(Affiliated to University of Mumbai)	*End Semester Examination (R-24/R-25) SH 2026 Answer Key with marking scheme*
Branch: ECS	Course: Discrete Structures and Automata Theory
Year/ Semester:TE/V	Course code: ECC503
Time: 03 hours	Marks: 80

		Marks
Q. 1	All questions are compulsory.	
A.	Principle of Mathematical Induction (PMI) The Principle of Mathematical Induction states that if a statement $P(n)$ satisfies: 1. Basis Step: $P(1)$ is true. 2. Inductive Step: If $P(k)$ is true for some positive integer k, then $P(k+1)$ is also true. Then $P(n)$ is true for all positive integers $n \geq 1$. --- To Prove $$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$ for all positive integers n. Step 1: Basis Step For $n = 1$, LHS: $$1$$ RHS: $$\frac{1(1+1)}{2} = \frac{1 \times 2}{2} = 1$$ LHS = RHS. Hence, the statement is true for $n = 1$. --- Step 2: Inductive Hypothesis Assume the statement is true for $n = k$, where k is any positive integer. That is, $$1 + 2 + 3 + \cdots + k = \frac{k(k+1)}{2}$$ --- Step 3: Inductive Step We must prove that the statement is true for $n = k + 1$. Consider: $$1 + 2 + 3 + \cdots + k + (k+1)$$ Using the inductive hypothesis,	

$$= \frac{(k+1)(k+2)}{2}$$
$$= \frac{(k+1)((k+1)+1)}{2}$$

which is exactly the required formula for $n = k+1$.
Hence,

$$1 + 2 + 3 + \cdots + k + (k+1) = \frac{(k+1)(k+2)}{2}$$

Conclusion
Since:

- The statement is true for $n = 1$, and
- Truth of $P(k)$ implies truth of $P(k+1)$,

by the Principle of Mathematical Induction,

$$\boxed{1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}}$$

for all positive integers n. ■

B. Definition of a Binary Relation
A binary relation R on a set A is a subset of the Cartesian product $A \times A$. That is,

$$R \subseteq A \times A$$

If $(a, b) \in R$, then a is said to be related to b.

Given

$$A = \{1, 2, 3\}$$
$$R = \{(1,1), (2,2), (3,3), (1,2), (2,1)\}$$

We check the properties one by one.

---

1. Reflexive
A relation R on A is reflexive if $(a, a) \in R$ for every $a \in A$.
Required pairs:

$$(1,1), (2,2), (3,3)$$

All these pairs are present in R.
Therefore, R is reflexive.

---

2. Symmetric
A relation R is symmetric if whenever $(a, b) \in R$, then $(b, a) \in R$.
Check:

- $(1,2) \in R$ and $(2,1) \in R$ ✓
- $(1,1), (2,2), (3,3)$ are self-pairs and satisfy symmetry automatically.

Therefore, R is symmetric.

---

3. Antisymmetric
A relation R is antisymmetric if whenever $(a, b) \in R$ and $(b, a) \in R$, then $a = b$.
But in R,

$$(1,2) \in R \text{ and } (2,1) \in R$$

while

$$1 \neq 2$$

	This violates the condition. Therefore, R is not antisymmetric. 4. Transitive A relation R is transitive if whenever $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$. Check the non-trivial cases: • $(1,2)$ and $(2,1) \Rightarrow (1,1)$ ✓ • $(2,1)$ and $(1,2) \Rightarrow (2,2)$ ✓ • $(1,1)$ and $(1,2) \Rightarrow (1,2)$ ✓ • $(2,2)$ and $(2,1) \Rightarrow (2,1)$ ✓ • $(1,2)$ and $(2,2) \Rightarrow (1,2)$ ✓ • $(2,1)$ and $(1,1) \Rightarrow (2,1)$ ✓ All required pairs are present. Therefore, R is transitive.	
C.	Regular Expressions and Regular Grammar 1. Regular Expression (RE) A Regular Expression is an algebraic notation used to represent a regular language over a given alphabet. It describes patterns of strings using operators such as union, concatenation, and Kleene star. Examples • a represents the language {a} • b represents the language {b} • a+b represents {a, b} • a* represents {ε, a, aa, aaa, ...} 2. Regular Grammar (RG) A Regular Grammar is a grammar in which productions are restricted to either right-linear or left-linear forms. Right-Linear Grammar Production rules are of the form: A → aB A → a A → ε where: • A, B are non-terminals • a is a terminal Example S → aS | bS | ε This grammar generates all strings over {a, b} such as: ε, a, b, ab, ba, aba, babb, ... Operations on Regular Expressions The three basic operations used to construct regular expressions are: 1. Union (+) Union represents a choice between two expressions. Definition If $R1$ and $R2$ are regular expressions, then R1 + R2	

denotes all strings belonging to either $R1$ or $R2$.
Example
a + b
Language:
{a, b}
Strings accepted:
a
b

---

2. Concatenation
Concatenation joins strings from two languages.
Definition
If $R1$ and $R2$ are regular expressions, then
R1R2
denotes strings formed by appending strings of $R2$ after strings of $R1$.
Example
ab
Language:
{ab}
Another Example
(a+b)c
Language:
{ac, bc}

---

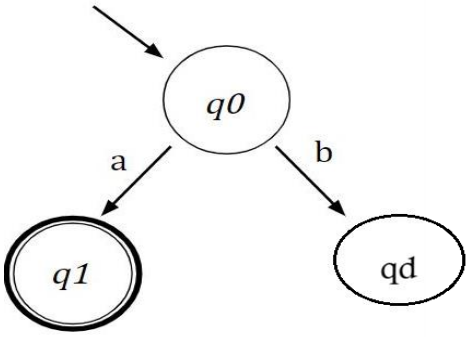
3. Kleene Star (*)
Kleene Star denotes zero or more repetitions of a regular expression.
Definition
For a regular expression R,
R*
* represents
{*} ∪ R ∪ RR ∪RRR ∪...
``
Example
a*
Language:
{ε, a* aa, aaa, aaaa, ...}

Example
(ab)*
Language:
{ε, ab, a*ab, ababab, ...}

Combined Example
Consider the regular expression:
(a+b)*ab
Explanation

- (a+b)* generates any string of a's and b's.
- ab must appear at the end.

Strings Generated

	ab aab *ab* *aaab* *abab* *bbab* Thus, it represents all strings over {a,b} ending with "ab". Relationship Between Regular Expressions and Regular Grammar Both regular expressions and regular grammars describe the same class of languages, called regular languages. Example Regular Expression: a*b Equivalent Regular Grammar: S → aS |*b Generated strings: b, ab, aab, aaab, ... Conclusion • A Regular Expression is a symbolic method for describing regular languages. • A Regular Grammar generates regular languages using restricted production rules. • The fundamental operations in regular expressions are: o Union (+): Choice between strings. o Concatenation: Joining strings. o Kleene Star (*): Repetition of strings zero or more times. These operations are sufficient to describe all regular languages accepted by finite automata.	
D.		

Present State	Input a	Input b
$\rightarrow q_0$	q_1	q_d
$*q_1$	q_1	q_1
q_d	q_d	q_d

Here, $\rightarrow$ indicates the initial state and $*$ indicates a final state.

E.

Sr. No.	DFA (Deterministic Finite Automaton)	NFA (Nondeterministic Finite Automaton)
1	For each state and input symbol, there is exactly one transition.	For each state and input symbol, there may be zero, one, or multiple transitions.
2	Only one transition path is possible for an input string.	Multiple transition paths may be possible for an input string.
3	epsilon -transitions are not allowed.	In an epsilon-NFA, epsilon-transitions are allowed.
4	δ: QxΣ -> Q i.e. next possible state belongs to Q.	δ: Qx(Σ U ε) -> 2^Q i.e. next possible state belongs to power set of Q.
5	A string is accepted if the single path ends in a final state.	A string is accepted if at least one possible path ends in a final state.
6	May require more states than an equivalent NFA.	May require fewer states than an equivalent DFA.
7	Both have the same computational power as finite automata.	Both recognize exactly the regular languages.

F.

Let $\Sigma = 0,1$.

(a) All strings ending in 1

Any number of 0s and 1s can appear before the last symbol, but the string must end with 1.

Regular Expression:

(0+1)*1

Examples: 1, 01, 101, 0001, 1101

(b) All strings containing at least one 0

A 0 must occur somewhere in the string. Any combination of 0s and 1s may appear before and after it.

Regular Expression:

(0+1)*0(0+1)*

Examples: 0, 01, 10, 1011, 11001

	(c) All strings containing exactly two 1s The string must contain exactly two occurrences of 1. Any number of 0s may appear before, between, and after the two 1s. Regular Expression: 0*10*10* Examples: 11, 101, 1100, 0010100 Not Accepted: 1, 111, 10101 (contains more than two 1s)	
Q.2 A	Let: • M =Mathematics students = 120 • S =Statistics students = 90 • C =Computer Science students = 70 Given: $\lvert M \cap S \rvert = 50,\ \lvert M \cap C \rvert = 40,\ \lvert S \cap C \rvert = 30$ $\lvert M \cap S \cap C \rvert = 20$ Total students = 200 Using the Principle of Inclusion and Exclusion $\lvert M \cup S \cup C \rvert = \lvert M \rvert + \lvert S \rvert + \lvert C \rvert - \lvert M \cap S \rvert - \lvert M \cap C \rvert - \lvert S \cap C \rvert + \lvert M \cap S \cap C \rvert$ Substituting the values: $= 120 + 90 + 70 - 50 - 40 - 30 + 20$ $= 280 - 120 + 20$ $= 180$ Number of students studying at least one subject $\boxed{180}$ Number of students studying none of the subjects $\text{None} = \text{Total Students} - \lvert M \cup S \cup C \rvert$ $= 200 - 180$ $= \boxed{20}$ • Students studying at least one subject: $\boxed{180}$ • Students studying none of the subjects: $\boxed{20}$	

OR

Q.2 B.

Proof by Mathematical Induction

We want to prove that

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \left[\frac{n(n+1)}{2}\right]^2$$

for all positive integers n.

Step 1: Base Case ($n = 1$)

LHS:

$$1^3 = 1$$

RHS:

$$\left[\frac{1(1+1)}{2}\right]^2 = \left(\frac{2}{2}\right)^2 = 1$$

Thus,

$$1^3 = \left[\frac{1(1+1)}{2}\right]^2$$

Hence, the statement is true for $n = 1$.

Step 2: Induction Hypothesis

Assume the statement is true for $n = k$, where $k \geq 1$.

That is,

$$1^3 + 2^3 + \cdots + k^3 = \left[\frac{k(k+1)}{2}\right]^2.$$

Step 3: Induction Step

We must prove that the statement is true for $n = k + 1$.

Consider

$$1^3 + 2^3 + \cdots + k^3 + (k+1)^3.$$

Using the induction hypothesis,

$$= \left[\frac{k(k+1)}{2}\right]^2 + (k+1)^3.$$

Factor out $(k+1)^2$:

$$= (k+1)^2\left(\frac{k^2}{4} + k + 1\right).$$

$$= (k+1)^2\left(\frac{k^2 + 4k + 4}{4}\right).$$

$$= (k+1)^2\left(\frac{(k+2)^2}{4}\right).$$

$$= \left[\frac{(k+1)(k+2)}{2}\right]^2.$$

This is exactly the required formula for $n = k + 1$.

Therefore,

$$1^3 + 2^3 + \cdots + k^3 + (k+1)^3 = \left[\frac{(k+1)(k+2)}{2}\right]^2.$$

Conclusion

Since the statement is true for $n = 1$and true for $n = k + 1$whenever it is true for $n = k$, by the Principle of Mathematical Induction,

$$\boxed{1^3 + 2^3 + \cdots + n^3 = \left[\frac{n(n+1)}{2}\right]^2}$$

for all positive integers n.

Q.3 A

1. DFA Definition

A DFA is defined as:

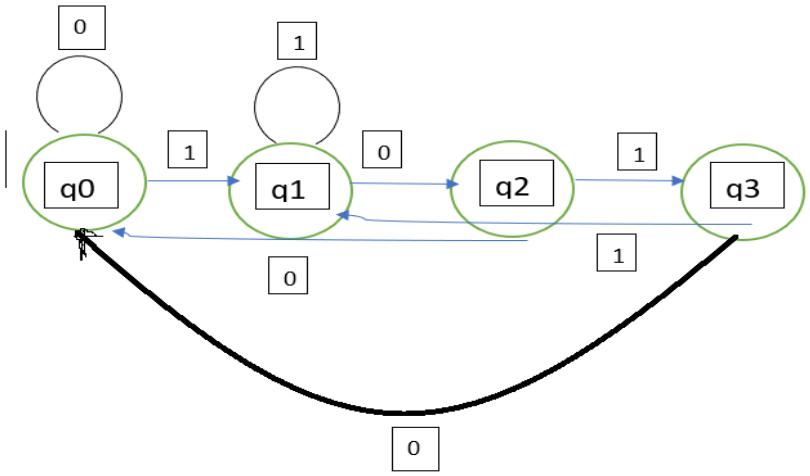
$$M = (Q, \Sigma, \delta, q0, F)$$

where

- $Q = q0, q1, q2, q3$
- $\Sigma = 0,1$
- Start State = $q0$
- Final State = $F = q3$

Meaning of States

- $q0$: No useful pattern detected.
- $q1$: Last symbol read is 1.
- $q2$: Last symbols read are 10.
- $q3$: Last symbols read are 101 (Accept State).



Transition Table

Present State	Input 0	Input 1
→ q0	q0	q1
q1	q2	q1
q2	q0	q3
*q3	q2	q0

(→ Start State, * Final State)

---

3. Testing the DFA

Test Case 1

Input: 101

q0 →*q1 → q2 → q3

Accepted

Test Case 2

Input: 1101

q0 → q1*→ q1 → q2 → q3

Accepted

Test Case 3

Input: 10101

q0→q1→*2*→*q3*→*q2*→*q3*
``

Accepted

Test Case 4

Input: 1001

q0→q1→q2→*0→q1

Rejected

Does not end with 101.

Test Case 5

Input: 1110

q0→q1→q1*q1→q2

Rejected

Does not end with 101.

Test Summary

Input String	Final State	Result
101	q3	Accept
1101	q3	Accept
10101	q3	Accept
1001	q1	Reject
1110	q2	Reject

Q.3 B

Problem Requirements

A valid vehicle tag has:

- First 2 characters: uppercase letters (A-Z)
- Last 3 characters: digits (0-9)

1. Alphabet

Let the input alphabet be:

$$\Sigma = A, B, \dots, Z, 0, 1, \dots, 9$$

or simply

$$\Sigma = L \cup D$$

where

- L =set of uppercase letters A-Z
- D =set of digits 0-9

---

2. States and Their Meanings

State	Meaning
q0	Start state, no characters read
q1	First uppercase letter read
q2	Two uppercase letters read
q3	Two letters + first digit read
q4	Two letters + two digits read

q5	Two letters + three digits read (Accept State)
qd	Dead/Trap state for invalid input

---

3. Start State

$$q0$$

---

4. Accepting State

$$F = q5$$

The DFA accepts only when exactly two letters followed by exactly three digits have been read.

---

5. Transition Logic

From q0

- Letter → q1
- Digit → qd

From q1

- Letter → q2
- Digit → qd

From q2

- Digit → q3
- Letter → qd

From q3

- Digit → q4
- Letter → qd

From q4

- Digit → q5
- Letter → qd

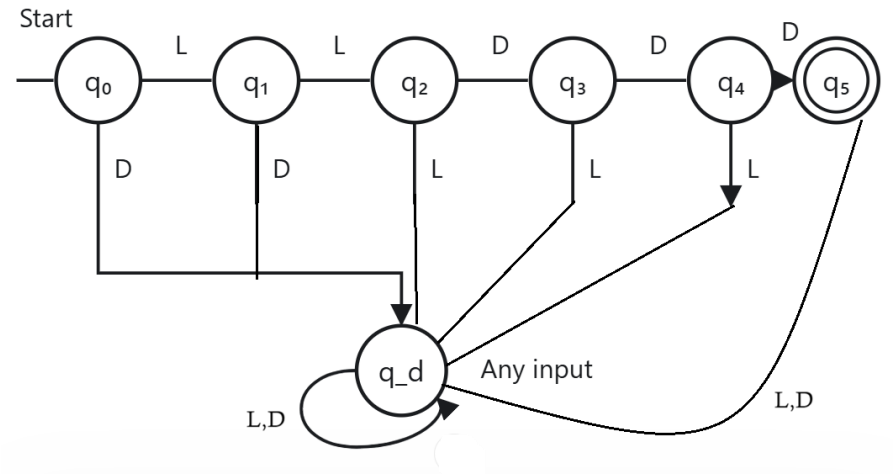
From q5

Any additional input → qd

From qd

Any input → qd

State Transition Diagram



q5* = Accept State

Any symbol from q5 → qd

Any symbol from qd → qd

Transition Table

Present State	Letter (A-Z)	Digit (0-9)
→ q0	q1	qd
q1	q2	qd
q2	qd	q3
q3	qd	q4
q4	qd	q5
*q5	qd	qd
qd	qd	qd

DFA Formal Definition

$$M = (Q, \Sigma, \delta, q, _0F)$$

where

$$Q = q0, q1, q2, q3, q4, q5, qd$$
$$\Sigma = A - Z, 0 - 9$$
$$q0 = \text{start state}$$
$$F = q5$$

Q.4 A

Pigeonhole Principle and Extended Pigeonhole Principle

1. Pigeonhole Principle

Statement:

If $n + 1$(or more) objects are placed into $n$boxes, then at least one box contains at least two objects.

Mathematically,

If $N > k$, then some box contains at least 2 objects.

---

2. Extended Pigeonhole Principle

Statement:

If Nobjects are distributed among $k$boxes, then at least one box contains at least

$$\left\lceil \frac{N}{k} \right\rceil$$

objects, where $\lceil x \rceil$denotes the ceiling function.

Problem 1: Among Any 13 People, At Least Two Were Born in the Same Month

Proof

Consider:

- Objects (pigeons) = 13 people
- Boxes (pigeonholes) = 12 months of the year

If each month had at most one person born in it, then the maximum number of people would be 12.

But there are 13 people and only 12 months.

Since

$$13 > 12,$$

by the Pigeonhole Principle, at least one month must contain the birthdays of at least two people.

Conclusion

Therefore, among any 13 people, at least two were born in the same month. ✓

Problem 2: Minimum Number of Objects Required to Guarantee at Least 4 Objects in One of 5 Boxes

Given:

- Number of boxes $k = 5$

- Need to guarantee at least 4objects in one box.

Step 1: Find the maximum number of objects that can be placed without having 4 objects in any box.

To avoid having 4 objects in a box, place at most 3 objects in each box.

$$5 \times 3 = 15$$

Thus, 15 objects can be distributed as:

$$(3, 3, 3, 3, 3)$$

and no box contains 4 objects.

Step 2: Add one more object

Placing the 16th object forces one box to contain

$$4$$

objects.

By the Extended Pigeonhole Principle,

$$\left\lceil \frac{16}{5} \right\rceil = \lceil 3.2 \rceil = 4$$

So at least one box must contain 4 objects.

Conclusion

$$\boxed{16}$$

objects are required to guarantee that at least 4 objects are placed in one of the 5 boxes.

$$\boxed{16}$$

OR

Q.4 B

Given:

$$f(x) = 3x + 5,\ f: \mathbb{R} \to \mathbb{R}$$

Step 1: Find the inverse function

Let

$$y = 3x + 5$$

Solve for x:

$$y - 5 = 3x$$

$$x = \frac{y - 5}{3}$$

Now interchange $x$and y:

$$f^{-1}(x) = \frac{x - 5}{3}$$

Inverse Function

$$\boxed{f^{-1}(x) = \frac{x - 5}{3}}$$

Step 2: Verify using composition

(i) Verify $f\left(f^{-1}(x)\right) = x$

$$f\left(\frac{x-5}{3}\right) = 3\left(\frac{x-5}{3}\right) + 5$$

$$= x - 5 + 5$$

$$= x$$

Hence,

$$\boxed{f\left(f^{-1}(x)\right) = x}$$

(ii) Verify $f^{-1}(f(x)) = x$

$$f^{-1}(3x+5) = \frac{(3x+5)-5}{3}$$
$$= \frac{3x}{3}$$
$$= x$$

Hence,

$$\boxed{f^{-1}(f(x)) = x}$$

Conclusion

The inverse of the function $f(x) = 3x + 5$is

$$\boxed{f^{-1}(x) = \frac{x-5}{3}}$$

Since

$$f(f^{-1}(x)) = x \text{ and } f^{-1}(f(x)) = x,$$

the inverse is verified by composition of functions.

Q.5 A

Given Weighted Graph

Edges:

- $AB = 4$
- $AC = 2$
- $BC = 1$
- $BD = 5$
- $CD = 8$
- $CE = 10$
- $DE = 2$

Start vertex: A

Step 1: Initialize

Vertex	Distance from A	Previous
A	0	—
B	∞	—
C	∞	—
D	∞	—
E	∞	—

Permanent set: $\{A\}$

Step 2: Update Neighbors of A

From A:

- $A \to B = 4$
- $A \to C = 2$

Vertex	Distance
A	0
B	4
C	2
D	∞
E	∞

Choose smallest temporary distance: C (2).
Permanent set: $\{A, C\}$

Step 3: Update Through C
From C:

- To B: $2 + 1 = 3$(better than 4)
- To D: $2 + 8 = 10$
- To E: $2 + 10 = 12$

Vertex	Distance
A	0
B	3
C	2
D	10
E	12

Choose smallest temporary distance: B (3).
Permanent set: $\{A, C, B\}$

Step 4: Update Through B
From B:

- To D: $3 + 5 = 8$(better than 10)

Vertex	Distance
A	0
B	3
C	2
D	8
E	12

Choose smallest temporary distance: D (8).
Permanent set: $\{A, C, B, D\}$

Step 5: Update Through D
From D:

- To E: $8 + 2 = 10$(better than 12)

Vertex	Distance
A	0
B	3

C	2
D	8
E	10

Choose E (10).
Permanent set: $\{A, C, B, D, E\}$
All vertices are finalized.

Shortest Paths from A

Destination	Shortest Distance	Path
A	0	A
B	3	A → C → B
C	2	A → C
D	8	A → C → B → D
E	10	A → C → B → D → E

Final Answer

- Shortest path from A to B: $A \to C \to B$, distance = 3
- Shortest path from A to C: $A \to C$, distance = 2
- Shortest path from A to D: $A \to C \to B \to D$, distance = 8
- Shortest path from A to E: $A \to C \to B \to D \to E$, distance = 10

Thus, the shortest-path distances from A are:

$$\boxed{A = 0, C = 2, B = 3, D = 8, E = 10}$$

OR

Q.5 B

Given Graph

$$V = A, B, C, D, E, F$$
$$E = AB, BC, CD, DA, DE, EF, FD$$

(a) Degree of Every Vertex

The degree of a vertex is the number of edges incident on it.

Vertex	Incident Edges	Degree
A	AB, AD	2
B	AB, BC	2
C	BC, CD	2
D	AD, CD, DE, DF	4
E	DE, EF	2

F	DF, EF	2

Result

$$deg(A) = 2,\ deg(B) = 2,\ deg(C) = 2$$
$$deg(D) = 4,\ deg(E) = 2,\ deg(F) = 2$$

(b) Classification of the Graph

Eulerian Graph Test

A connected graph is Eulerian if every vertex has an even degree.

Degrees:

- A = 2 (even)
- B = 2 (even)
- C = 2 (even)
- D = 4 (even)
- E = 2 (even)
- F = 2 (even)

All vertices have even degree and the graph is connected.

Therefore,

The graph is Eulerian

It contains an Euler circuit.

One Euler circuit is:

$$A \to B \to C \to D \to E \to F \to D \to A$$

(c) Hamiltonian Cycle

A Hamiltonian cycle visits every vertex exactly once and returns to the starting vertex.

Consider the cycle:

$$A \to B \to C \to D \to E \to F \to A$$

This is not possible because edge FAdoes not exist.

Trying another route:

$$A \to B \to C \to D \to F \to E$$

To complete the cycle we must return to A, but neither EA nor EB exists.

Observe that vertices E and F can only be reached through D and each has degree 2 inside the triangle $D - E - F$. To include both E and F in a Hamiltonian cycle, vertex D would need to be used more than once, which is not allowed in a Hamiltonian cycle.

Therefore,

The graph does not contain a Hamiltonian cycle

(d) Justification Using Degree Properties and Connectivity

Eulerian Property

- The graph is connected.
- Every vertex has even degree:

$$(2, 2, 2, 4, 2, 2)$$

- Hence, by Euler's theorem, the graph has an Euler circuit.

Graph is Eulerian

Hamiltonian Property

- The graph consists of two cycles:
  - $A - B - C - D - A$
  - $D - E - F - D$
- Vertex D acts as a cut-vertex joining the two parts of the graph.
- Any Hamiltonian cycle must visit every vertex exactly once.
- To include vertices E and F and also return to the square $ABCD$, the cycle would have to pass through D twice, violating the Hamiltonian-cycle condition.

No Hamiltonian cycle exists

Q.6 A	PDA for $L = a^n b^n \mid n \geq 1$	

The PDA pushes one stack symbol for every a read and pops one stack symbol for every b read. The string is accepted when all input is consumed and the stack returns to its initial symbol.

PDA Definition

$M = Q, \Sigma, \Gamma, \mathrm{q0}, \mathrm{Z0}, \mathrm{qf}$

Where:

- $Q = q0, q1, qf$
- $\Sigma = a, b$
- $\Gamma = A, Z0$
- Start state: $q0$
- Initial stack symbol: $Z0$
- Final state: qf
- $F = qf$

Instantaneous Descriptions for Input

Format:

$$(s, t, a, t, eremaining\ inputstack)$$

$$\delta(q_0, a, Z_0) = (q_1, XZ_0)$$
$$\delta(q_1, a, X) = (q_1, XX)$$
$$\delta(q_1, b, X) = (q_2, \varepsilon)$$
$$\delta(q_2, b, X) = (q_2, \varepsilon)$$
$$\delta(q_2, \varepsilon, Z_0) = (q_f, Z_0)$$

$$aabb \in L = a^n b^n \mid n \geq 1$$

Initial configuration:

$(q_0,\ aabb,\ Z_0)$

Read first **a** and push **A**:

$(q_0,\ abb,\ AZ_0)$

Read second **a** and push **A**:

$(q_0,\ bb,\ AAZ_0)$

Read first **b** and pop **A**:

$(q_1,\ b,\ AZ_0)$

Read second **b** and pop **A**:

$(q_1,\ \varepsilon,\ Z_0)$

Read second **b** and pop **A**:

$(q_1,\ \varepsilon,\ Z_0)$

Apply ε-move to final state:

$(q_f,\ \varepsilon,\ Z_0)$

Since the input is completely consumed and the PDA reaches the final state with only the bottom stack symbol remaining, the string aabb is accepted.

OR

Q.6 B

Ambiguity in Context-Free Grammars

A context-free grammar (CFG) is said to be ambiguous if there exists at least one string in its language that can be generated by more than one distinct parse tree (or equivalently, more than one leftmost derivation or rightmost derivation).

In an ambiguous grammar, the structure and meaning of a string are not uniquely determined.

Given Grammar

$$E \rightarrow E + E \mid E*E \mid id$$

Consider the string:

$$id + id*id$$

This string can be derived in two different ways.

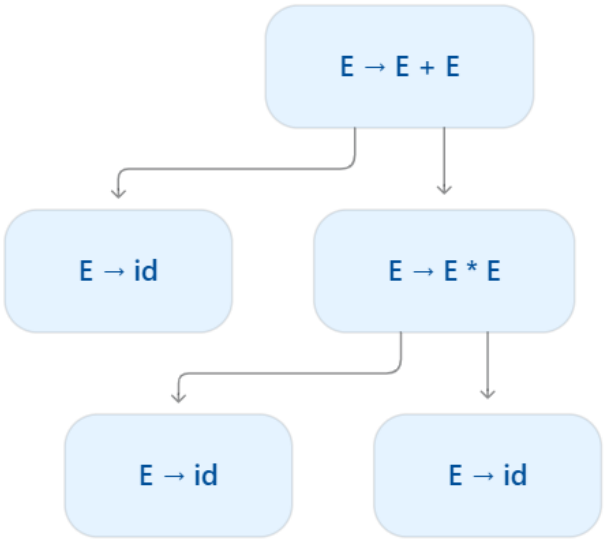
Derivation 1: $id + (id\backslash{*}id)$

Leftmost Derivation

$$E \Rightarrow E + E \Rightarrow id + E \Rightarrow id + E{*}E \Rightarrow id + id{*}E \Rightarrow id + id{*}id$$

Parse Tree

```
id + (id * id)
```



This corresponds to:

$$id + (id{*}id)$$

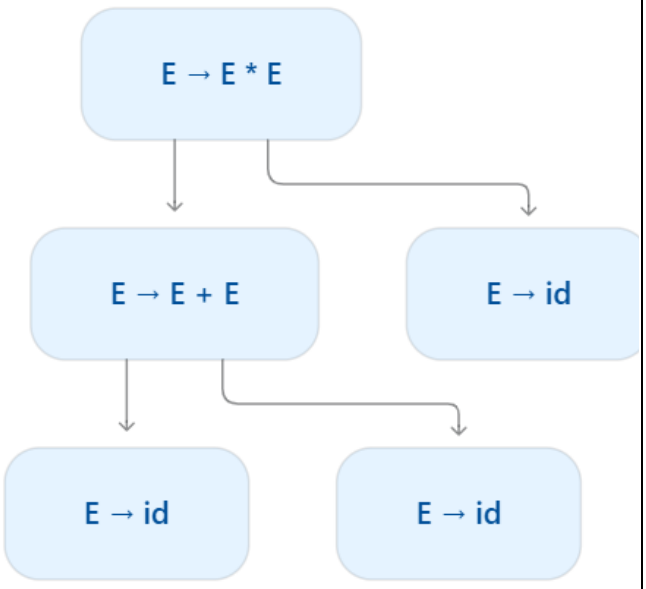
Derivation 2: $(id + id){*}id$

Leftmost Derivation

$$E \Rightarrow E{*}E \Rightarrow E + E{*}E \Rightarrow id + E{*}E \Rightarrow id + id{*}E \Rightarrow id + id{*}id$$

Parse Tree

```
(id + id) * id
```



This corresponds to:

$$(id + id)*id$$

Conclusion

The string

$$id + id*id$$

has two different parse trees:

1. $id + (id*id)$
2. $(id + id)*id$

Since the same string can be generated with two distinct parse trees, the grammar

$$E \rightarrow E + E \mid E*E \mid id$$

is ambiguous.

********* All the Best*********