SIES Graduate School of Technology (Autonomous) NAAC A+ RISE WITH EDUCATION (Affiliated to University of Mumbai)	***End Semester Examination (R-24/R-25) SH 2026 — Answer Key with Marking Scheme***
Branch: CSE(IOT&CSIBCT)	Course: Discrete Structure and Automata Theory
Year/ Semester: SE/III	Course Code: CSEC303
Time: 03 hours	Marks: 80

		Marks
Q. 1	**All questions are compulsory. Award marks according to the points indicated below.**	
A.	• Reflexive: Verify that every element is related to itself. Since (1,1), (2,2), and (3,3) are present in R, the relation is reflexive — 1.5 marks • Symmetric: For each ordered pair (a,b) in R, check that (b,a) also belongs to R. In particular, both (1,2) and (2,1) are present — 1 mark • Transitive: For every (a,b) and (b,c) in R, verify that (a,c) is also in R. All possible compositions satisfy this condition — 1.5 marks • Conclusion: As R satisfies reflexivity, symmetry, and transitivity, R is an equivalence relation — 1 mark	05
B.	• Definition: State that an equivalence relation is a relation that is reflexive, symmetric, and transitive — 1 mark • Advantages: Explain any four relevant advantages, such as partitioning a set into disjoint equivalence classes, grouping objects with the same properties, simplifying classification, and supporting quotient-set construction — 4 marks	05
C.	• States: Clearly define states that represent the progress made toward detecting the suffix 01, and identify the initial and accepting states — 1 mark • Transitions: Provide correct transitions on inputs 0 and 1 so that all strings whose last two symbols are 01 reach an accepting state — 2 marks • Diagram: Draw a complete and properly labelled NFA, including the start arrow and double circle for the final state — 1 mark • Verification: Explain why a string is accepted exactly when it ends in 01, and illustrate with one accepted and one rejected string — 1 mark	05
D.	• Definition: An NFA is a finite automaton in which a state may have zero, one, or several possible next states for an input symbol — 1 mark • Formal representation: Write the five-tuple $M = (Q, \Sigma, \delta, q_0, F)$ and define every component correctly — 2 marks • Transition function: Explain that δ maps a state and input symbol to a set of possible states, and describe how an input is processed nondeterministically — 1 mark • Example: Give a suitable labelled NFA and briefly trace at least one	05

	accepted input string — 1 mark	
E.	• Statement: State the Pumping Lemma for regular languages using a pumping length p and a decomposition w = xyz — 1 mark • Conditions: Specify $\|xy\| \leq p$, $\|y\| > 0$, and $xy^iz \in L$ for every $i \geq 0$ — 1 mark • String selection: Choose an appropriate string in L of length at least p, for example a^pb^{p+1} — 1 mark • Contradiction: Since y lies within the initial block of a symbols, pumping y changes the number of a symbols without making the corresponding required change in b symbols; the pumped string therefore leaves L — 1.5 marks • Conclusion: The contradiction shows that $L = \{a^nb^{n+1} \mid n > 1\}$ is not regular — 0.5 mark	05
F.	• Hierarchy diagram: Draw the nested inclusion Regular ⊂ Context-Free ⊂ Context-Sensitive ⊂ Recursively Enumerable — 1.5 marks • Type 0: Describe unrestricted grammar, its production form, equivalent Turing-machine model, and one valid example language — 0.75 mark • Type 1: Describe context-sensitive grammar, its non-contracting productions, equivalent linear-bounded automaton, and one example — 0.75 mark • Type 2: Describe context-free grammar with a single nonterminal on the left, equivalent pushdown automaton, and one example — 0.75 mark • Type 3: Describe regular grammar, equivalent finite automaton, and one example of a regular language — 0.75 mark • Inclusion: State the proper containment relationship among all four language classes — 0.5 mark	05
Q. 2A	• State identification: Use states q_0, q_1, q_2, and q_3 to represent the longest suffix of the processed input that is also a prefix of 101: ε, 1, 10, and 101 respectively — 3 marks • Transition design: Give a complete transition table or diagram on inputs 0 and 1, including fallback transitions that retain the longest useful suffix — 3 marks • Output assignment: Assign output a to the state reached when the processed input ends in 101; assign output b to all other states — 2 marks • Verification: Trace suitable strings such as 101, 1101, and 100, showing that only strings ending in 101 produce output a — 2 marks	10
	OR	
Q. 2B	• State identification: Define states that record matched prefixes of the	10

	target substring 1101, from no match through the complete match — 3 marks • Transition design: Draw a complete state-transition diagram or table, including overlap and fallback transitions for both binary inputs — 3 marks • Output assignment: Produce the occurrence-detection output when the machine reaches the state corresponding to the complete substring 1101; otherwise produce the non-detection output — 2 marks • Verification: Trace at least one string containing 1101 and one string not containing it, and show the output generated at each relevant step — 2 marks.	
Q. 3A	• Interpretation: (0+1)*000 represents all binary strings ending with the suffix 000 — 1 mark • NFA construction: Use an initial state with loops on 0 and 1, then nondeterministically start a three-zero path leading to the accepting state — 3 marks • States: Mark the looping state as initial and the state reached after reading the third consecutive 0 as final — 1 mark • Subset construction: Compute ε-closures if needed, generate each reachable subset, and record transitions on 0 and 1 — 3 marks • DFA result: Present the complete DFA transition table or diagram, clearly identifying the initial subset and every accepting subset — 2 marks	10
	OR	
Q. 3B	• Statement: State Arden's Theorem: if R = Q + RP and ε is not in P, then the unique solution is R = QP*, with the corresponding left-sided form where applicable — 2 marks • Proof: Substitute QP* into the equation and use P* = ε + PP*; then establish uniqueness by repeated substitution under the stated condition — 4 marks • Application: Rewrite R = aR + b in the left-recursive Arden form and identify the coefficient a and constant b — 2 marks • Final expression: By Arden's Theorem, obtain R = a*b, and briefly verify that it satisfies R = aR + b — 2 marks	10
Q. 4A	• Procedure: List the standard CNF conversion sequence—introduce a new start symbol if required, remove ε-productions, remove unit productions, remove useless symbols, and convert remaining rules — 2 marks • ε-productions: Identify nullable variables and add all valid alternatives obtained by omitting nullable variables, while preserving ε only when the	10

	language requires it — 2 marks • Unit productions: Replace each rule of the form A → B by the non-unit productions derivable from B, and then delete the unit rules — 2 marks • CNF form: Replace terminals appearing in long right-hand sides with new variables, and split every production of length greater than two into binary productions — 2 marks • Final grammar: Present an equivalent grammar containing only productions A → BC or A → a, with the permitted start-symbol ε case if applicable — 2 marks	
	OR	
Q. 4B	• ε-elimination: Identify every nullable variable, generate the necessary alternative productions, and remove ε-productions without changing the language except for the permitted start case — 3 marks • Unit elimination: Compute unit-pair reachability and replace each unit production with the corresponding non-unit productions — 2 marks • Useless-symbol removal: Delete non-generating variables first and then remove symbols that are unreachable from the start symbol — 2 marks • Simplified grammar: Write the final equivalent grammar clearly and verify that no ε-productions, unit productions, or useless symbols remain — 3 marks	10
Q. 5A	• Language and strategy: Identify the required relationship between the blocks and explain how the stack records the count contributed by the first block — 2 marks • PDA design: Draw clearly labelled states for reading the first block, processing the second block, and accepting; include all necessary transitions — 3 marks • Stack operation: For every 0, push three stack markers; for every subsequent 1, pop one marker. Reject if the symbols occur out of order or the stack cannot be matched — 3 marks • Acceptance: Mark the start and final states and accept only after the complete input has been read and the stack returns to its bottom marker — 2 marks	10
	OR	
Q. 5B	• Language pattern: Identify which two symbol counts must be matched and explain the role of any unrestricted middle portion — 2 marks • PDA design: Provide states and transitions for the initial block, intermediate portion, matching block, and final acceptance — 3 marks • Stack operation: Push one marker for each 0 and pop one marker for each c, while allowing the permitted middle symbols without disturbing the	10

	stored count — 3 marks • Boundary cases: Handle m,n ≥ 0 explicitly, including empty permitted blocks, and accept only after the entire input is consumed with the required stack condition satisfied — 2 marks	
Q. 6A	• Language pattern: Recognize that the machine must compare the symbols on the two sides of separator C in the required order — 1 mark • Marking strategy: Repeatedly mark the leftmost unprocessed symbol, move across C, and locate and mark its corresponding symbol on the right — 3 marks • Transitions: Provide separate transition sequences for matching a and b, returning the head to the next unmarked symbol after each successful match — 3 marks • Separator and rejection: Treat C as the midpoint; reject on a wrong symbol, missing partner, extra unmarked symbol, or malformed input — 2 marks • Acceptance: Enter the final state only when all symbols on both sides have been matched and no unmatched input remains — 1 mark	10
	OR	
Q. 6B	• Language pattern: Identify that L = $\{a^n b^n c^n \mid n \geq 0\}$ requires equal numbers of a, b, and c in that order — 1 mark • Marking pass: In each cycle, mark one unprocessed a, then one corresponding b, and then one corresponding c using distinct tape symbols — 3 marks • Head movement: Specify transitions that move right to find b and c, then return left to the next unmarked a for the following pass — 3 marks • Validation: Reject if the order is incorrect, a matching symbol is absent, or any unmarked b or c remains after all a symbols are processed — 2 marks • Acceptance: Accept when all three blocks are completely and equally marked; also include the empty-string case n = 0 — 1 mark	10
	********* **All the Best**********|	