Installation and test file hello.l

```
hello.l:
%{

#include "y.tab.h"
int yyerror(char *errormsg);

%}

%%

("hi"|"oi")"\n"       { return HI;  }
("tchau"|"bye")"\n"   { return BYE; }
.                     { yyerror("Unknown char");  }

%%

int main(void)
{
   yyparse();
   return 0;
}

int yywrap(void)
{
   return 0;
}

int yyerror(char *errormsg)
{
    fprintf(stderr, "%s\n", errormsg);
    exit(1);
}
```

Finished lab
Github: https://github.com/pukoarmin/Formal-Langauges-and-Compiler-Design.git
In order to run the file, execute from the Ubuntu terminal:

```
lex lab8.l
cc lex.yy.c -o rezultat -lfl
./rezultat SAU ./rezultat<<file.in
```

Lex code:

```
%{
#include <math.h>
#include <stdio.h>
int line = 0;
%}
```

```
%option noyywrap

LETTER          [A-Za-z_]
DIGIT           [0-9]
IDENTIFIER      [a-zA-Z][a-zA-Z0-9]{0,32}
DELIMIT         [;.,:]
OPERATOR  [-|+|*|/|%|=|!=|<|>|<=|>=]
INTEGER         [1-9][0-9]*|0
STRING          ["][^\n"]*["]
BOOLEAN         [^True$|^False$]

%%

"BEGIN"         {printf("Reserved word: %s\n", yytext); /*return BEGIN;*/}
"END"           {printf("Reserved word: %s\n", yytext); /*return END;*/}
"INTEGER" {printf("Reserved word: %s\n", yytext); /*return INTEGER;*/}
"STRING"  {printf("Reserved word: %s\n", yytext); /*return STRING;*/}
"BOOLEAN"       {printf("Reserved word: %s\n", yytext); /*return BOOLEAN;*/}
"List"          {printf("Reserved word: %s\n", yytext); /*return List;*/}
"append"  {printf("Reserved word: %s\n", yytext); /*return append;*/}
"remove"  {printf("Reserved word: %s\n", yytext); /*return remove;*/}
"length"  {printf("Reserved word: %s\n", yytext); /*return length;*/}
"Struct"  {printf("Reserved word: %s\n", yytext); /*return Struct;*/}
"True"          {printf("Reserved word: %s\n", yytext); /*return True;*/}
"False"   {printf("Reserved word: %s\n", yytext); /*return False;*/}
"read"          {printf("Reserved word: %s\n", yytext); /*return read;*/}
"print"   {printf("Reserved word: %s\n", yytext); /*return print;*/}
"if"            {printf("Reserved word: %s\n", yytext); /*return if;*/}
"else"          {printf("Reserved word: %s\n", yytext); /*return else;*/}
"for"           {printf("Reserved word: %s\n", yytext); /*return for;*/}

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

":"             {printf("Separator: %s\n", yytext); /*return COLON;*/}
";"             {printf("Separator: %s\n", yytext); /*return SEMI_COLON;*/}
","             {printf("Separator: %s\n", yytext); /*return COMA;*/}
"."             {printf("Separator: %s\n", yytext); /*return DOT;*/}
"+"             {printf("Operator: %s\n", yytext); /*return PLUS;*/}
"-"             {printf("Operator: %s\n", yytext); /*return MINUS;*/}
"*"             {printf("Operator: %s\n", yytext); /*return MULTIPLY;*/}
"/"             {printf("Operator: %s\n", yytext); /*return DIVISION;*/}
"("             {printf("Separator: %s\n", yytext); /*return LEFT_ROUND_PARANTHESIS;*/}
")"             {printf("Separator: %s\n", yytext); /*return RIGHT_ROUND_PARANTHESIS;*/}
"["             {printf("Separator: %s\n", yytext); /*return LEFT_SQUARE_PARANTHESIS;*/}
"]"             {printf("Separator: %s\n", yytext); /*return
RIGHT_SQUARE_PARANTHESIS;*/}
"{"             {printf("Separator: %s\n", yytext); /*return LEFT_CURLY_PARANTHESIS;*/}
"}"             {printf("Separator: %s\n", yytext); /*return RIGHT_CURLY_PARANTHESIS;*/}
">"             {printf("Operator: %s\n", yytext); /*return GREATER_THAN;*/}
```

```
">="      {printf("Operator: %s\n", yytext); /*return GREATER_OR_EQUAL_THAN;*/}
"<="      {printf("Operator: %s\n", yytext); /*return LESS_OR_EQUAL_THAN;*/}
"<"           {printf("Operator: %s\n", yytext); /*return LESS_THAN;*/}
"="           {printf("Operator: %s\n", yytext); /*return ASSIGNMENT;*/}
"=="      {printf("Operator: %s\n", yytext); /*return EQUAL;*/}
"!="      {printf("Operator: %s\n", yytext); /*return DIFFERENT;*/}
"!"           {printf("Operator: %s\n", yytext); /*return NEGATION;*/}

[ \t]+    /* elimina spatii */     {}
[\n]+     {++line;}

[a-zA-Z][a-zA-Z0-9]{8,}  {printf("Illegal size of the identifier at line %d\n", line);
return -1;}

[0-9][a-zA-Z0-9]{0,32}   {printf("Illegal identifier at line %d\n", line); return -1;}

.    {printf("Illegal symbol at line %d\n", line); return -1;}
%%
```