	Theory of Computation: A Problem-Solving Approach, Kavi Mahesh, Wiley India, 2012	
		Learning Objectives (Refer to respective chapter numbers)
1	L1.1	Learn to distinguish between computing machines and other kinds of machines
2	L1.2	Learn what it means to compute
3	L1.3	Learn the relationships between computing machines, problems, languages and grammars
4	L1.4	Learn to define computer science
5	L1.5	Appreciate the relationships between the science of computing and a core set of subject areas in computer science
6	L1.6	Appreciate the historical development of the science of computing
7	L1.7	Get an introduction to computability, intractability and intelligence
8	L1.8	Understand the reasons for studying the science of computing
9	L2.1	Recognize that everyday machines such as vending machines are computing devices
10	L2.2	Learn to design a finite automaton for a given problem
11	L2.3	Learn to handle end conditions in automata
12	L2.4	Learn to handle reject states in automata
13	L2.5	Learn to use states as memory
14	L2.6	Use a step-by-step method for constructing complex automata
15	L2.7	Appreciate some limitations of deterministic finite automata
16	L3.1	Learn to use nondeterminism as a tool in designing automata
17	L3.2	Learn how to design nondeterministic automata for a given problem
18	L3.3	Learn to convert a nondeterministic automaton to a deterministic one through subset construction
19	L3.4	Learn the use of λ -transitions in designing nondeterministic automata
20	L3.5	Learn to minimize the states in an automaton
21	L3.6	Understand how finite state transducers work
22	L4.1	Learn what is a formal language and how it is related to an automaton
23	L4.2	Learn to construct simple regular expressions
24	L4.3	Learn to convert a regular expression to an equivalent automaton
25	L4.4	Learn to convert an automaton to a regular expression
26	L4.5	Learn a method to compare two regular expressions or finite automata to determine their equivalence
27	L4.6	Learn to construct regular expressions for user data validation
28	L5.1	Learn how grammars can be more compact than regular expressions
29	L5.2	Learn how grammars are used to parse or derive strings in a language
30	L5.3	Learn how to construct grammars for regular languages
31	L5.4	Learn to convert automata to regular grammars
32	L5.5	Learn to convert regular grammars to automata
33	L6.1	Learn to combine regular languages using their closure properties to obtain more complex regular languages

34	L6.2	Learn to answer questions such as emptiness and finiteness of regular languages
35	L6.3	Learn why some languages are not regular
36	L6.4	Learn why regular languages are called regular
37	L6.5	Learn to apply the Pumping Lemma in an adversarial game to show that some languages are not regular
38	L7.1	Learn what is context-free behavior
39	L7.2	Learn how context-free grammars can derive languages that are not regular
40	L7.3	Learn to construct linear context-free grammars
41	L7.4	Learn to construct non-linear context-free grammars
42	L7.5	Learn to construct leftmost and rightmost derivation trees for strings
43	L7.6	Learn the relationship between parsing and ambiguity in grammars
44	L7.7	Learn to eliminate ambiguity in grammars
45	L7.8	Learn to convert a context-free grammar to Chomsky Normal Form
46	L7.9	Learn to convert a context-free grammar to Greibach Normal Form
47	L7.10	Learn to parse a string using the CYK algorithm
48	L8.1	Learn how to add memory to a finite automaton
49	L8.2	Learn the equivalence of stack memory and context-free behavior
50	L8.3	Learn to construct a pushdown automaton for a context-free language
51	L8.4	Learn to convert a context-free grammar to a pushdown automaton
52	L8.5	Learn to convert a pushdown automaton to a context-free grammar
53	L8.6	Understand why nondeterminism is required in pushdown automata
54	L9.1	Learn how to combine context-free languages to obtain more complex languages
55	L9.2	Learn why context-free languages are not closed under intersection or complementation
56	L9.3	Learn to answer questions such as emptiness and finiteness of context-free languages
57	L9.4	Learn that equivalence or ambiguity questions of context-free languages cannot be answered!
58	L9.5	Learn why some languages are not context-free
59	L9.6	Learn to apply the Pumping Lemma to show that a language is not context-free
60	L10.1	Learn how a simple computing machine can computing anything that is computable
61	L10.2	Learn to construct simple Turing Machines for languages and computable functions
62	L10.3	Learn techniques for constructing more complex Turing Machines
63	L10.4	Understand how variations of Turing Machines are all equivalent
64	L10.5	Learn how to compile a Turing Machine into a binary string
65	L10.6	Understand how a Universal Turing Machines works as a stored-program computer
66	L11.1	Learn the difference between countably infinite and uncountably infinite sets

67	L11.2	Learn the diagonalization technique to show that a language is uncountable
68	L11.3	Learn how to properly enumerate a language
69	L11.4	Learn why some languages are not enumerable
70	L11.5	Understand the difference between acceptance and membership in a language
71	L11.6	Understand why some languages are enumerable but not recursive
72	L11.7	Learn to construct a context-sensitive grammar for a language
73	L11.8	Learn to construct a pushdown automaton with two stacks
74	L11.9	Learn how unrestricted grammars work
75	L11.10	Learn how linear-bounded automata work
76	L11.11	Learn how all formal languages can be arranged in the Chomsky Hierarchy
77	L12.1	Learn to solve simple cases of the Post Correspondence Problem
78	L12.2	Learn why PCP is unsolvable
79	L12.3	Learn the difference between decidability and computability
80	L12.4	Learn why some problems are not computable
81	L12.5	Understand why Turing Machines suffer from the Halting Problem
82	L12.6	Appreciate kinds of problems that are undecidable
83	L12.7	Get an overview of the field of computational complexity
84	L12.8	Understand the difference between recognition and recall in computing solutions
85	L12.9	Wonder about a number of open questions in the science of computing