**B.Tech. in Computer Science &Engineering**

**Semester: VII, Academic Year: 2022-23, Term: Odd**

|  |  |
| --- | --- |
| **Course Code & Name** | 2CS701 – Compiler Construction |

**Lesson Plan**

|  |  |  |
| --- | --- | --- |
| **Lecture No.** | **Topic** | **Mapped CLO** |
|  | **Introduction** |  |
| 1 | Course overview and significance  Translator , Compilers, Interpreter | 1 |
| 2 | Analysis-Synthesis Compiler Model | 1 |
| 3 | Single Pass vs Multi-pass  Cousins of Compiler  Types of compiler and applications  Symbol Table | 1 |
|  | Lexical Analysis |  |
| 4 | Role of lexical analyzer  Token, Lexeme  Use of Regular Expression for token  Look-ahead operator | 1 |
| 5 | Input Buffering, and Significance Sentinel | 1 |
| 6 | Finite Automata,  Optimization of DFA based Pattern Matching  Lexical analyzer generator | 1,2 |
| 7 | Regular expression to DFA | 1,2 |
|  | **Syntax Analysis** |  |
| 8 | Role of Parser  Introduction to Error recovery  Context Free Grammar  Comparison of CFG and Regular Expressions | 1 |
| 9 | Context Free Grammar for programming constructs | 1,2 |
| 10 | Introduction to Top Down Parsing  LL(1) Parsing | 1,2 |
| 11 | Recursive-Descent Parser | 1,2 |
| 12 | Need of Left Recursion elimination  Left Recursion elimination : Direct, Indirect | 1 |
| 13 | Predictive parse table generation  LL(1), LL(k) Grammar | 1,2 |
| 14 | Error recovery strategies  Error recovery at LL(1) parsers | 1,2 |
| 15 | Bottom-up Parsing  Shift Reduce Parsing  Shift/Reduce conflict, Reduce/Reduce conflict | 1 |
| 16 | LR Parsing Model | 1 |
| 17 | LR(0) Parse table Generation | 1,2 |
| 18 | SLR Parse table Generation | 1,2,3 |
| 19 | LR(1) Parse table Generation | 1,2 |
| 20 | LALR(1) Parse table Generation  Error recovery at LR parsers | 1,2 |
| 21 | Comparison of LR Parsers and LL Parsers  Dealing with ambiguous grammar  Parsing Generator Tools | 1,2 |
| 22 | Operator Precedence Parsing  Error recovery at operator precedence parsing | 1,2 |
|  | Syntax Directed Translation |  |
| 23 | Syntax Directed Definition (SDD)  Annotated Tree  S-attributed and L-attributed SDD | 1 |
| 24 | Type checking SDD | 1,2,4 |
| 25 | Bottom – Up Evaluation of S – Attributed Definitions | 1,2 |
| 26 | Bottom – Up Evaluation of L – Attributed Definitions | 1,2 |
| 27 | Translation scheme | 1 |
| 28 | Top – Down Translation | 1,2 |
| 29 | Recursive evaluators | 1,2 |
|  | Run-time Environments |  |
| 30 | Static Memory Allocation  Stack Memory Allocation | 1 |
| 31 | Symbol Table Management | 1 |
|  | Intermediate code generation |  |
| 32 | Intermediate code representations | 1 |
| 33 | Types of three address statements | 1 |
| 34 | Three address code for declaration, assignment statements, Boolean expressions | 1,2 |
| 35 | Three address code for control(if..else) construct | 1,2 |
| 36 | Three address code for Loop construct | 1,2 |
| 37 | Back-patching | 1,2 |
|  | Code Generation |  |
| 38 | Functionalities of Code Generation phase  Issues in design of a code generation | 1 |
| 39 | Basic code generation | 1,2 |
| 40 | Basic blocks and flow graph, Control flow | 1 |
| 41 | A simple code generator | 1,2 |
| 42 | Register allocation and assignment | 1,3 |
|  | Code Optimization |  |
| 43 | Machine independent optimization techniques | 3 |
| 44 | Local and Global Code optimization | 3 |
| 45 | Peephole Optimization | 3 |