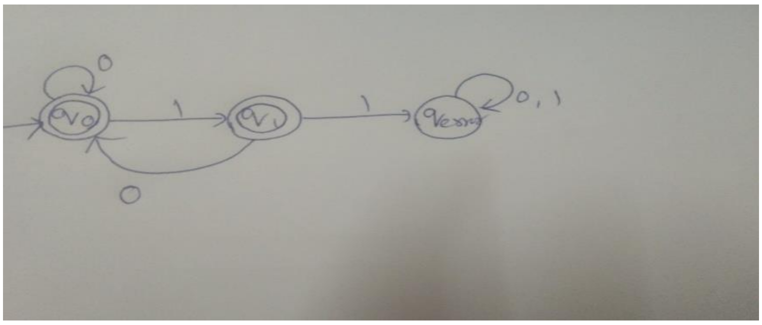
USN

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

# Internal Assessment Test 1 – Oct 2025

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Sub: | Theory of Computation | | | | | Sub Code: | BCS503 | Branch: | CSE |
| Date: | 01.10.2025 | Duration: | 90 mins | Max Marks: | 50 | Sem/Sec: | 5 A,B,C | | OBE |

| Answer any FIVE FULL Questions | | MARKS | CO | RBT |
|---|---|---|---|---|
| 1 (a) | Design a DFA for the following languages over Σ ={0,1}<br>L={w\| w is a string that doesn't contain consecutive 1's}. Show the computation for **w = 0110** and state whether it is an accepting or rejecting configuration using extended transition function. | **5M** | **CO1** | **L3** |



Transition table is given below for the above diagram.

| **State/ Input** | **0** | **1** |
|---|---|---|
| → *q0 | q0 | q1 |
| *q1 | q0 | Ø |

Steps for computation of **w = 0110**

$\hat{\delta}$ **(q0,Ɛ) = q0**

$\hat{\delta}$ **(q0,Ɛ0) = δ(**$\hat{\delta}$ **(q0,Ɛ) ,0)=δ(q0,0)=q0**

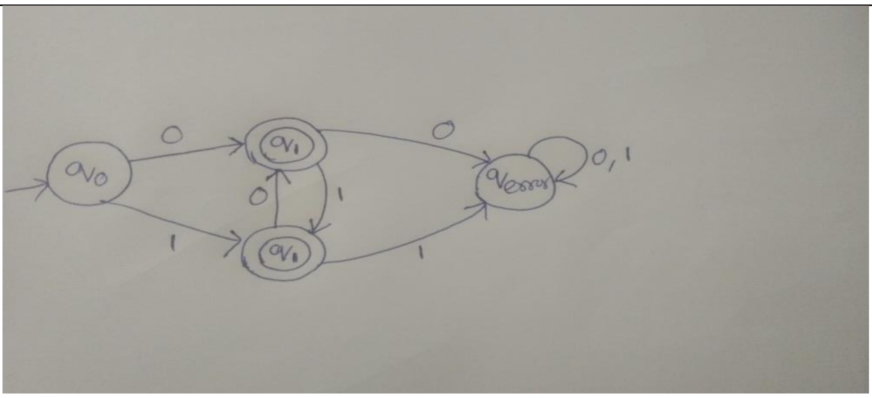
$\hat{\delta}$ **(q0,01) = δ(**$\hat{\delta}$ **(q0,0) ,1)=δ(q0,1)=q1**

$\hat{\delta}$ **(q0,011) = δ(**$\hat{\delta}$ **(q0,01) ,1)=δ(q1,1)=qerror**

$\hat{\delta}$ **(q0,0110) = δ(**$\hat{\delta}$ **(q0,011) ,0)=δ(qerror,1)=**$\mathbf{q_{error}}$

**It is not accepted.**

| | | | | |
|---|---|---|---|---|
| (b) | L={w\| No two consecutive characters are same in w}. Show the computation for **w = 1010** and state whether it is an accepting or rejecting configuration using extended transition function. | **5M** | **CO1** | L3 |



Transition table is given below for the above diagram.

| **State/ Input** | **0** | **1** |
|---|---|---|
| → q0 | q1 | q2 |
| *q1 | Ø | q2 |
| *q2 | q1 | Ø |

Steps for computation of **w = 1010**

**$\hat{\delta}$ (q0,Ɛ) = q0**

**$\hat{\delta}$ (q0,Ɛ1) = δ($\hat{\delta}$ (q0,Ɛ) ,1)=δ(q0,1)=q2**

**$\hat{\delta}$ (q0,10) = δ($\hat{\delta}$ (q0,1) ,0)=δ(q2,0)=q1**

**$\hat{\delta}$ (q0,101) = δ($\hat{\delta}$ (q0,10) ,1)=δ(q1,1)=q2**

**$\hat{\delta}$ (q0,1010) = δ($\hat{\delta}$ (q0,101) ,0)=δ(q2,0)=q1**

**It is accepted.**

2 (a)

Define NFA. Design a NFA over Σ={a,b} which accepts the strings containing substring **abba or bbaa.**

An NFA, or Non-Deterministic Finite Automaton, is a type of computational model where, for a given input, there can be multiple possible next states, including zero, one, or more.

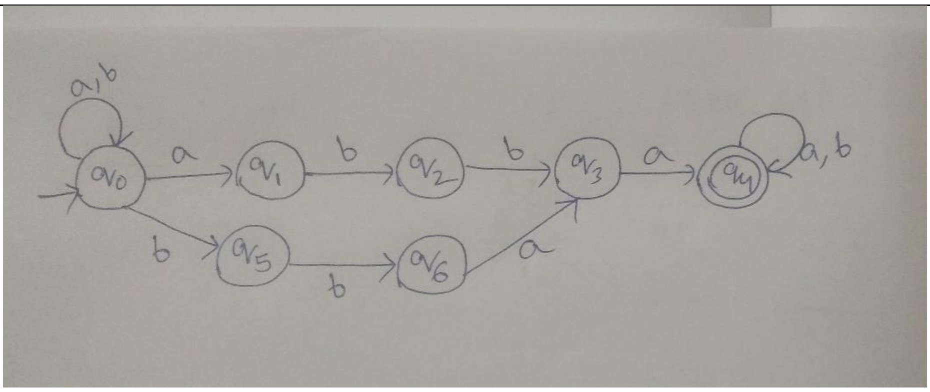
An NFA can be represented by a 5-tuple (Q, ∑, δ, $q_0$, F) where −

- **Q** is a finite set of states.
- ∑ is a finite set of symbols called the alphabets.
- **δ** is the transition function where δ: Q × ∑ → $2^Q$

(Here the power set of Q ($2^Q$) has been taken because in case of NFA, from a state, transition can occur to any combination of Q states)

- **$q_0$** is the initial state from where any input is processed ($q_0$ ∈ Q).
- **F** is a set of final state/states of Q (F ⊆ Q).

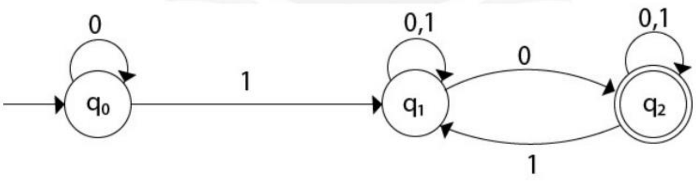
**5M** **CO1** L1



Transition table is given below for the above diagram.

| State/ Input | a | b |
|---|---|---|
| → q0 | {q0,q1} | {q0,q5} |
| q1 | Ø | q2 |
| q2 | Ø | q3 |
| q3 | q4 | Ø |
| q4 | q4 | q4 |
| q5 | Ø | q6 |
| q6 | q3 | Ø |

(b) Convert the following NFA to DFA.



DFA Transition Table

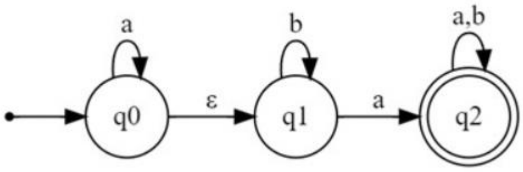
| State/ Input | 0 | 1 |
|---|---|---|
| → q0 | q0 | q1 |
| q1 | {q1, q2} | q1 |
| *{q1,q2} | {q1, q2} | {q1, q2} |

δ ({q1, q2},0) = δ (q1,0) U δ (q2,0) = {q1, q2}U{q2}={q1, q2}

δ ({q1, q2},1) = δ (q1,1) U δ (q2,1) = {q1}U{q1,q2}={q1, q2}

**5M** **CO1** L3

3 (a) Find the ECLOSE of each state for the given Ɛ-NFA. Convert the following Ɛ-NFA to equivalent DFA.



**7M** **CO1** L1

ECLOSE (q0)= {q0, q1}

ECLOSE (q1)= {q1}

ECLOSE (q2)= {q2}

DFA Transition Table

| State/ Input | a | b |
|---|---|---|
| →{q0, q1} | {q0, q1,q2} | {q1} |
| *{q0, q1,q2} | {q0, q1,q2} | {q1,q2} |
| *{q1,q2} | { q2} | {q1, q2} |
| *{q2} | { q2} | { q2} |
| {q1} | { q2} | { q1} |

δ ({q0, q1},a)= ECLOSE(δ(q0,a) U δ(q1,a)) = ECLOSE(q0Uq2) = {q0, q1,q2}

δ ({q0, q1},b)= ECLOSE(δ(q0,b) U δ(q1,b)) = ECLOSE(Ø Uq1) = { q1}

δ ({q0, q1,q2},a)= ECLOSE(δ(q0,a) U δ(q1,a)U δ(q2,a)) = ECLOSE(q0Uq2 Uq2) = {q0, q1,q2}

δ ({q0, q1,q2},b)= ECLOSE(δ(q0,b) U δ(q1,b)U δ(q2,b)) = ECLOSE(Ø Uq1 Uq2) = {q1,q2}

δ ({ q1,q2},a)= ECLOSE( δ(q1,a)U δ(q2,a)) = ECLOSE(q2 Uq2) = {q2}

δ ({q1,q2},b)= ECLOSE( δ(q1,b)U δ(q2,b)) = ECLOSE(q1 Uq2) = {q1,q2}

(b) Write the difference between DFA, NFA and Ɛ-NFA

| **DFA** | **NFA** | **Ɛ- NFA** |
|---|---|---|
| **1.**Only one transition on each Input | Zero, one or more transitions on same input | Zero, one or more transitions |
| 2.No Ɛ-transitions | No Ɛ-transitions | Ɛ-transitions |
| 3.$\delta: QX\sum \rightarrow Q$ | $\delta: QX\sum \rightarrow 2^{Q}$ | $\delta: QX\sum U\{\varepsilon) \rightarrow 2^{Q}$ |

**3M** **CO1** L3

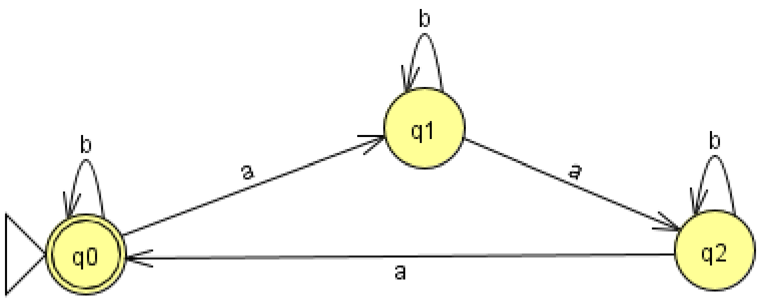
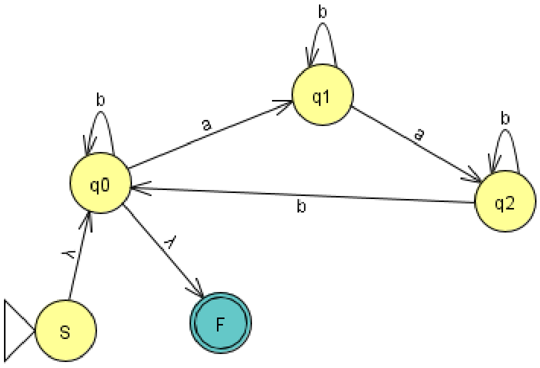
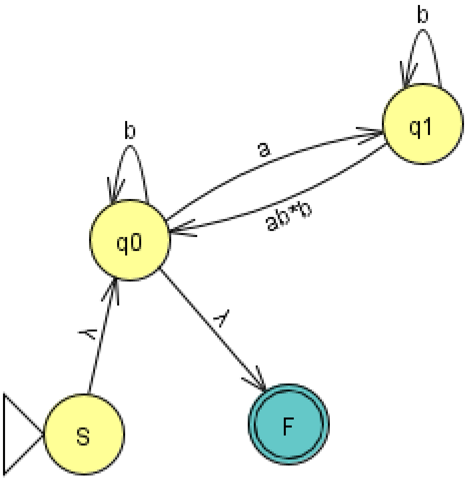
4 (a) Define Regular Expression. Construct RE for

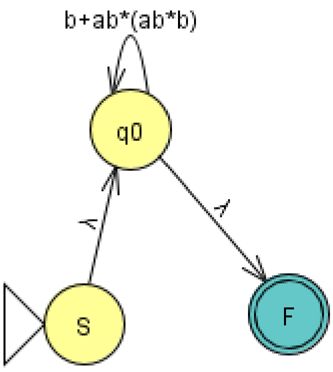
A regular expression offers a declarative way to express strings of a regular language. The constants ε, Φ are regular expressions. Any symbol a ∈Σ is a regular expression.

Let E, F be variables denoting a regular expression. The operators of a regular expression are

**6M** **CO2** L3

<table>
<tr><td></td><td>E+F (Union), E.F (Concatenation), E* (Kleene Closure), and (E) – parenthesis for readability.<br>i) L = {$a^{2n}b^{2m}$ | n, m ≥ 0 }<br>(aa)*(bb)*<br><b>ii)</b> strings over Σ={a,b,c} starting with <b>a</b> and ending with <b>b</b><br>a (a+b+c)*b<br>iii) L = {$a^nb^m$ | n ≥ 2, m ≤ 3 }<br>aaa*(ε+b+b+bb+bbb)</td><td></td><td></td><td></td></tr>
<tr><td>(b)</td><td>State and prove L = {$a^nb^m$ | n ≥ m} is not regular<br>According to the pigeon hole principle, if there are k states and there is k+1 input, then it must stay in a state multiple number of times.<br>The pumping lemma can prove this.<br>Theorem:<br>Let L be a regular language. Then there exists a constant n (which depends on L ) such that for every string w in L such that |w| ≥n, we can break w into 3 strings, w=xyz such that<br>1. y ≠ ε<br>2. |xy| ≤ n<br>3. For all k ≥ 0, the string $xy^kz$ is also in L<br>That is, we can always find a non-empty string y not too far from the beginning of w that can be “pumped”, i.e., repeating y any number of times or deleting it keeps the resulting string in the language L.<br><br>Player 1 : Language L is not regular<br>Player 2 : Choose a value for n<br>Let’s assume a DFA exists for L with <b>n</b> states, with n=3.<br><br>Player 1 : w = aaabbb<br><br>Player 2 : Decide on string split |xy| ≤ n, y ≠ ε<br><table><tr><td><b>aa</b></td><td><b>a</b></td><td><b>bbb</b></td></tr><tr><td><b>x</b></td><td><b>y</b></td><td><b>z</b></td></tr></table>Player 1 : k = 0<table><tr><td><b>aa</b></td><td><b>ε</b></td><td><b>bbb</b></td></tr><tr><td><b>x</b></td><td><b>y</b></td><td><b>z</b></td></tr></table>Let us pump y <b>0</b> times. The resulting string would be w = $a^2b^3 \notin$ L<br>Hence it is proved that the language $a^nb^m$: n>=m is not <b>regular.</b></td><td><b>4M</b></td><td><b>CO2</b></td><td>L3</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| 5 (a) | Convert to RE using state elimination method<br><br> <br>Create New start state and final state<br><br> b<br>b<br>b<br>a<br>a<br>b<br>q0<br>q1<br>q2<br>λ<br>λ<br>S<br>F<br>Eliminate State q2<br>q1-q2-q0<br><br> <br>Eliminate q1<br>Q0-q1-q0 | **6M** | **CO1** | L1 |

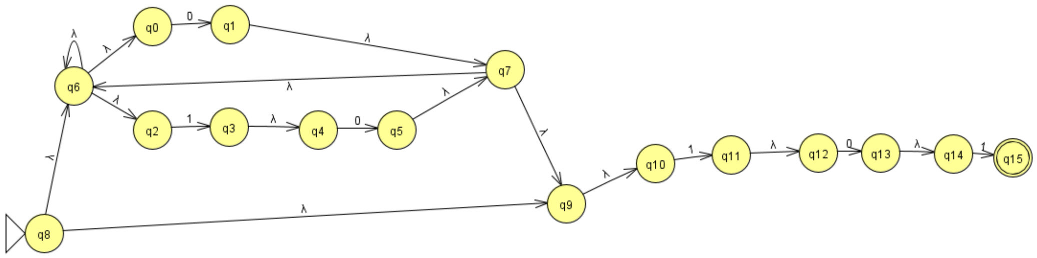


RE from S-F

(b+ab*(ab*b))*

(b)

Convert to ε-NFA, (0+10)*101



**4M** **CO1** L3

6 (a)

Minimize the following DFA:

| **δ** | **0** | **1** |
|---|---|---|
| **→A (0)** | B(1) | E (4) |
| **B(1)** | C(2) | F(5) |
| ***C(2)** | D(3) | H(7) |
| **D(3)** | E(4) | H(7) |
| **E(4)** | F(5) | I(8) |
| ***F(5)** | G(6) | B(1) |
| **G(6)** | H(7) | B(1) |
| **H(7)** | I(8) | C(2) |
| ***I(8)** | A(0) | E(4) |

(A,H),0 → (B,I)x

(A,G),0 → (B,H), (A,G),1 → (E,B)

(A,E),0 → (B,F)x

**6M** **CO2** L2

(A,D),0→ (B,E), (A,D),1 → (E,H)

(A,B),0 →(B,C)x

(B,H),0 → (C,I), (B,H), 1 →(F,C)

(B,G),0 → (C,H)x

(B,E),0 → (C,F), (B,E),1 →(F,I)

(B,D),0 → (C,E)x

(C,I),0 →(D,A), (C,I),1 →(H,E)

(C,F),0→(D,G), (C,F),1→(H,B)

(D,H),0→(E,I)x

(D,G),0→ (E,H), (D,G),1→(H,B)

(D,E),0→ (E,F)x

(E,H),0→(F,I), (E,H),1→(I,C)

(E,G),0→(F,H)x

(F,I),0→(G,A), (F,I),1→(B,E)

(G,H),0→(H,I)x

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| B | x | | | | | | | |
| *C | x | x | | | | | | |
| D | | x | x | | | | | |
| E | x | | x | x | | | | |
| *F | x | x | | x | x | | | |
| G | | x | x | | x | x | | |
| H | x | | x | x | | x | x | |
| *I | x | x | | x | x | | x | x |
| | A | B | *C | D | E | *F | G | H |

| | 0 | 1 |
|---|---|---|
| →[A,D,G] | [B,E,H] | [B,E,H] |
| [B,E,H] | [C,F,I] | [C,F,I] |

| | | |
|---|---|---|
| *[C,F,I] | [A,D,G] | [B,E,H] |

| | | | | |
|---|---|---|---|---|
| (b) | Define CFG. Write CFG for L = $\{ww^R \mid w \in \{0,1\}^*\}$. Define the grammar and derive the strings w= 011110, w = 1001<br>G = (V,T,S,P)<br>V : the finite set of variables, called non-terminals<br>T : the finite set of terminals that form the string of the language being defined<br>S: $S \in V$, the start symbol<br>P: finite set of productions/rules that consist of<br>(a) Variable – head on the LHS<br>(b) → Production symbol<br>(c) A string of 0 or more terminals or variables $(V \cup T)^*$<br>$S \to 0S0 \mid 1S1 \mid \varepsilon$<br>G= (V,T,S, P)<br>= $(\{S\}, \{0,1\}, S, \{S \to 0S0, S \to 1S1, S \to \varepsilon\})$<br>w = 011110<br>S=>0S0<br>=>01S10<br>=>011S110<br>=>011110<br>W= 1001<br>S=>1S1<br>=>10S01<br>=>1001 | **4M** | **CO2** | L3 |

**CI** **CCI** **HOD**

| Course Outcomes | | Blooms Level | Modules covered | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 | PSO4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | Apply the fundamentals of automata theory to write DFA, NFA, Epsilon-NFA and conversion between them. | **L1, L2, L3** | **1** | 3 | 2 | 2 | - | 2 | - | - | - | - | - | - | - | 2 | 2 | - | 3 |
| CO2 | Prove the properties of regular languages using regular expressions. | **L1, L2** | **2** | 3 | 3 | 2 | 3 | - | - | - | - | - | - | - | - | 2 | 2 | - | 3 |
| CO3 | Design context-free grammars (CFGs) and pushdown automata (PDAs) for formal languages. | **L1, L2, L3** | **3,4** | 3 | 3 | 2 | 3 | 2 | - | - | - | - | - | - | - | - | 2 | - | 3 |
| CO4 | Design Turing machines to solve the computational problems. | **L1, L2, L3** | **5** | 2 | 3 | 2 | 3 | 2 | - | - | - | - | - | - | - | - | 2 | - | 3 |
| CO5 | Explain the concepts of decidability and undecidability | **L1, L2, L3** | **5** | 3 | 2 | 2 | 3 | - | - | - | - | - | - | - | - | - | 2 | - | 3 |

## CO PO Mapping

| COGNITIVE LEVEL | REVISED BLOOMS TAXONOMY KEYWORDS |
|---|---|
| L1 | List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc. |
| L2 | summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend |

| | |
|---|---|
| L3 | Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover. |
| L4 | Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer. |
| L5 | Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize. |

| PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO) | | | | CORRELATION LEVELS | |
|---|---|---|---|---|---|
| PO1 | Engineering knowledge | PO7 | Environment and sustainability | 0 | No Correlation |
| PO2 | Problem analysis | PO8 | Ethics | 1 | Slight/Low |
| PO3 | Design/development of solutions | PO9 | Individual and team work | 2 | Moderate/ Medium |
| PO4 | Conduct investigations of complex problems | PO10 | Communication | 3 | Substantial/ High |
| PO5 | Modern tool usage | PO11 | Project management and finance | | |
| PO6 | The Engineer and society | PO12 | Life-long learning | | |
| PSO1 | Develop applications using different stacks of web and programming technologies | | | | |
| PSO2 | Design and develop secure, parallel, distributed, networked, and digital systems | | | | |
| PSO3 | Apply software engineering methods to design, develop, test and manage software systems. | | | | |
| PSO4 | Develop intelligent applications for business and industry | | | | |