32. Write a LEX Program to convert the substring abc to ABC from the given input string

%{

#include <stdio.h>

%}

%%

abc { printf("ABC"); } // If "abc" is encountered, print "ABC"

. { printf("%s", yytext); } // Otherwise, print the character as it is

%%

int main() {

yylex(); // Start the lexical analysis

return 0;

}  
33. Implement Lexical Analyzer using FLEX (Fast Lexical Analyzer). The program should separate the tokens in the given C program and display with appropriate caption.

%{

#include <stdio.h>

#include <string.h>

void print\_token(char\* type, char\* value) {

printf("%s: %s\n", type, value);

}

%}

%%

int|long|short|void|char|float|double|if|else|while|for|return|struct|union|typedef { print\_token("Keyword", yytext); }

[0-9]+ { print\_token("Integer", yytext); }

\"[^\"]\*\" { print\_token("String", yytext); }

[+-/\*=<>!&|^%] { print\_token("Operator", yytext); }

[ \t\n]+ { /\* Ignore spaces, tabs, and newlines \*/ }

[A-Za-z\_][A-Za-z0-9\_]\* { print\_token("Identifier", yytext); }

[(){};,] { print\_token("Punctuation", yytext); }

%%

int main() {

yylex(); // Start the lexical analysis

return 0;

}  
34. Write a LEX program to separate the keywords and identifiers.

%{

#include <stdio.h>

void print\_keyword(char \*value) {

printf("Keyword: %s\n", value);

}

void print\_identifier(char \*value) {

printf("Identifier: %s\n", value);

}

%}

%%

int|long|short|void|char|float|double|if|else|while|for|return|struct|union|typedef { print\_keyword(yytext); }

[A-Za-z\_][A-Za-z0-9\_]\* { print\_identifier(yytext); }

%%

int main() {

FILE \*file = fopen("input.c", "r"); // Open the C source file

if (file) {

yyin = file; // Set the input file for Lex

yylex(); // Start lexical analysis

fclose(file); // Close the file after processing

} else {

printf("Unable to open the file.\n");

return 1;

}

return 0;

}