Documentation LEX

https://github.com/iuliaaai/LFTC

specif.lxi:

```
%{
   #include <stdio.h>
   #include <string.h>
   int lines = 0;
%}

%option noyywrap
%option caseless

DIGIT               [0-9]
NON_ZERO_DIGIT [1-9]
INT_CONSTANT [-]?{NON_ZERO_DIGIT}{DIGIT}*|0
LETTER [a-zA-Z]
STRING_CONSTANT ["|'](LETTER}|{DIGIT}|" ")*["|']
IDENTIFIER {LETTER}({LETTER}|{DIGIT})*

%%

func {printf("Reserved word: %s\n", yytext);}
Int    {printf( "Reserved word: %s\n", yytext);}
String    {printf( "Reserved word: %s\n", yytext);}
Bool    {printf( "Reserved word: %s\n", yytext);}
Char    {printf( "Reserved word: %s\n", yytext);}
if    {printf( "Reserved word: %s\n", yytext);}
elif    {printf( "Reserved word: %s\n", yytext);}
else    {printf( "Reserved word: %s\n", yytext);}
let    {printf( "Reserved word: %s\n", yytext);}
var    {printf( "Reserved word: %s\n", yytext);}
ret    {printf( "Reserved word: %s\n", yytext);}
True    {printf( "Reserved word: %s\n", yytext);}
False    {printf( "Reserved word: %s\n", yytext);}
read    {printf( "Reserved word: %s\n", yytext);}
print    {printf( "Reserved word: %s\n", yytext);}
loop    {printf( "Reserved word: %s\n", yytext);}
GO    {printf( "Reserved word: %s\n", yytext);}
STOP  {printf( "Reserved word: %s\n", yytext);}

{IDENTIFIER}    {printf( "Identifier: %s\n", yytext );}
```

```
{INT_CONSTANT}      {printf( "Constant: %s\n", yytext );}
{STRING_CONSTANT}    {printf( "Constant: %s\n", yytext );}

"@"    {printf( "Separator: %s\n", yytext );}
"#"    {printf( "Separator: %s\n", yytext );}
";"    {printf( "Separator: %s\n", yytext );}
"'"    {printf( "Separator: %s\n", yytext );}
"{"    {printf( "Separator: %s\n", yytext );}
"}"    {printf( "Separator: %s\n", yytext );}
"("    {printf( "Separator: %s\n", yytext );}
")"    {printf( "Separator: %s\n", yytext );}
"["    {printf( "Separator: %s\n", yytext );}
"]"    {printf( "Separator: %s\n", yytext );}
"+"    {printf( "Operator: %s\n", yytext );}
"-"    {printf( "Operator: %s\n", yytext );}
"*"    {printf( "Operator: %s\n", yytext );}
"/"    {printf( "Operator: %s\n", yytext );}
"<"    {printf( "Operator: %s\n", yytext );}
">"    {printf( "Operator: %s\n", yytext );}
"<="    {printf( "Operator: %s\n", yytext );}
">="    {printf( "Operator: %s\n", yytext );}
"!="    {printf( "Operator: %s\n", yytext );}
"=="    {printf( "Operator: %s\n", yytext );}
"="    {printf( "Separator: %s\n", yytext );}
"!"    {printf( "Operator: %s\n", yytext );}
"?:"    {printf( "Operator: %s\n", yytext );}
"%"    {printf( "Operator: %s\n", yytext );}

[ \t]+      {}
[\n]+ {lines++;}

%%

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

p1.txt:

```
GO
var n@Int;
read(n);
var isPrime@Bool = True;
if n < 2{
        isPrime = False;
}
elif n == 2{
        isPrime = True;
}
elif n % 2 == 0{
        isPrime = False;
}

var d@Int = 3;

loop d <= n/2 {
        if n % d == 0{
        isPrime = False;
        }
        d = d + 3;
}

isPrime = True;
STOP
```

Output:

```
Reserved word: GO
Reserved word: var
Identifier: n
Separator: @
Reserved word: Int
Separator: ;
Reserved word: read
Separator: (
Identifier: n
Separator: )
Separator: ;
Reserved word: var
Identifier: isPrime
Separator: @
Reserved word: Bool
Separator: =
```

Reserved word: True
Separator: ;
Reserved word: if
Identifier: n
Operator: <
Constant: 2
Separator: {
Identifier: isPrime
Separator: =
Reserved word: False
Separator: ;
Separator: }
Reserved word: elif
Identifier: n
Operator: ==
Constant: 2
Separator: {
Identifier: isPrime
Separator: =
Reserved word: True
Separator: ;
Separator: }
Reserved word: elif
Identifier: n
Operator: %
Constant: 2
Operator: ==
Constant: 0
Separator: {
Identifier: isPrime
Separator: =
Reserved word: False
Separator: ;
Separator: }
Reserved word: var
Identifier: d
Separator: @
Reserved word: Int
Separator: =
Constant: 3
Separator: ;
Reserved word: loop
Identifier: d
Operator: <=

Identifier: n
Operator: /
Constant: 2
Separator: {
Reserved word: if
Identifier: n
Operator: %
Identifier: d
Operator: ==
Constant: 0
Separator: {
Identifier: isPrime
Separator: =
Reserved word: False
Separator: ;
Separator: }
Identifier: d
Separator: =
Identifier: d
Operator: +
Constant: 3
Separator: ;
Separator: }
Identifier: isPrime
Separator: =
Reserved word: True
Separator: ;
Reserved word: STOP

p2.txt:

```
GO
var n@Int;
read(n);
let lastDigit@Int = n % 10;
print(lastDigit);
print("hello there");
print('a');
STOP
```

Output:

Reserved word: GO
Reserved word: var

Identifier: n
Separator: @
Reserved word: Int
Separator: ;
Reserved word: read
Separator: (
Identifier: n
Separator:)
Separator: ;
Reserved word: let
Identifier: lastDigit
Separator: @
Reserved word: Int
Separator: =
Identifier: n
Operator: %
Constant: 10
Separator: ;
Reserved word: print
Separator: (
Identifier: lastDigit
Separator:)
Separator: ;
Reserved word: print
Separator: (
Constant: "hello there"
Separator:)
Separator: ;
Reserved word: print
Separator: (
Constant: 'a'
Separator:)
Separator: ;
Reserved word: STOP