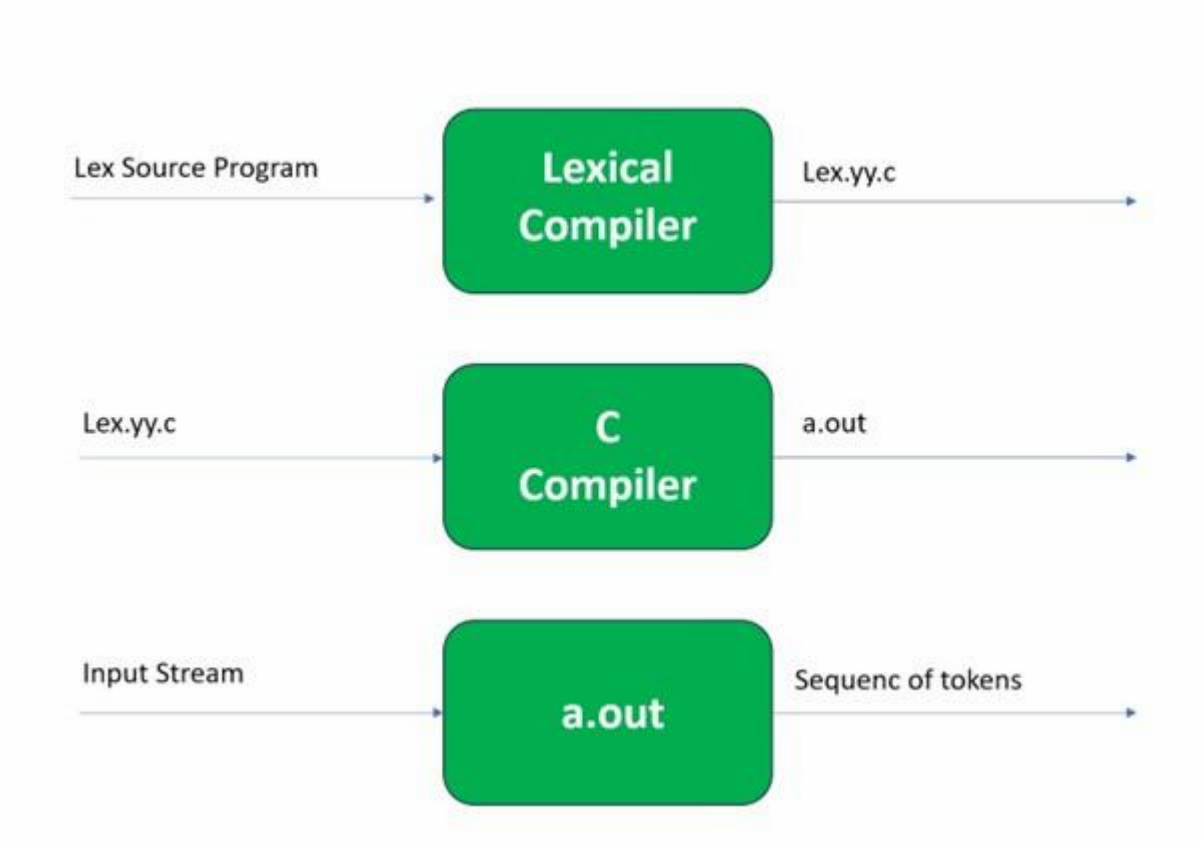
LAB 2: Write a flex program to identify keywords, numbers, operators and identifiers.

Theory:

Lex is a tool or a computer program that generates Lexical Analyzers (converts the stream of characters into tokens). The lex compiler takes the input and transforms that input into input patterns.



Here first source code is lex language with file name with extension '.l' is given as input to Lex compiler to get output as lex.yy.c.

Then lex.yy.c is used as input to the Compiler which gives the output in the form of a.out and finally a.out will take the streams of character and generate tokens as output.

Lex File Format

Lex program consists of three parts and is separated by %% delimiters.

Declarations

%%

Translation rules

%%

Auxiliary Procedures

Declarations: It includes declarations of variables.

Transition rules: These rules consist of Pattern and Action.

Auxiliary Procedures: The auxiliary section holds auxiliary functions used in the actions.

Lex program to count number of words

Download identification.l file for source code.

Place Screenshot of Command Line and Output Here using **Cygwin**

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

#define MAX_KEYWORDS 10

// List of keywords
char *keywords[] = {"int", "float", "if", "else", "while", "for", "return", "void", "char", "double"};

int isKeyword(const char *str) {
    for (int i = 0; i < MAX_KEYWORDS; i++) {
        if (strcmp(str, keywords[i]) == 0)
            return 1;
    }
```

```
    return 0;
}
%}

%%
[a-zA-Z_][a-zA-Z0-9_]* {
    if (isKeyword(yytext)) {
        printf("Keyword: %s\n", yytext);
    } else {
        printf("Identifier: %s\n", yytext);
    }
}
[0-9]+(\.[0-9]+)? {
    printf("Number: %s\n", yytext);
}
[+\-*/=<>!&|] {
    printf("Operator: %s\n", yytext);
}
[ \t\n]+ ; // Ignore whitespaces
. {
    printf("Unknown character: %s\n", yytext);
}
%%

int main(int argc, char *argv[]) {
    printf("Enter the input: ");
    yylex(); // Start scanning input
    return 0;
```

```
}
int yywrap() {
    return 1;  // Indicates the end of input
}
```