| | ITER, SIKSHA 'O' ANUSANDHAN<br>Deemed to be University | | LESSON PLAN |
|---|---|---|---|
| Programme | **B.Tech.** | Academic Year | **2023-24** |
| Department | **CSE/CSIT** | Semester | **5$^{th}$** |
| Instructor | **Dr. Lambodar Jena** | Grading Pattern | **6** |
| Subject Code | **CSE 3731** | | |
| Subject Name | **Introduction to the Theory of Computation** | | |

## Text Books(s):

(1) Introduction to the Theory of Computation, by Michael Sipser, Cengage learning.

**Course Format:** 3 Classes/week, 1 hr/Class; 3 Credits

| | | |
|---|---|---|
| **Course Outcomes** | By the end of the course, through lectures, readings, home works, assignments, and exams, students will be able to: | |
| | **CO1** | Enhance/develop ability to understand and conduct mathematical proofs for computation and algorithms. |
| | **CO2** | Design and analyze finite automata and regular expression for describing regular languages. |
| | **CO3** | Design and analyze pushdown automata, and context-free grammars. |
| | **CO4** | Design and analyze Turing machines. |
| | **CO5** | Enhance the ability to understand the decidability, undecidability, and reducibility criteria of various computational problems. |
| | **CO6** | Demonstrate the understanding of key notions, such as algorithm, computability and complexity through problem solving. |

| **Lecture** | **Lessons/Topics to be covered** | **Book reference (sections)** | **Mapping with COs** |
|---|---|---|---|
| ***Week #1:*** | | | |
| **Lecture#1** | Introduce the grading pattern, credit, classes, and problem-solving session of the course. Motivation behind the course. Introduction to automata, computability and complexity theory. | Sipser (pg.1-3) | **CO1** |
| **Lecture#2** | Mathematical Notions and Terminology: Set, Sequences and tuples, Functions and Relations, Graphs, Strings and Languages, Boolean logic. | Sipser (pg.3-16) | **CO1** |

| | | | |
|---|---|---|---|
| **Lecture#3** | Definitions, Theorems, and proofs: Finding Proofs Types of Proof: Proof by construction, proof by contradiction, proof by induction. Specifically, proof $\sqrt{2}$ is an irrational number in different ways (i.e. by contradiction, by geometry etc.) | Sipser (pg.17-24) | **CO1** |
| ***Week #2:*** | | | |
| **Lecture#4** | Regular Languages: Finite Automata: Formal definition of a finite automaton, Examples of finite automata, formal definition of computation. | Sipser (pg.31-37) | **CO2** |
| **Lecture#5** | Designing finite automata. | Sipser (pg.37-41) | **CO2** |
| **Lecture#6** | Designing finite automata contd... | Sipser (pg.37-41) | **CO2** |
| ***Week #3:*** | | | |
| **Lecture#7** | Regular operations (union, concatenation, star). The class of regular languages is closed under the union operation, and concatenation operation. | Sipser (pg.44-47) | **CO2** |
| **Lecture#8** | Nondeterminism: Formal definition of a nondeterministic finite automaton, NFA examples and sample design. | Sipser (pg.47-53) | **CO2** |
| **Lecture#9** | Equivalence of NFAs and DFAs. | Sipser (pg.54) | **CO2** |
| ***Week #4:*** | | | |
| **Lecture#10** | NFA and regular operations and introduction to regular expressions. | Sipser (pg.63-66) | **CO2** |
| **Lecture#11** | Equivalence of regular expression and finite automata. | Sipser (pg.66) | **CO2** |
| **Lecture#12** | Equivalence of regular expression and finite automata contd... | Sipser (pg.66) | **CO2** |
| ***Week #5:*** | | | |
| **Lecture#13** | Non-regular languages: The pumping lemma for regular languages, proof, pigeonhole principle. | Sipser (pg.77) | **CO2** |
| **Lecture#14** | Examples on pumping lemma. | Sipser (pg.77) | **CO1,CO2** |
| **Lecture#15** | Examples on pumping lemma contd... More discussion on closure properties of regular sets. | Sipser (pg.77) | **CO1,CO2** |
| ***Week #6:*** | | | |

| | | | |
|---|---|---|---|
| **Lecture#16** | Context-Free Languages: Context-Free Grammars: Formal Definition of a context-free grammars, Examples of context-free grammars, Designing context-free grammars. | Sipser (pg.101-106) | **CO3** |
| **Lecture#17** | Ambiguity, and Chomsky normal form. | Sipser (pg.107-108) | **CO3** |
| **Lecture#18** | Introduction to pushdown automata (PDA): formal definition of a push-down automata, Examples on pushdown automata. | Sipser (pg.111-113) | **CO3** |
| ***Week #7:*** | | | |
| **Lecture#19** | Pushdown automata and Equivalence with context-free languages. | Sipser (pg.117) | **CO3** |
| **Lecture#20** | Non-context-free languages: The pumping lemma for context-free languages. | Sipser (pg.125) | **CO3** |
| **Lecture#21** | The pumping lemma for context-free languages contd... | Sipser (pg.125) | **CO3** |
| ***Week #8:*** | | | |
| **Lecture#22** | Deterministic context-free languages (DCFL): Properties of DCFLs. | Sipser (pg.130-133) | **CO3** |
| **Lecture#23** | Deterministic context-free grammars, Relationship of Deterministic PDAs and DCFGs. | Sipser (pg.135-146) | **CO3** |
| **Lecture#24** | Deterministic context-free grammars, Relationship of DPDAs and DCFGs, Parsing and LR(k) grammars. | Sipser (pg.146-151) | **CO3** |
| ***Week #9:*** | | | |
| **Lecture#25** | Computability Theory: Turing Machines: Formal definition of a Turing machine, Examples of Turing machine | Sipser (pg.165-170) | **CO4** |
| **Lecture#26** | Examples of Turing machine contd... | Sipser (pg.167-170) | **CO4** |

| | | | |
|---|---|---|---|
| **Lecture#27** | Variants of Turing machines: Multitape Turing machines, Nondeterministic Turing machine. | Sipser (pg.176-178) | **CO4** |
| ***Week #10:*** | | | |
| **Lecture#28** | Enumerators, equivalence with other models. | Sipser (pg.180-181) | **CO4** |
| **Lecture#29** | Variants of Turing machines contd... | Sipser (pg.176-181) | **CO4** |
| **Lecture#30** | The Definition of Algorithm: Hilbert's Problem | Sipser (pg.182) | **CO6** |
| ***Week #11:*** | | | |
| **Lecture#31** | Decidability: Decidable Languages, decidable problems concerning regular languages. | Sipser (pg.193-194) | **CO5** |
| **Lecture#32** | Decidable problems concerning context-free languages. | Sipser (pg.194-198) | **CO5** |
| **Lecture#33** | Decidable problems contd... | Sipser (pg.194-198) | **CO5** |
| ***Week #12:*** | | | |
| **Lecture#34** | Undecidability: The diagonalization method. | Sipser (pg.201-202) | **CO5** |
| **Lecture#35** | An undecidable language. | Sipser (pg.207) | **CO5** |
| **Lecture#36** | A Turing-unrecognizable language. | Sipser (pg.207-209) | **CO5** |
| ***Week #13:*** | | | |
| **Lecture#37** | Reducibility: undecidable problems from language theory. | Sipser (pg.215-216) | **CO5** |

| | | | |
|---|---|---|---|
| **Lecture#38** | Reduction via computation histories. | Sipser (pg.216-220) | **CO5** |
| **Lecture#39** | Reduction via computation histories contd... | Sipser (pg.216-220) | **CO5** |
| ***Week #14:*** | | | |
| **Lecture#40** | Mapping reducibility. | Sipser (pg.234) | **CO5** |
| **Lecture#41** | Computable functions. | Sipser (pg.234) | **CO5** |
| **Lecture#42** | Formal definition of mapping reducibility. | Sipser (pg.235) | **CO5** |
| ***Week #15:*** | | | |
| **Lecture#43** | Problems and Discussions related to decidability. | Sipser (pg.210) | **CO5, CO6** |
| **Lecture#44** | Problems and Discussions on undecidability' | Sipser (pg.210) | **CO5, CO6** |
| **Lecture#45** | Problems and discussions on reducibility. | Sipser (pg.239) | **CO5, CO6** |