Documentation

Link to GitHub : https://github.com/diana-dr/Formal-Languages-and-Compiler-Design/tree/master/Lab%208

spec.lxi

```
%{
#include <stdio.h>
#include <string.h>

int currentLine = 1;
%}

%option noyywrap
%option caseless

DIGIT           [0-9]
NZ_DIGIT    [1-9]
ZERO        [0]
NUMBER      {NZ_DIGIT}{DIGIT}*
SIGN        [+]|[-]
INTEGER         {ZERO}|{NUMBER}|{SIGN}{NUMBER}
SIGNER_INTEGER {SIGN}{NUMBER}
SPECIAL_CHAR
"_"|"."|","|";"|":"|"?"|"!"|"@"|"/"|"("|")"|"-"|"+"|"="|"{"|"}"|"*"|"["|"]"|"$"|
"%"|"^"|" "
CHAR        {DIGIT}|{SPECIAL_CHAR}|[a-zA-Z]
CHARACTER       "'"{CHAR}"'"
STRING          [\"]{CHAR}*[\"]
CONSTANT                {STRING}|{INTEGER}|{CHARACTER}
IDENTIFIER              [a-zA-Z_][a-zA-Z0-9_]*

%%

and {printf("%s - reserved word\n", yytext);}
or {printf("%s - reserved word\n", yytext);}
not {printf("%s - reserved word\n", yytext);}
if {printf("%s - reserved word\n", yytext);}
do {printf("%s - reserved word\n", yytext);}
else {printf("%s - reserved word\n", yytext);}
elif {printf("%s - reserved word\n", yytext);}
while {printf("%s - reserved word\n", yytext);}
for {printf("%s - reserved word\n", yytext);}
read {printf("%s - reserved word\n", yytext);}
write {printf("%s - reserved word\n", yytext);}
int {printf("%s - reserved word\n", yytext);}
string {printf("%s - reserved word\n", yytext);}
char {printf("%s - reserved word\n", yytext);}
function {printf("%s - reserved word\n", yytext);}
bool {printf("%s - reserved word\n", yytext);}
return {printf("%s - reserved word\n", yytext);}


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

; {printf("%s - separator\n", yytext);}
\, {printf("%s - separator\n", yytext);}
\t {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 - operator\n", yytext);}
\- {printf("%s - operator\n", yytext);}
\* {printf("%s - operator\n", yytext);}
\/ {printf("%s - operator\n", yytext);}
\% {printf("%s - operator\n", yytext);}
\< {printf("%s - operator\n", yytext);}
\> {printf("%s - operator\n", yytext);}
\<= {printf("%s - operator\n", yytext);}
\>= {printf("%s - operator\n", yytext);}
"=" {printf("%s - operator\n", yytext);}
\== {printf("%s - operator\n", yytext);}
\!= {printf("%s - operator\n", yytext);}

[\n\r] {currentLine++;}
[ \t\n]+ {}

[a-zA-Z_0-9][a-zA-Z0-9_]* {printf("%s - illegal identifier found at line %d\n",
yytext, currentLine);}
\'[a-zA-Z0-9]*\' {printf("%s - illegal char at line %d\n", yytext,
currentLine);}

%%

int 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();
}
```

p1.in

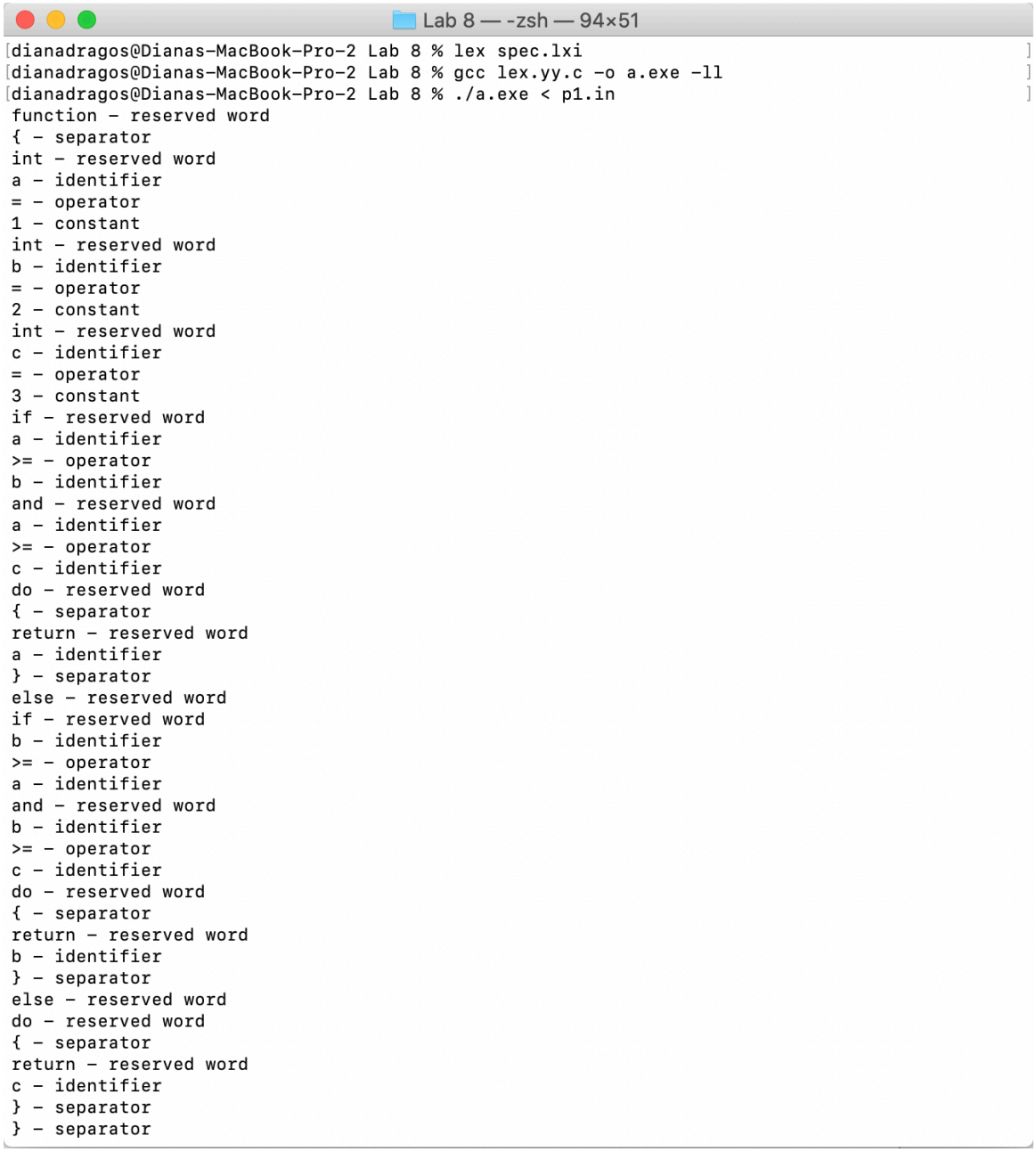
```
function
{
    int a = 1
    int b = 2
    int c = 3

    if a >= b and a >= c do {
        return a
    }

    else if b >= a and b >= c do {
        return b
    }

    else do {
        return c
    }
}
```

Example


Lab 8 — -zsh — 94×51
dianadragos@Dianas-MacBook-Pro-2 Lab 8 % lex spec.lxi
dianadragos@Dianas-MacBook-Pro-2 Lab 8 % gcc lex.yy.c -o a.exe -ll
dianadragos@Dianas-MacBook-Pro-2 Lab 8 % ./a.exe < p1.in
function - reserved word
{ - separator
int - reserved word
a - identifier
= - operator
1 - constant
int - reserved word
b - identifier
= - operator
2 - constant
int - reserved word
c - identifier
= - operator
3 - constant
if - reserved word
a - identifier
>= - operator
b - identifier
and - reserved word
a - identifier
>= - operator
c - identifier
do - reserved word
{ - separator
return - reserved word
a - identifier
} - separator
else - reserved word
if - reserved word
b - identifier
>= - operator
a - identifier
and - reserved word
b - identifier
>= - operator
c - identifier
do - reserved word
{ - separator
return - reserved word
b - identifier
} - separator
else - reserved word
do - reserved word
{ - separator
return - reserved word
c - identifier
} - separator
} - separator