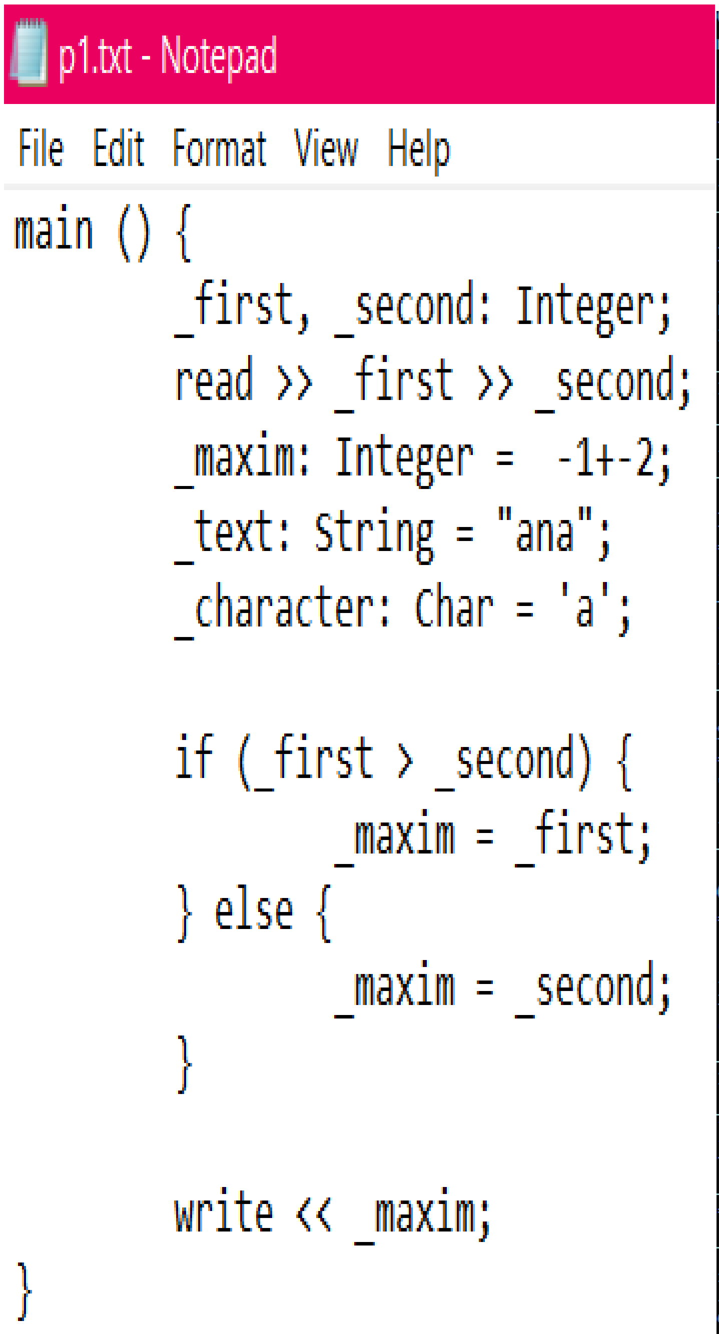
```
%%

"read"|"write"|"if"|"else"|"while"|"for"|"in"|"range"|"Integer"|"String"|"Char"|"main" printf("%s -
reserved word\n", yytext);

{IDENTIFIER} printf("%s - identifier\n", yytext);

{CONSTANT} printf("%s - constant\n", yytext);

"+"|"-"|"*"|"/"|"%"|"="|">"|">="|"<"|"<="|"=="|"!=" printf("%s - operator\n", yytext);

">>"|"<<"|";"|":" printf("%s - separator\n", yytext);
\( printf("%s - separator\n", yytext);
\) printf("%s - separator\n", yytext);
\[ printf("%s - separator\n", yytext);
\] printf("%s - separator\n", yytext);
\{ printf("%s - separator\n", yytext);
\} printf("%s - separator\n", yytext);
\" printf("%s - separator\n", yytext);
\' printf("%s - separator\n", yytext);
\, printf("%s - separator\n", yytext);

[ \t]+ {} /* elimina spatii */

\n ++no_of_lines;

[+-]0 {printf("Illegal integer constant at line %d: a number cannot start with 0.\n", no_of_lines); return
0;}
```

```
0{DIGIT}* {printf("Illegal integer constant at line %d: a number cannot start with 0.\n", no_of_lines);
return 0;}

\'[^({DIGIT}|{LETTER})]\' {printf("Illegal char constant at line %d: a character should be a digit or a
letter.\n", no_of_lines); return 0;}

\'({DIGIT}|{LETTER}) {printf("Illegal char constant at line %d: unclosed quotes.\n", no_of_lines); return
0;}

\"(({LETTER}|{DIGIT})*[^({LETTER}|{DIGIT})]({LETTER}|{DIGIT})*)*\" {printf("Illegal string constant at line
%d: a string should contain only digits and letters.\n", no_of_lines); return 0;}

\"({LETTER}|{DIGIT})* {printf("Illegal string constant at line %d: unclosed quotes.\n", no_of_lines); return
0;}

. {printf("Illegal token at line %d.\n", no_of_lines); return 0;}

%%

void main(argc, argv)
int argc;
char** argv;
{
	if (argc > 1)
	{
		FILE *file;
		file = fopen(argv[1], "r");
		if (!file)
		{
		fprintf(stderr, "Could not open %s\n", argv[1]);
```

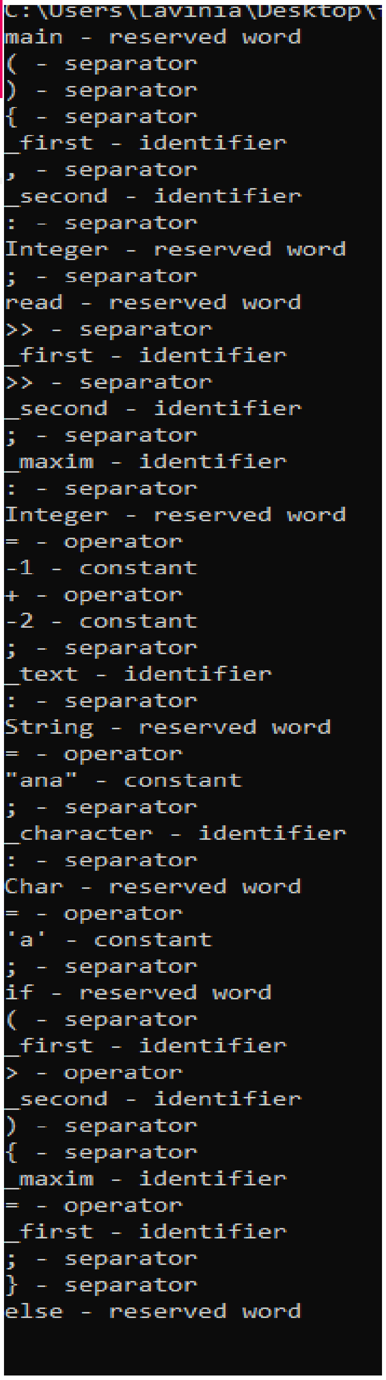
```
                exit(1);
                }
                yyin = file;
        }

        yylex();
}
```

<u>EXAMPLE</u>:

p1.txt

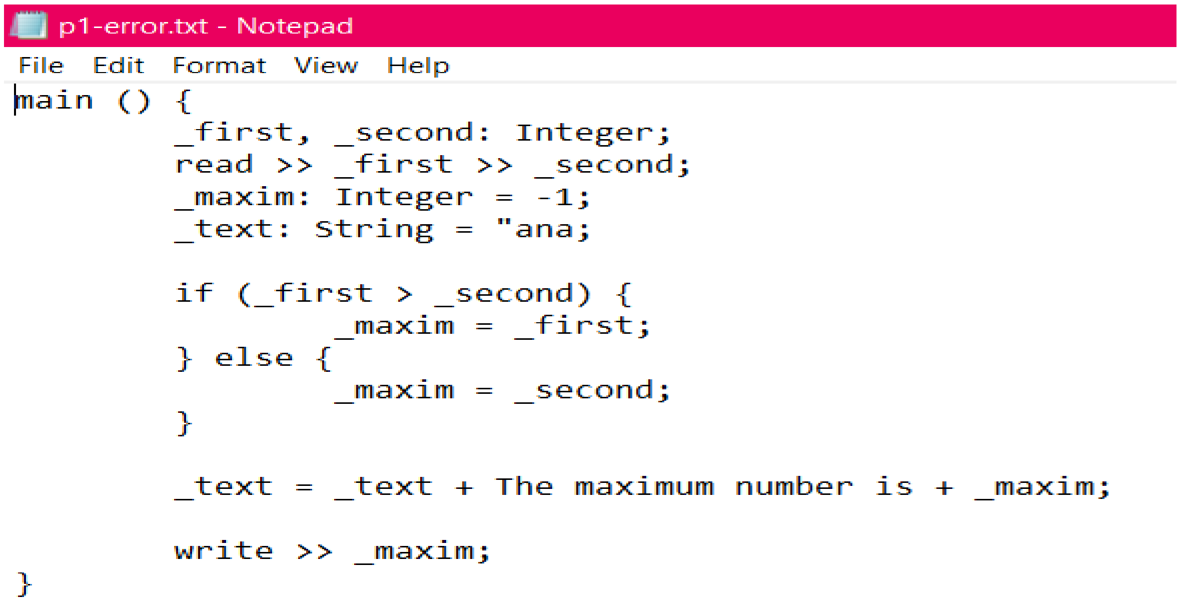


```
main () {
        _first, _second: Integer;
        read >> _first >> _second;
        _maxim: Integer =  -1+-2;
        _text: String = "ana";
        _character: Char = 'a';

        if (_first > _second) {
                _maxim = _first;
        } else {
                _maxim = _second;
        }

        write << _maxim;
}
```



```
C:\Users\Lavinia\Desktop\
main - reserved word
( - separator
) - separator
{ - separator
_first - identifier
, - separator
_second - identifier
: - separator
Integer - reserved word
; - separator
read - reserved word
>> - separator
_first - identifier
>> - separator
_second - identifier
; - separator
_maxim - identifier
: - separator
Integer - reserved word
= - operator
-1 - constant
+ - operator
-2 - constant
; - separator
_text - identifier
: - separator
String - reserved word
= - operator
"ana" - constant
; - separator
_character - identifier
: - separator
Char - reserved word
= - operator
'a' - constant
; - separator
if - reserved word
( - separator
_first - identifier
> - operator
_second - identifier
) - separator
{ - separator
_maxim - identifier
= - operator
_first - identifier
; - separator
} - separator
else - reserved word
```

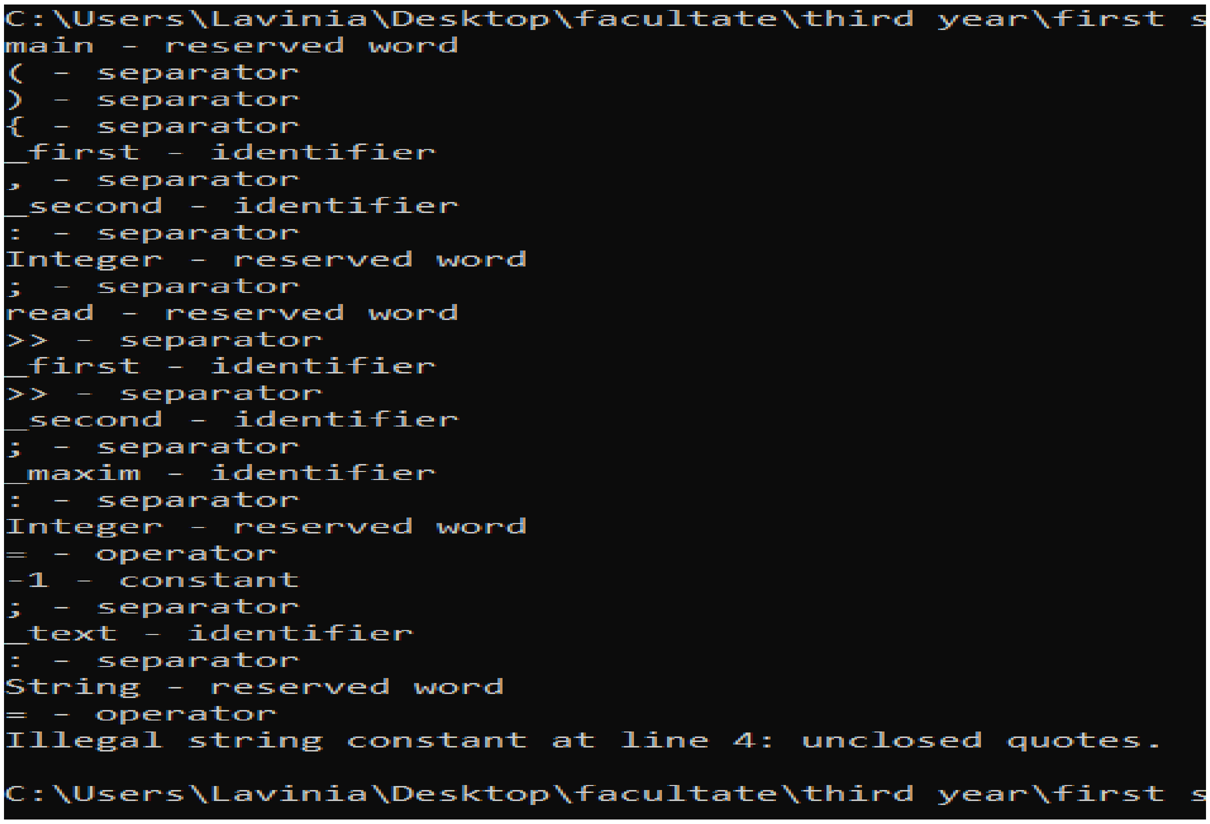
p1-eror.txt



p1-error.txt - Notepad

File Edit Format View Help

```
main () {
        _first, _second: Integer;
        read >> _first >> _second;
        _maxim: Integer = -1;
        _text: String = "ana;

        if (_first > _second) {
                _maxim = _first;
        } else {
                _maxim = _second;
        }

        _text = _text + The maximum number is + _maxim;

        write >> _maxim;
}
```



```
C:\Users\Lavinia\Desktop\facultate\third year\first s
main - reserved word
( - separator
) - separator
{ - separator
_first - identifier
, - separator
_second - identifier
: - separator
Integer - reserved word
; - separator
read - reserved word
>> - separator
_first - identifier
>> - separator
_second - identifier
; - separator
_maxim - identifier
: - separator
Integer - reserved word
= - operator
-1 - constant
; - separator
_text - identifier
: - separator
String - reserved word
= - operator
Illegal string constant at line 4: unclosed quotes.

C:\Users\Lavinia\Desktop\facultate\third year\first s
```