CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH

Subject : Design of Language Processor **Semester:** 7

Subject Code: CS450 **Academic Year :**2023-24(ODD)

Course Outcome (COs):

At the end of the course, the students will be able to:

CO1	Understand fundamentals of compiler and identify the relationships among different phases of the compiler and use the knowledge of the Lex tool
CO2	Describe Role of Parser and the various error recovery strategies
CO3	Develop the parsers and experiment with the knowledge of different parsers design.
CO4	Design syntax directed translation schemes for a given context free grammar
CO5	Develop semantic analysis scheme to generate intermediate code
CO6	Summarize various optimization techniques used for dataflow analysis and generate machine code from the source code

Practical List

Sr. No.	AIM	Hrs.	Cos.
1	Write a Lex program to count the number of characters, words and lines in the given input. **Supplementary Experiment:** **1.** Implement a lexical analyzer for a subset of Java using LEX. Implementation should support Error handling. [**L: M**]	02	CO1
2	Implement a lexical analyzer for identification of numbers. **Supplementary Experiments:** **1.** It gives the students an idea about how to specify regular expressions for covering all possible formats of floating point numbers. **[L: M]** 2. Implement a Lexical Analyzer which copies a file, replacing each nonempty sequence of white spaces by a single blank. [**L: A**]	02	CO2
3	Implement a Calculator using LEX and YACC. **Supplementary Experiment:**	02	CO1

	Perform control flow analysis to analyze loops, conditionals, and function calls. This analysis can be used to optimize the code, detect unreachable code, and apply loop transformations. **[L: M]**		
4	Implement a program to identify keywords and identifiers using finite automata. **Supplementary Experiment:** Test with different programing languages like python, java, c++ etc. [**L: M**]	02	CO1
5	Write an ambiguous CFG to recognize an infix expression and implement a parser that recognizes the infix expression using YACC **Supplementary Experiment:** Generate an intermediate representation (e.g., three-address code) from the AST. Implement code to traverse the AST and emit the intermediate code accordingly. **[M: A]**	02	CO2
6	Implement a C program to find FIRST and FOLLOW set of given grammar. **Supplementary Experiment:** Test with different grammar and evaluate correctness of the calculated FIRST and FOLLOW sets by manually computing them for a few non-terminals and comparing the results with your program's output. [**L: A**]	02	CO3
7	Write a program to remove the Left Recursion from a given grammar. **Supplementary Experiment:** Add type-checking to your compiler. Implement type inference and perform checks for type compatibility and consistency throughout the code. [**L: A**]	02	CO3
8	Implementation of Context Free Grammar. **Supplementary Experiment:** Optimization techniques: Apply basic optimization techniques, such as constant folding, common subexpression elimination, and dead code elimination, to improve the efficiency of the generated code. **[M: A]**	02	CO4
9	Implementation of code generator. **Supplementary Experiment:** Compare with existing code generators: Evaluate the quality, readability, and efficiency of the generated code. [**L: M**]	02	CO6
10	Implementation of code optimization for Common sub-expression elimination, Loop invariant code movement. **Supplementary Experiment:** Apply the code optimization techniques to various code examples with different levels of complexity. Experiment with simple arithmetic operations, conditional statements, nested loops, and function calls. [**L: A**]	02	CO5

Initialisms:

L: Level

M: Medium

A: Advanced

Note: The levels medium and advanced show the understanding, problem solving and application of the domain knowledge.

Submission of supplementary experiments would be done separately**.**