**EXERCISE 2: IMPLEMENTATION OF LEXICAL ANALYZER USING LEX (Token Generation)**

**AIM:** To perform token separation by writing patterns and actions using LEX.

**ALGORITHM:**

1. Declare the necessary symbols and header files needed.

2. Write the pattern and corresponding actions to be taken

2.1. if the pattern is #,\* print it as a header file

2.2. if the pattern is int|float|char|return|main|… print it as a keyword

2.3. if the pattern is [“][a-zA-z][“] print it as a string

2.4. Similarly identify patterns and actions for all tokens.

3. In the main function declare a file pointer to read input file and call yylex() to find matching patterns and corresponding actions to be taken.

**CODE:**

**%option noyywrap**

**%{**

**#include<stdio.h>**

**void yyerror(char \*);**

**%}**

**letter [a-zA-Z]**

**digit [0-9]**

**op [-+\*/]**

**punct [,.;"]**

**%%**

**else|if|void|int {printf("%s is a keyword",yytext);}**

**{digit}+ {printf("%s is a number",yytext);}**

**{letter}({letter}|{digit})\* {printf("%s is an identifier",yytext);}**

**{op} {printf("%s is an operator",yytext);}**

**[ ] ;**

**\) {printf("%s is close parenthesis",yytext);}**

**{punct} {printf("%s is a punctuation",yytext);}**

**. yyerror("error");**

**%%**

**void yyerror(char \*s)**

**{**

**fprintf(stderr,"%s\n",s);**

**}**

**int main(int argc, char \*argv[])**

**{**

**FILE \*fp;**

**if((fp=fopen(argv[1],"r"))==NULL)**

**{printf("file does not exist");}**

**yyin=fp;**

**yylex();**

**return 0;**

**}**

**Result :**