Practical 5

AIM:
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.

TESTCASES:

```
/* salary calculation*/
void main( )
{
long int bs , da , hra , gs;
//take basic salary as input
scanf("%ld",&bs);
//calculate allowances
```

```
// user defined data type
struct student
{
int id;
float cgpa;
}
void main( )
```

```
da=bs*.40;
hra=bs*.20;
gs=bs+da+hra;
// display salary slip
printf("\n\nbs  : %ld",bs);
printf("\nda  : %ld",da);
printf("\nhra : %ld",hra);
printf("\ngs  : %ld",gs);
}
```

```
{
student s;
s.id = 10;
s.cgpa = 8.7;
}
```

```
//function prototype
void add ( int , int );
void main( )
{
int a , b;
a = 10;
b = 20;
// function call
add ( a , b );
}
void add ( int x , int y )
{
return x + y;
}
```

CODE:

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

void printToken(const char* type, const char* value) {
   printf("%s: %s\n", type, value);
}
%}

KEYWORDS
"int"|"char"|"return"|"if"|"else"|"for"|"while"|"do"|"switch"|"case"|"break"|"continue"|"void"|"float"|"double"
IDENTIFIER   [a-zA-Z_][a-zA-Z0-9_]*
CONSTANT     [0-9]+
STRING       \'[^']\'|\"[^\"]*\"
OPERATOR     \+|\-|\*|\/|=|==|!=|<|>|<=|>=|&&|\|\|
PUNCTUATION  [\(\)\{\}\[\],;]
COMMENT      \/\/.|\/\([^]|\+[^\/])\+\/

%%
{KEYWORDS}       { printToken("Keyword", yytext); }
{IDENTIFIER}     { printToken("Identifier", yytext); }
{CONSTANT}       { printToken("Constant", yytext); }
{STRING}         { printToken("String", yytext); }
{OPERATOR}       { printToken("Operator", yytext); }
{PUNCTUATION}    { printToken("Punctuation", yytext); }
{COMMENT}        { /* Ignore comments */ }
[ \t\n]+         { /* Ignore white spaces */ }
.                { printToken("Lexical Error", yytext); } // Detect invalid tokens
%%

int main(int argc, char *argv[]) {
   if (argc < 2) {
      printf("Usage: %s <C source file>\n", argv[0]);
      return 1;
   }

   FILE *file = fopen(argv[1], "r");
   if (!file) {
      printf("Error: Cannot open file %s\n", argv[1]);
      return 1;
   }

   yyin = file;  // Set input file for Lex
   yylex();      // Process file
   fclose(file); // Close file
```

```
    return 0;
}

int yywrap() {
    return 1;
}
```

OUTPUT:

```
Keyword: void
Identifier: main
Punctuation: (
Punctuation: )
Punctuation: {
Identifier: long
Keyword: int
Identifier: bs
Punctuation: ,
Identifier: da
Punctuation: ,
Identifier: hra
Punctuation: ,
Identifier: gs
Punctuation: ;
Identifier: scanf
Punctuation: (
String: "%ld"
Punctuation: ,
Lexical Error: &
Identifier: bs
Punctuation: )
Punctuation: ;
Identifier: da
Operator: =
Identifier: bs
Operator: *
Lexical Error: .
Constant: 40
Punctuation: ;
Identifier: hra
Operator: =
Identifier: bs
Operator: *
Lexical Error: .
Constant: 20
Punctuation: ;
Identifier: gs
```

```
Operator: =
Identifier: bs
Operator: +
Identifier: da
Operator: +
Identifier: hra
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\n\nbs  : %ld"
Punctuation: ,
Identifier: bs
Punctuation: )
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\nda  : %ld"
Punctuation: ,
Identifier: da
Punctuation: )
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\nhra : %ld"
Punctuation: ,
Identifier: hra
Punctuation: )
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\ngs  : %ld"
Punctuation: ,
Identifier: gs
Punctuation: )
Punctuation: ;
Punctuation: }
```