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

%{

#include <stdio.h>

int line = 0;

%}

%option noyywrap

NUMBER [1-9][0-9]\*

OPERATOR [-+\*/^%<>=!]|">="|"<="|"!="|"=="

IO "<<"|">>"

SEPARATOR \[|\]|\{|\}|\(|\)|;

ID [a-zA-Z][a-zA-Z0-9\_]{0,255}

CHARACTER [a-zA-Z0-9\_]

RESERVED "if"|"while"|"bool"|"char"|"int"|"string"|"float"

%%

[0-9]+{ID}|"+0"|"-0"|[+-]?"0"[0-9]+ {printf("Error: %s on line: %d\n", yytext, line);}

{IO} printf("IO: %s\n", yytext);

{OPERATOR} printf("Operator: %s\n", yytext);

{SEPARATOR} printf("Separator: %s\n", yytext);

[+-]?{NUMBER}|"0" printf("Int: %s\n", yytext);

[+-]?({NUMBER}|"0")"."[0-9]+ printf("Float: %s\n", yytext);

"'"{CHARACTER}"'" printf("Char: %s\n", yytext);

\"({CHARACTER}|[ ])\*\" printf("String: %s\n", yytext);

"true"|"false" printf("Bool: %s\n", yytext);

{RESERVED} printf("Reserved: %s\n", yytext);

{ID} printf("ID: %s\n", yytext);

[ \t]+ /\* eat up whitespace \*/

[\n] {line++;}

. {printf("Error: %s on line: %d\n", yytext, line);}

%%

void main(argc, argv)

int argc;

char \*\*argv;

{

++argv, --argc;

if(argc>0)

yyin = fopen(argv[0], "r");

else

yyin = stdin;

yylex();

}

int a=9;

int b=6;

if(a>b){

>>"a is the maximum";

}else{

>>"b is the maximum";

}

Reserved: int

ID: a

Operator: =

Int: 9

Separator: ;

Reserved: int

ID: b

Operator: =

Int: 6

Separator: ;

Reserved: if

Separator: (

ID: a

Operator: >

ID: b

Separator: )

Separator: {

IO: >>

String: "a is the maximum"

Separator: ;

Separator: }

ID: else

Separator: {

IO: >>

String: "b is the maximum"

Separator: ;

Separator: }

+091

Error: +091 on line: 0

"asdass asd

Error: " on line: 1

ID: asdass

ID: asd

'aa'

Error: ' on line: 2

ID: aa

Error: ' on line: 2