CVM UNIVERSITY
Aegis: Charutar Vidya Mandal (Estd.1945)

FACULTY OF ENGINEERING & TECHNOLOGY

Effective from Academic Batch: 2020-21

Programme: **Bachelor of Engineering (Information Technology)**

Semester: **VII**

Course Code: **102046711**

Course Title: **Language Processors**

Course Group: **Professional Elective Course - III**

Course Objectives: This course provides the fundamental knowledge of the various aspects of system software and enables students to understand the basic concepts of language processors. It describes the steps and algorithms used by language translators. Recognize the underlying formal models such as finite state automata, push-down automata, and their connection to language definition through regular expressions and grammars.

Teaching & Examination Scheme:

Contact hours per week			Course Credits	Examination Marks (Maximum / Passing)				
				Theory		J/V/P*		Total
Lecture	Tutorial	Practical		Internal	External	Internal	External	
3	0	2	4	40 / 14	60 / 21	20 / 7	30 / 10	150 / 52

* **J**: Jury; **V**: Viva; **P**: Practical

Detailed Syllabus:

Sr.	Contents	Hours
1	**Introduction of Language Processor:** Programming Languages and Language Processors, Language Processing Activities, Program Execution, Fundamental of Language Processing, Symbol Tables Data Structures for Language Processing: Search Data structures, Allocation Data Structures.	05
2	**Assembler and Macro Processor** Assemblers - Elements of Assembly Language Programming, Assembly Scheme, single pass Assembler, two pass assembler, Macro Processors - Macro Definition and Call, Macro Expansion, Advanced Macro Facilities.	06
3	**Loader, Linker, Compiler, and Interpreter** Loader and Linkers - Relocation of Linking Concept, Design of Linker, Linking for overlays, Different Loading Schemes, Sequential and Direct Loaders Introduction to Compiler - Phases and Use of Compiler, Introduction to Interpreter, YACC, LEX.	06

4	**Scanning:** The scanning Process, Regular Expressions, Finite Automata, NFA, Kleene's Theorem, Conversion from NFA to FA, ɜ - Non-Deterministic Finite Automata, Conversion of NFA-ɜ. to NFA, DFA.	**07**
5	**Parsing:** Parsing Process, Chomsky hierarchy, CFG, CFL, PDA and Turing Machine, Parse Trees and Abstract Syntax Trees, Ambiguity, Elimination of Left Recursion, Left Factoring, Top-Down Parsing, First and Follow Sets, Bottom-up Parsing.	**10**
6	**Runtime Environments:** Activation Records, Intermediate code generation. Local optimization, Data flow analyses: constant propagation, liveness analysis, common sub-expression elimination.	**06**
	Total	**40**

List of Practicals / Tutorials:

1	**Study of Lex Tool**: 1. Write a lex program to identify numbers, words and other characters and generate tokens for each. 2. Write a lex program to identify all the lexemes from the input file that follow the given RE. Provide the RE as a command line argument.
2	**Study of Yacc Tool:** 1. Write a Yacc program for desktop calculators with ambiguous grammar. 2. Write a Yacc program for calculators with unambiguous grammar.
3	Construct a DFA for following regular expression. (a | b)*aab# (a*|b*)a*ab# (1*)*0(0/1)*#
4	Construct NFA for the following regular expressions: • abba • bb(a)* • (a|b)* • a* | b* • a(a)*ab • aa*+ bb* • (a+b)*abb • 10(0+1)*1 • (a+b)*a(a+b) • (0+1)*010(0+1)* • (010+00)*(10)* • 100(1)*00(0+1)*
5	Convert following regular expression to DFA using subset construction method: (a+b)*a(a+b) (a+b)*ab*a

6	Check whether following grammars are ambiguous or not: S → aS | Sa | ε (output string: aaaa) S → aSbS | bSaS | ε (output string: abab) S → SS+ | SS* | a (output string: aa+a*) <exp> → <exp> + <term> | <term> <term> → <term> * <letter> | <letter> <letter> → a|b|c|...|z (output string: a+b*c) Prove that the CFG with productions: S→ a | Sa | bSS | SSb | SbS is ambiguous

7	Remove left recursion from following grammar:
	• A → Abd | Aa | a
	  ○ B → Be | b
	• A → AB | AC | a | b
	• S → A | B
	  ○ A → ABC | Acd | a | aa
	  ○ B → Bee | b
	• Exp → Exp+term | Exp-term | term
	• S → A
	A → Ad | Ae | aB | aC
	B → bBC | f
	C → g

8	Given a Grammar: E → TA, A → +TA | ε T → VB B → *VB | ε V → id | (E) Develop an LL (1) parser table and parse following string using the parsing table. id * (id + id) Consider the grammar: S → AB | ABad A → d E → b D → b | ε B → c Construct the predictive parsing table. Show that the given grammar is LL(1) or not.

9	Find first and follow for the given Production Rules: • S → aBDh B → cC C → bC | ε D → EF E → g | ε F → f | ε • S → 1AB | ε A → 1AC | 0C B → 0S C → 1 • S → iCtSeS | iCtS | a C → b
10	Construct Predictive Parsing Table for the following grammar: • E → BA A → &BA | ε B → true | false • E → TA A → +TA | ε T → VB B → *VB | ε V → id | (E) • S → iEtSS'|a S' → eS| ε E → b
11	Given the grammar, evaluate the string id – id * id using shift reduce parser: • E → E – E E → E * E E → id
12	Given the grammar, perform the top-down parsing for the string +*35*45. • E → +TE | E • T → *VT | V • V → 0|1|2|3|...|9

Reference Books:

1	System Programming by D M Dhamdhere, McGraw Hill Publication
2	System Programming by Srimanta Pal, OXFORD Publication
3	Compilers: Principles, Techniques and Tools by Alfred Aho, Ravi Sethi, Jeffrey Ullman
4	Compiler Construction: Principles and Practice by Kenneth C Louden
5	Introduction to Languages and the Theory of Computation by John C Martin

Supplementary learning material:	
1	NPTEL – Swayam Course on Theory of Computation, IIT Kanpur: https://nptel.ac.in/courses/106104148

2	NPTEL – Swayam Course on Theory of Computation, IIT Kanpur: https://nptel.ac.in/courses/106104148

Pedagogy:

- Direct classroom teaching
- Audio Visual presentations/demonstrations
- Assignments/Quiz
- Continuous assessment
- Interactive methods
- Seminar/Poster Presentation
- Industrial/ Field visits
- Course Projects

Suggested Specification table with Marks (Theory) (Revised Bloom's Taxonomy):

Distribution of Theory Marks in %						**R**: Remembering; **U**: Understanding; **A**: Applying; **N**: Analyzing; **E**: Evaluating; **C**: Creating
R	**U**	**A**	**N**	**E**	**C**	
15%	25%	15%	25%	20%	---	

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcomes (CO):

Sr.	Course Outcome Statements	%weightage
CO-1	To understand the working of Language Processors.	**15**
CO-2	Explain and classify different methodologies, concepts, and approaches to System Programming	**25**
CO-3	Ability to apply automata theory and knowledge on formal languages.	**25**
CO-4	Ability to identify and select suitable parsing strategies for a compiler for various cases.	**20**
CO-5	Build various system programs using language processor development tools such as YACC and Lex	**15**

Curriculum Revision:	
Version:	1.0
Drafted on (Month-Year):	June-2020
Last Reviewed on (Month-Year):	-
Next Review on (Month-Year):	June-2025