GRAPHIC ERA HILL UNIVERSITY, DEHRADUN
SEMESTER VI

Name of Department: - Computer Science and Engineering

1. Subject Code: **PCS-601** Course Title: **Compiler Design Lab**
2. Contact Hours: **L:** 0 **T:** 0 **P:** 4
3. Examination Duration (Hrs): **Theory:** 0 **Practical:** 3
4. Relative Weight: **CIE:** 25 **MSE:** 25 **SEE:** 50
5. Credits: 2
6. Semester: 6
7. Category of Course: DC
8. Pre-requisite: **TCS-402, PCS-251**

9. Course Outcome:	After completion of the course the students will be able to: **CO1:** Construct lexical analyzer and parser layout by using modern tools like Flex and Bison. **CO2:** Explore the different finite automata problems with the help of tools. **CO3:** Compare and contrast various parsing techniques such as SLR, CLR, LALR with the help of bison tool. **CO4:** Analyze the syntax rules by designing the syntax trees from different aspects of programming languages.

10. **Details of the Course:**

Sr. No	List of problems for which student should develop program and execute in the Laboratory	Contact Hrs
1	Study about Lex and Yacc tools. **Hint:** In this, students need to write the explanation about the structure of lex, structure of lex program and structure of yacc as well. Moreover, they also required the explanation about the pre-defined patterns in the form of regular expressions and set of auxiliary functions.	2
2	Design a LEX Code to count the number of lines, space, tab-meta character, and rest of characters in given Input pattern. Sample Input: Hello and Welcome to Compiler Design Lab	2

	Sample Output: The number of lines are: 2 The number of spaces are: 3 The number of tab-meta characters are: 2 The rest of the characters are: 34	
3	Design a LEX Code to identify and print valid Identifier of C in given Input pattern. Sample Input: 1) Enter any identifier you want to check: **abc** 2) Enter any identifier you want to check: **abc123** 3) Enter any identifier you want to check: **abc_12** 4) Enter any identifier you want to check: **_abc12** 5) Enter any identifier you want to check: **12abc** 6) Enter any identifier you want to check: **ab@c** Sample Output: 1) This is a Valid Identifier 2) This is a Valid Identifier 3) This is a Valid Identifier 4) This is a Valid Identifier 5) This is not a Valid Identifier 6) This is not a Valid Identifier	2
4	Design a LEX Code to identify and print integer and float value in given Input pattern. Sample Input: 1) Enter any number you want to check: **10** 2) Enter any number you want to check: **1.5** 3) Enter any number you want to check: **abc** Sample Output: 1) This is an Integer Number 2) This is a Floating Point Number 3) This is not a Valid Number	2
5	Design a LEX Code for Tokenizing (Identify and print OPERATORS, SEPERATORS, KEYWORDS, and IDENTIFERS) in the given input: Sample Input: Write Your C Program for Tokenizing: int a, b = 10; Sample Output: int is a keyword. a is an identifier. , is a separator. b is an identifier. = is an operator. 10 is a constant. ; is a separator.	2

6	Design a LEX Code to count and print the number of total characters, words, and white spaces in given 'Input.txt' file. Sample Input.txt file: Hello and Welcome to the CD Lab Sample Output: The number of total characters in the given input file is: 31 The number of total words in the given input file is: 7 The number of total white spaces in the given input file is: 6	2
7	Design a LEX Code to replace white spaces of 'Input.txt' file by a single blank character into 'Output.txt' file. Sample Input.txt: Lex is a program that generates lexical analyzer. It is used with YACC parser generator. The lexical analyzer is a program that transforms an input stream into a sequence of tokens. Sample Output.txt: Lex is a program that generates lexical analyzer. It is used with YACC parser generator. The lexical analyzer is a program that transforms an input stream into a sequence of tokens.	2

8	Design a LEX Code to remove the comments from any C Program given at run-time and store into 'out.c' file.
	Sample Input:
	int a = 10;
	int b = 20;
	// int sum = a + b;
	printf("%d", sum);
	
	Sample Output:
	int a = 10;
	int b = 20;
	printf("%d", sum);
		2

9	Design a LEX Code to extract all html tags in the given HTML file at run time and store into Text file given at run time.	2
10	Design a LEX Code to recognize and print the following tokens: a) string b) keywords c) constants d) identifiers e) literals	2

11	Design a LEX Code to take check whether the given number is even or odd. Sample Input: Enter a string: 25 Enter a string: 54 Enter a string: 25ab Sample Output: 25 is an odd number. 54 is an even number. 25ab is not a valid number.	2
12	Design a LEX Code to count number of vowels and consonants in a given pattern. Sample Input: Enter a string: This is a LEX Code Sample Output: Total number of Vowels are: 6 Total number of Consonants are: 8	2
13	Design a LEX Code to check for a valid E-mail Id. Sample Input: Enter a string: abc123@gmail.com Enter a string: abc123.ss@gmail.com Enter a string: abc123#ss@gmail.com Sample Output: abc123@gmail.com is a valid email. abc123.ss@gmail.com is a valid email. abc123#ss@gmail.com is not a valid email.	2
14	Design a DFA in LEX Code which accepts all possible set of string containing even number of 'a' and even number of 'b' over input alphabet Σ = {a, b}. Hint: Construct an appropriate DFA for the given problem then write the lex program accordingly. Sample Input: Enter a string: **ε** Enter a string: ab Enter a string: aa Enter a string: bb Enter a string: aabb Sample Output: **ε** is accepted. ab not accepted aa is accepted. bb is accepted. aabb is accepted.	2
15	Design a DFA in LEX Code to Identify and print Integer, Float Constants and Identifier. Hint: Construct an appropriate DFA for the given problem then write the lex program accordingly.	2

16	Design a DFA in LEX Code over Σ = {a, b} which contains set of all possible strings where every string starts with a and ends with b. Hint: Construct an appropriate DFA for the given problem then write the lex program accordingly.	2
17	Design YACC / LEX code to recognize valid arithmetic expression with operators **+**, **-**, ***** and **/**	2
18	Design YACC / LEX code to evaluate arithmetic expression involving operators **+**, **-**, ***** and **/** without operator precedence grammar and with operator precedence grammar.	2
19	Design YACC / LEX code that translates INFIX Expression to POSTFIX Expression.	2
20	Design a Desk Calculator using YACC / LEX code.	2