Practical-5

DEFINATION: Implementation of a Lexical Analyzer for C Language Compiler

OBJECTIVE: To design and implement a lexical analyser to perform 1st, 2nd, 3rd, and 5th task as per the list given in practical 2.

CODE:

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

%option noyywrap

digit [0-9]+
letter [a-zA-Z_][a-zA-Z0-9_]*
ws [ \t\n]+

%%

int { printf("Keyword: %s\n", yytext); }
char { printf("Keyword: %s\n", yytext); }
return { printf("Keyword: %s\n", yytext); }

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

```
"+"|"-"|"*"|"/"|"%" { printf("Operator: %s\n", yytext); }
"=" { printf("Operator: %s\n", yytext); }
";"|"(" | ")"|"{"|"}" { printf("Punctuation: %s\n", yytext); }

{ws} {/* Ignore white spaces */}

. { printf("Invalid Token: %s\n", yytext); }

%%

int main() {
    FILE *f;
    f = fopen("test.c", "r");
    if (!f) {
        printf("Cannot open file\n");
        return 1;
    }
    yyin = f;
    yylex();
    fclose(f);
    return 0;
}
```

OUTPUT:

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PREM\Desktop\SEM-6\DLP LAB\PRACTICAL-5>flex pr5.l

C:\Users\PREM\Desktop\SEM-6\DLP LAB\PRACTICAL-5>gcc lex.yy.c -o pr5.exe

C:\Users\PREM\Desktop\SEM-6\DLP LAB\PRACTICAL-5>pr5.exe<test.c
Operator: /
Operator: /
Operator: /
Operator: /
Identifier: function
Identifier: prototype
Operator: /
Operator: /
Identifier: void
Identifier: add
Keyword: int
Invalid Token: ,
Keyword: int
Invalid Token: )
Operator: /
Operator: /
Identifier: void
Identifier: main
Invalid Token: )
Operator: /
Operator: /
Invalid Token: {
Operator: /
Operator: /
Keyword: int
Identifier: a
Invalid Token: ,
Identifier: b
Operator: /
Operator: /
Identifier: a
Operator: =
Constant: 10
Operator: /
Operator: /
Identifier: b
Operator: =
Constant: 20
Operator: /
Operator: /
Operator: /
Operator: /
Identifier: function
Identifier: call
Operator: /
Operator: /
Identifier: add
Identifier: a
Invalid Token: ,
Identifier: b
Invalid Token: )
Operator: /
Operator: /
Invalid Token: }
Operator: /
Operator: /
Identifier: void
```

```
Operator: /
Invalid Token: }
Operator: /
Operator: /
Identifier: void
Identifier: add
Keyword: int
Identifier: x
Invalid Token: ,
Keyword: int
Identifier: y
Invalid Token: )
Operator: /
Operator: /
Invalid Token: {
Operator: /
Operator: /
Keyword: return
Identifier: x
Operator: +
Identifier: y
Operator: /
Operator: /
Invalid Token: }
Keyword: int
Identifier: main
Invalid Token: )
Invalid Token: {
Keyword: int
Identifier: a
Operator: =
Constant: 5
Invalid Token: ,
Constant: 7
Identifier: H
Operator: /
Operator: /
Identifier: assign
Identifier: value
Keyword: char
Identifier: b
Operator: =
Invalid Token: '
Identifier: x
Invalid Token: '
Operator: /
Operator: /
Keyword: return
Identifier: value
Keyword: return
Identifier: a
Operator: +
Identifier: b
Invalid Token: }
```