

# Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

## ❧ CERTIFICATE ❧

Certify that Mr./Miss Pswita Roshan. Hondsamalan  
of COMPUTER Department, Semester VI with  
Roll No. 2108058 has completed a course of the necessary  
experiments in the subject SPCC under my  
supervision in the **Thadomal Shahani Engineering College**  
Laboratory in the year 2023 - 2024

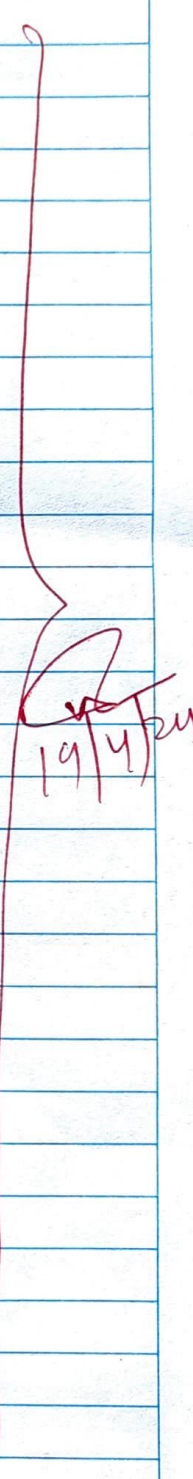
  
Teacher In-Charge

Head of the Department

Date 19/04/2024.

Principal

## CONTENTS

Sr. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.	Write a program to implement lexical Analyser.		19/01	
2.	To study & implement programs LEX & YACC tool.		02/02	
3.	Write a program to implement the FIRST & FOLLOW set for the given grammar		09/02	
4.	Write a program to implement Parser		16/02	
5.	Write a program to implement YAC		01/03	
6.	Write a program to implement code optimization.		15/03	
7.	a) Write a program to implement pass 1 of Multi-Pass Assembler b) Write a program to implement Pass 2 of Multi-Pass Assembler		16/03	
8.	Write a program to implement MULTIPass Macroprocessor		22/03	
9.	Write a program to implement code Generation		06/04	
10.	Write a program to eliminate left recursion of the given grammar.		06/04	
11.				







## EXPERIMENT No.1.

Aim: Write a program to implement Lexical Analyzer.

### Theory:

Lexical Analyzer is the first phase of the compiler, also known as a scanner. It converts high level input program into a sequence of tokens.

Lexical analysis can be implemented with the DFA.

### Tokens:

A lexical token is a sequence of characters that can be treated as a unit in the grammar of programming languages.

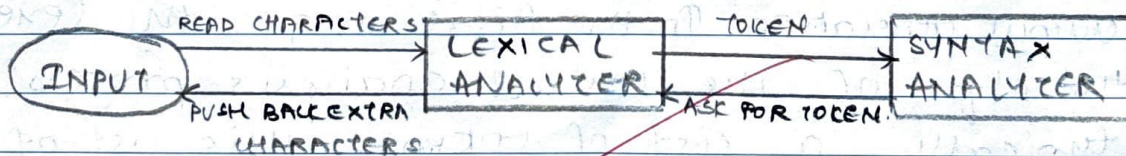
### Eg of tokens:

Keywords: for, while, if, int, etc.

Identifier: variable-name, function name, etc.

Operators: '+', '-', '++', '\*', etc.

Separators: ',', ';', etc.



Lexeme: The sequence of characters matched by a pattern to form the corresponding token or a sequence of input characters that comprises of a single token is called Lexeme.



## Working of a Lexical Analyser:

- ① Input preprocessing: this stage involves cleaning up the input text & preparing it for analysis. this may include removing comments, whitespace, etc. from input text.
- ② Tokenization - It is a process of breaking the input text into a sequence of tokens. This is usually done by matching the characters in the text against a set of patterns that define the different types of tokens.
- ③ Token classification: On this stage, the lexer checks the type of each token in prog. lang. the lexer might classify keywords, identifiers, operators, etc. as separate token types.
- ④ Token validation: In this, the lexer checks that each token is valid according to the rules of prog. lang. Eg: It might check that a variable name is valid identifier or an operator, etc.
- ⑤ Output Generation: In the final stage, the lexer generates the o/p of the lexical analysis process, which is typically a list of tokens. this list of tokens can then be passed to the next stage of compilation or interpretation.

Eg: i/p: `int a = b * c = 3;`

o/p: no. of tokens = 9

variables: a, b, c

operators: =, \*, -

keywords: int

numbers: 3

separator: ;

✖

12/24