

**SAVEETHA SCHOOL OF ENGINEERING**

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**LIST OF EXPERIMENTS**

COURSE CODE :     CSA14

COURSE NAME :     Compiler Design

Course Coordinator: Dr.W.Deva Priya

**Software Required :**

1. FLEX Packages
2. Dev C

**Topics – I**

1. Develop a lexical analyzer that, in addition to tokenizing the input, extracts and stores attributes of tokens. For instance, for identifiers, it should store their names; for constants, it should store their values; and for operators, it should store their types..
2. Implement a lexical analyzer that can recognize comments defined by // for single-line and /\* ... \*/ for block comments in C. The analyzer should be able to identify these comments correctly in the given code and report their presence.
3. Create a lexical analyzer that scans a code file to recognize and validate the usage of arithmetic operators (+, -, \*, /). The analyzer should accurately detect these operators and verify their correct usage within the context of the code.
4. Implement a lexical analyzer that scans a program file to determine the frequency of whitespace and newline characters. The analyzer should output the total number of spaces and newlines found in the file.
5. Design a lexical analyzer that processes an input text to verify the validity of each identifier based on the naming conventions of the target programming language.
6. Write a C program that converts a grammar with left recursion into a non-left-recursive grammar to facilitate top-down parsing techniques.

S → Sa / b

1. Implement a C program that eliminates left factoring from a given context-free grammar to make it suitable for predictive parsing.

A → aB / aC

1. Create a C program to build and maintain a symbol table, supporting operations like adding new symbols, removing symbols, and retrieving symbol information
2. The lexical analyzer should ignore redundant spaces, tabs, and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Develop a lexical Analyzer to identify identifiers, constants, and operators using the C program.
3. Extend the lexical Analyzer to Check comments, defined as follows in C:

a) A comment begins with // and includes all characters until the end of that line.

b) A comment begins with /\* and includes all characters through the next occurrence of the character sequence \*/Develop a lexical Analyzer to identify whether a given line is a comment or not.

1. Design a lexical Analyzer to validate operators to recognize the operators +,-,\*,/ using regular Arithmetic operators.
2. Design a lexical Analyzer to find the number of whitespaces and newline characters.
3. Develop a lexical Analyzer to test whether a given identifier is valid or not.
4. Implement a C program to eliminate left recursion.
5. Implement a C program to eliminate left factoring.
6. Implement a C program to perform symbol table operations.

**Topics – II**

1. Implement a program to determine whether an input string adheres to the syntactic rules defined by a context-free grammar.
2. Develop a C program to construct a recursive descent parser for a given simple grammar, ensuring correct parsing of input strings.

S → bA

A → aA / ε

1. Write a C program to parse and evaluate mathematical expressions using the rules of operator precedence (PEMDAS: Parentheses, Exponentiation, Multiplication, Division, Addition, Subtraction)
2. Develop a C program that generates the three-address code representation for a given arithmetic expression.

a + b + c

1. Write a C program to analyze a text input and calculate the total number of characters, words, and lines.
2. Write a C program that optimizes code by identifying and eliminating redundant computations to improve efficiency.
3. Write a C program to generate machine code from intermediate representations, simulating the backend phase of a compiler.
4. Write a LEX specification to analyze a C program from a .c file, ignoring unnecessary whitespace and comments, to count the number of characters, lines, and words. **Input Source Program: (sample.c)**

#include <stdio.h>

int main()

{

int number1, number2, sum;

printf("Enter two integers: ");

scanf("%d %d", &number1, &number2);

sum = number1 + number2;

printf("%d + %d = %d", number1, number2, sum);

return 0;

}

1. All languages have Grammar. When people frame a sentence we usually say whether the sentence is framed as per the rules of the Grammar or Not. Similarly use the same ideology , implement to check whether the given input string is satisfying the grammar or not .
2. Write a C program to construct recursive descent parsing.
3. In a class of Grade 3, Mathematics Teacher asked for the Acronym PEMDAS?. All of them are thinking for a while. A smart kid of the class Kishore of the class says it is Parentheses, Exponentiation, Multiplication, Division, Addition, Subtraction. Can you write a C Program to help the students to understand about the operator precedence parsing for an expression containing more than one operator, the order of evaluation depends on the order of operations.
4. The main function of the Intermediate code generation is producing three address code statements for a given input expression. The three address codes help in determining the sequence in which operations are actioned by the compiler. The key work of Intermediate code generators is to simplify the process of Code Generator. Write a C Program to Generate the Three address code representation for the given input statement.
5. Write a C program for implementing a Lexical Analyzer to Count the number of characters, words, and lines .
6. Write a C Program for code optimization to eliminate common subexpression.
7. Write a C program to implement the back end of the compiler.
8. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Write a LEX specification file to take input C program from a .c file and count t he number of characters, number of lines & number of words.

**Input Source Program: (sample.c)**

#include <stdio.h>

int main()

{

int number1, number2, sum;

printf("Enter two integers: ");

scanf("%d %d", &number1, &number2);

sum = number1 + number2;

printf("%d + %d = %d", number1, number2, sum);

return 0;

}

**Topics -III**

1. Write a LEX program to print all the constants in the given C source program file.

**Input Source Program: (sample.c)**

#define PI 3.14

#include<stdio.h> #include<conio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}

1. Write a LEX program to count the number of Macros defined and header files included in the C program.

**Input Source Program: (sample.c)**

#define PI 3.14

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}

1. Write a LEX program to print all HTML tags in the input file.

**Input Source Program: (sample.html)**

<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

1. Write a LEX program which adds line numbers to the given C program file and display the same in the standard output.

**Input Source Program: (sample.c)**

#define PI 3.14

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}

1. Write a LEX specification count the number of characters, number of lines & number of words.
2. Write a LEX program to count the number of comment lines in a given C program and eliminate them and write into another file.

**Input Source File: (input.c)**

#include<stdio.h>

int main()

{

int a,b,c; /\*varible declaration\*/ printf(“enter two numbers”); scanf(“%d %d”,&a,&b); c=a+b;//adding two numbers printf(“sum is %d”,c);

return 0;

}

1. Write a LEX program to identify the capital words from the given input.
2. Write a LEX Program to check the email address is valid or not.
3. Students are tasked with writing a LEX program that reads a C source file and prints out all the constants found in the file. The constants can be integer constants, floating-point constants, or string literals. The program should ignore comments and other non-constant tokens.
4. You are developing a static code analysis tool to help C programmers better understand and manage their code. One of the features requested by your users is the ability to count the number of macros defined and the number of header files included in their C source programs. Your task is to write a LEX program that reads a C source file and counts the number of #define macro definitions and #include header file inclusions.
5. You are working on a web scraping tool that needs to process HTML files. One of the features you want to implement is the ability to extract and print all HTML tags from an input HTML file. This feature will help in understanding the structure of the HTML content and in performing further analysis or manipulation of the tags.
6. As part of a code review tool, you need a feature that automatically adds line numbers to a given C source file. This feature will help reviewers easily reference specific lines of code. Your task is to write a LEX program that reads a C program file, adds line numbers to each line, and displays the output with line numbers on the standard output.
7. You are tasked with developing a command-line tool that analyzes text files and provides statistics such as the number of characters, lines, and words in the text. You decide to use Lex to implement the lexical analysis part of your tool. Your task is to write a Lex specification that counts the number of characters, lines, and words in the input text file. Your Lex specification should: a) Count the number of characters (including spaces, tabs, and newline characters) b) . Count the number of lines. c) Count the number of words (words are defined as sequences of alphabetic characters separated by whitespace).
8. You're working on a project where you need to analyze C source code files. Your task is to develop a tool that counts the number of comment lines in a C program and creates another file without comments. Write a Lex program to accomplish this task. Your program should:Count the number of comment lines (both single-line and multi-line comments).Eliminate comments from the input C program and write the remaining code into another file.
9. You are developing a text processing tool that identifies capital words from a given input text. Capital words are words that start with an uppercase letter and are followed by zero or more lowercase letters. Write a Lex program (capital\_word\_identifier.l) to identify capital words from the input text. Your program should: a) Identify capital words in the input text. b) Output each identified capital word on a separate line.

Ensure your program correctly handles words with mixed cases (e.g., "OpenAI", "HelloWorld"). Provide appropriate error handling for file not found or any other relevant errors.

1. You're building a web application where users can sign up using their email addresses. Before allowing users to register, you need to validate the email addresses they provide. Write a Lex program (email\_validator.l) to check whether an email address is valid or not. Your program should: Check if the email address follows the standard format: username@domain.extension.Ensure that username and domain parts contain only letters, digits, dots (.), hyphens (-), and underscores (\_). The extension part should contain only letters and can be 2 to 4 characters long. Allow uppercase and lowercase letters in the username and domain, but the extension should be lowercase. Ensure there are no spaces in the email address. Output "Valid email address" if the email is valid, otherwise output "Invalid email address". Provide appropriate error handling for invalid inputs.

**Topics – IV**

1. Write a LEX Program to convert the substring abc to ABC from the given input string.
2. The Company ABC runs with employees with several departments. The Organization manager had all the mobile numbers of employees. Assume that you are the manager and need to verify the valid mobile numbers because there may be some invalid numbers present. Implement a LEX program to check whether the mobile number is valid or not.
3. Implement Lexical Analyzer using LEX or FLEX (Fast Lexical Analyzer). The program should separate the tokens in the given C program and display with appropriate caption.

**Input Source Program: (sample.c)**

#include<stdio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}

1. In a class, an English teacher was teaching the vowels and consonants to the students. She says “Vowel sounds allow the air to flow freely, causing the chin to drop noticeably, whilst consonant sounds are produced by restricting the air flow”. As a class activity the students are asked to identify the vowels and consonants in the given word/sentence and count the number of elements in each. Write an algorithm to help the student to count the number of vowels and consonants in the given sentence.
2. Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier. In general there are 32 keywords. The prime function of Lexical Analyser is token Generation. Among the 6 types of tokens, differentiating Keyword and Identifier is a challenging issue. Thus write a LEX program to separate keywords and identifiers.
3. Write a LEX program to recognise numbers and words in a statement. Pooja is a small girl of age 3 always fond of games. Due to the pandemic, she was not allowed to play outside. So her mother designs a gaming event by showing a flash card. Pooja has to separate the numbers in one list and words in another list shown in the flash card.
4. Write a LEX program to identify and count positive and negative numbers.
5. A networking company wants to validate the URL for their clients. Write a LEX program to implement the same.
6. You're developing a text transformation tool that converts all occurrences of the substring "hello world" to "HELLO WORLD" in a given input text. Write a Lex program (substring\_converter.l) to perform this conversion. Your program should: Identify all occurrences of the substring "hello world” (case insensitive) in the input text. Replace each occurrence with "Hello World”.Output the modified text.
7. As the manager of Company ABC, you need to verify the validity of mobile numbers stored in the employee database. You decide to develop a tool using Lex to check whether the mobile numbers are valid or not. Write a Lex program (mobile\_number\_validator.l) to perform this validation. Your program should:Identify and validate mobile numbers based on the following criteria:Must start with a country code (e.g., +91 for India).Must contain 10 digits after the country code.Allow hyphens (-) or spaces between digits for readability. Output "Valid mobile number" for valid mobile numbers and "Invalid mobile number" for invalid ones.
8. You're developing a C program tokenizer tool that separates various tokens in a given C program and displays them with appropriate captions. Write a Flex program (c\_tokenizer.l) to tokenize a C program. Your program should: Identify and categorize different tokens such as keywords, identifiers, constants, operators, etc. Display each token along with its appropriate caption.
9. The English teacher has explained the concept of vowels and consonants to the class. Now, it's time for a hands-on activity! Write an algorithm to help the students count the number of vowels and consonants in a given sentence. Algorithm: Count Vowels and Consonants Start. Take the input sentence from the user. Initialize variables vowelCount and consonantCount to zero. Loop through each character in the input sentence: If the character is an alphabet (a-z or A-Z): Convert the character to lowercase for uniformity. If the character is 'a', 'e', 'i', 'o', or 'u', increment vowel count. Otherwise, increment consonantCount.Print the counts of vowels and consonants. End.
10. As a programming student, you've learned about keywords and identifiers. Now, you're tasked with developing a tool to separate keywords and identifiers from a given input. Write a Lex program (keyword\_identifier\_differentiator.l) to accomplish this task. Your program should: Differentiate between keywords and identifiers. Output each token along with its appropriate caption.
11. Sriram's mother wants to organize a fun gaming event using flashcards. Sriram's task is to separate the numbers and words shown in the flashcards into different lists. Write a Lex program (flashcard\_recognizer.l) to help sriram separate numbers and words from a given statement. Your program should: Recognize numbers and words. Output each number and word into separate lists.
12. You're working on a project where you need to identify and count the occurrences of positive and negative numbers in a dataset. Write a Lex program (number\_counter.l) to accomplish this task. Your program should: Identify positive and negative numbers. Count the occurrences of positive and negative numbers separately. Output the counts of positive and negative numbers.
13. A networking company wants to ensure that the URLs provided by their clients are valid before integrating them into their systems. You've been assigned to develop a tool to validate URLs using Lex. Write a Lex program (url\_validator.l) to implement URL validation. Your program should: Validate URLs according to basic URL format rules. Output "Valid URL" for valid URLs and "Invalid URL" for invalid ones.

**Topics – V**

1. School management wants to validate DOB of all students. Write a LEX program to implement it.
2. Write a LEX program to check whether the given input is digit or not.
3. A School student was asked to do basic mathematical operations. Implement a LEX program to implement the same.
4. Write a [LEX program to accept string starting with vowel](https://copyprogramming.com/howto/what-s-the-difference-between-flex-start-and-baseline).
5. Write a LEX [program to find the length of the longest](https://copyprogramming.com/howto/python-find-longest-orf-in-dna-sequence) word.
6. Write a LEX program to count the frequency of the given word in a given sentence.
7. Write a LEX code to replace a word with another word in a file.
8. Write a LEX program to recognize a word and relational operator.
9. The school management wants to ensure that the Date of Birth (DOB) provided by students is in a valid format before storing it in their records. Write a Lex program (dob\_validator.l) to implement DOB validation. Your program should: Validate the DOB according to a specified format (e.g., dd/mm/yyyy). Output "Valid DOB" for valid DOB entries and "Invalid DOB" for invalid ones.
10. Scenario: Digit Checker. Suppose you're developing a tool to check whether the given input is a digit or not. Let's create a scenario around it: You're developing a tool that checks whether the input provided by the user is a digit or not. Write a Lex program (digit\_checker.l) to implement this functionality. Your program should: Check whether the input is a digit.Output "Digit" if the input is a digit and "Not a Digit" otherwise.
11. Suppose you're developing a tool to perform basic mathematical operations such as addition, subtraction, multiplication, and division. Let's create a scenario around it: Scenario: Basic Mathematical Operation Evaluator You're developing a tool to perform basic mathematical operations based on user input. Write a Lex program (math\_operations.l) to implement this functionality. Your program should: Recognize basic mathematical operations (+, -, \*, /). Evaluate the expression and output the result.
12. You're developing a tool to check if a given string starts with a vowel. Write a Lex program (vowel\_checker.l) to implement this functionality. Your program should: Check if the input string starts with a vowel. Output "Starts with Vowel" if it starts with a vowel, and "Doesn't start with Vowel" otherwise
13. Scenario: Longest Word Length Finder . You're developing a tool to find the length of the longest word in a given input. Write a Lex program (longest\_word\_length.l) to implement this functionality. Your program should: Identify words in the input. Find the length of the longest word. Output the length of the longest word found.
14. You're developing a tool to count the frequency of a given word in a given sentence. Write a Lex program (word\_frequency\_counter.l) to implement this functionality. Your program should: Identify words in the input sentence. Count the frequency of a given word. Output the frequency of the given word.
15. You're developing a tool to replace a specific word with another word in a file. Write a Lex program (word\_replacement.l) to implement this functionality. Your program should: Identify words in the input file. Replace a specific word with another word. Output the modified content with the replaced word.
16. In the Scenario: Word and Relational Operator Recognizer. You're developing a tool to recognize words and relational operators in a given input. Write a Lex program (word\_relational\_operator.l) to implement this functionality. Your program should: Identify words and relational operators in the input. Output the type of token for each recognized word and relational operator.