MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE CODE: CSE-304, COURSE TITLE: COMPILER SESSIONAL SPRING 2025

Assignment on Tokenizer Using Flex

Write a lexical analyzer for C
– Ignore white space
– Match all keywords
– Match all identifiers(variables)
– Match all functions
– Match all operators
– Match all numbers
– Match parentheses, curly braces
– Match comma, semicolon, colon, etc.
– Match literals – Count line numbers

Lexemes	**Example**	**Token**
Variables	Variables contains any alphanumeric character or (_) but cannot start with a digit Ex : a, a9bc, _abc but not 8cde.	Token name is Identifier Ex: Token: <Identifier, _abc>
Numbers	Numbers may contain optional fraction or exponent Ex: 10E2, .25, 3, 3.056, 3.45E5, 3.45E-2, 3E+2	Token name is Number Ex: Token: <Number, 3.056>
Bitwise operators	>>, <<, &, ^, |	Token name is Bitwise Operator Ex: Token :<Bitwise Operator, +>
Assignment Operator	=, <<=, >>=, &=, ^=, |=	Token name is Assignment Operator Ex: Token :< Assignment Operator, =>
Function	Any Function name maintaining the same convention of variable name. printf("Hello World");	Token name is Function. Ex: Token: <Function, "printf">
Keywords	if, else, switch, case, while, for, int, float, double, break, default, void	Token name is Keyword Ex: Token is <Keyword, if>
Parentheses	(or)	Token name is Parentheses Ex: Token: < Parentheses , (>
curly braces	{ or }	Token name is curly Ex: Token: < curly , }>
comma, semicolon, colon	, ; :	Token name is separator Ex: Token: <Separator, :>
String Literals	"Anything….."	Token name is String literal Ex: Token: <String Literal, "Anything…">
Comment Single/ Multiple line	// This is comment /* This is Multiple line comment*/	The comment should be skipped.

Notes:
1. For each token, print the corresponding token name and line no of occurring.
2. Single line or multiple line comments should be skipped.
3. Take input from file and give output to file