RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Go, change the world®

USN										

Academic year 2024-2025 (ODD Sem)

DEPARTMENT OF
INDUSTRIAL ENGINEERING & MANAGEMENT

Date	**27-01-2025**	**Maximum Marks**	**10 + 50**
Course Code	**HS251TA**	**Duration**	**30 + 90 Min**
Sem	**V**	**CIE – I Scheme & Solution**	
Principles of Management and Economics			

Note:
1. Answer all the Questions.

Sl. No.	Questions	M	BT	CO
	Part – A			
1	Adam's Equity Theory	01	1	3
2	The value an individual places on the rewards.	01	1	3
3	Transactional leadership focuses on rewards and punishments, while transformational leadership focuses on vision and inspiration.	01	1	3
4	Team management	01	1	2
5	Lazy and averse to work	01	1	3
6	Equilibrium in the goods market.	01	1	5
7	The Complete Keynesian Model.	01	1	5
8	Aggregate expenditure determines output in the short run.	01	1	5
9	It influences aggregate expenditure and output levels.	01	1	4
10	The total output firms are willing to produce at different price levels.	01	1	5
	Part –B			
1	Herzberg's Two-Factor Theory categorizes job factors into two groups: **Hygiene Factors:** These are factors that are necessary to prevent job dissatisfaction but do not significantly contribute to higher job satisfaction or motivation. They include: **Salary and Benefits:** Employees may feel undervalued if their compensation is not competitive, leading to dissatisfaction. To address this, the company should offer competitive salaries and benefits that align with industry standards. **Work Conditions**: Poor working conditions, such as uncomfortable office spaces, lack of proper tools, or unsafe environments, can lead to dissatisfaction. Ensuring a safe, comfortable, and well-equipped workplace will mitigate dissatisfaction. **Job Security:** A lack of job security is a common cause of stress and dissatisfaction. Offering stable contracts and transparent communication about job stability can ease employee concerns. **Company Policies:** Ambiguous or unfavorable company policies can create confusion and frustration. The management should ensure clear, fair policies that are applied consistently. **Application of Hygiene Factors in the Case Study:**	10	4	3

	The company should review its salary structure and benefits package to ensure they are competitive in the market. Improving physical work conditions (e.g., better office equipment, ergonomically designed workstations, etc.) and enhancing job security through clear communication about business stability can prevent dissatisfaction. Clear and fair company policies would also reduce employees' feelings of being undervalued. **Motivators:** These factors are directly related to job satisfaction and can inspire employees to excel and remain engaged. They include: **Achievement:** Providing employees with opportunities to accomplish meaningful and challenging tasks helps to improve their sense of accomplishment and job satisfaction. **Recognition:** Employees seek recognition for their hard work and achievements. Publicly acknowledging accomplishments, offering awards, or providing constructive feedback can help increase motivation. **Growth and Advancement:** Offering opportunities for professional development and career growth can significantly impact employee morale and retention. Employees who feel they can advance within the company are more likely to stay committed. **Work Itself:** Providing employees with work that is interesting, challenging, and allows them to use their skills effectively can lead to greater job satisfaction. Responsibility: Allowing employees to take on more responsibility and make decisions fosters a sense of ownership and engagement. **Application of Motivators in the Case Study:** To address the issue of employees feeling undervalued, the company should focus on recognition and growth opportunities. Implementing a performance recognition program and offering career advancement training or leadership programs can improve engagement. Providing employees with more meaningful work that challenges them and offers opportunities for personal growth can also motivate them. This could include giving employees the chance to work on cross-functional projects, taking ownership of tasks, or having more autonomy in their roles. The company can also introduce mentoring programs, leadership training, and continuous learning opportunities to enhance professional growth.			
2	Vroom’s Expectancy Theory suggests that motivation is influenced by three key factors: **Expectancy**, **Instrumentality**, and **Valence**. These factors determine how employees perceive the relationship between their efforts, performance, and rewards, and thus influence their motivation. 1. **Expectancy**: This refers to the belief that increased effort will lead to improved performance. Employees are motivated when they believe their efforts will result in the desired outcomes. In the case of the project manager, the issue is that employees do not understand how their efforts contribute to the overall goals, leading to a drop in performance. **Recommendation**: To enhance **Expectancy**, the project manager should clarify how individual efforts align with the broader organizational goals. This can be done through regular communication, setting clear expectations, and providing training and resources that help employees feel confident in their ability to	10	3	3

	achieve the required performance levels. When employees understand that their hard work will lead to successful results, they are more likely to exert effort. 2. **Instrumentality**: This refers to the belief that good performance will lead to rewards or outcomes. If employees believe that their efforts will be recognized and rewarded, they are more likely to be motivated to perform well. **Recommendation**: The project manager should establish a clear link between performance and rewards. This can be done by introducing performance-based incentives, recognition programs, or feedback systems that show employees how their contributions are valued. By ensuring that rewards are based on performance, the project manager can motivate employees to perform at their best. 3. **Valence**: This refers to the value or importance that an individual places on the rewards or outcomes they expect to receive. Different employees value different types of rewards, so understanding what motivates each individual is crucial. **Recommendation**: The project manager should assess what rewards are most valued by employees, whether financial rewards, career advancement, recognition, or job satisfaction. By offering rewards that align with employees' preferences, the project manager can increase **Valence**, making the expected outcomes more desirable and motivating employees to perform at higher levels. **Conclusion**: To enhance motivation, the project manager should: • Improve **Expectancy** by clearly communicating how individual efforts contribute to organizational goals and ensuring employees have the resources and confidence to succeed. • Strengthen **Instrumentality** by creating a clear connection between performance and rewards, ensuring employees see that their hard work will lead to tangible outcomes. • Maximize **Valence** by offering rewards that are meaningful and desirable to employees, thus increasing the overall motivation to perform well. By addressing all three components of Vroom’s Expectancy Theory, the project manager can significantly enhance employee motivation and improve performance.			
3	Hersey and Blanchard’s Situational Leadership Model suggests different leadership styles based on the maturity level of team members: 1. **Low Maturity (M1)**: For team members with low skill or motivation, the leader should use a **Directing** style (high direction, low support). This involves providing clear instructions and close supervision. 2. **Moderate Maturity (M2)**: For team members who have some skills but need encouragement and guidance, the leader should use a **Coaching** style (high direction, high support). This includes providing direction while also supporting and motivating them. 3. **High Maturity (M3)**: For team members with the necessary skills but who need less direction, the leader should use a **Supporting** style (low direction, high support). Here, the leader focuses on providing encouragement and involvement, allowing the team member to take more responsibility.	10	3	3

	4. **Very High Maturity (M4)**: For highly skilled and motivated team members, the leader should use a **Delegating** style (low direction, low support). This involves giving them autonomy and trusting them to make decisions. By adjusting leadership styles based on the maturity of each team member, the leader can effectively manage diverse skill levels and enhance overall team performance.			
4	The Keynesian Cross Model represents the equilibrium level of national income in an economy where aggregate demand (AD) is determined by consumption, investment, and government spending. The basic idea is that output (income) is determined by the total expenditure in the economy, which includes consumption (C), investment (I), and government spending (G). In the Keynesian Cross, the equilibrium level of national income occurs when total expenditure (AD) equals the total output (Y). This is where the 45-degree line (which represents points where income equals expenditure) intersects the aggregate demand curve. At this point, the economy is in equilibrium. **Government Spending and Investment:** An increase in government spending (G) or investment (I) shifts the AD curve upward, leading to a higher equilibrium national income. Since investment and government spending are autonomous components of demand, they directly affect the level of output. This is the basis for Keynesian fiscal policy, where increasing G or I can boost output during periods of economic downturn.	10	2	4
5	The **IS-LM model** represents the equilibrium in both the goods market (IS curve) and the money market (LM curve). The model shows the relationship between the interest rate (i) and national income (Y) in determining equilibrium in both markets. • **IS Curve (Goods Market):** The IS curve shows combinations of interest rates and national income where the goods market is in equilibrium (i.e., where investment equals saving). A higher interest rate leads to lower investment, shifting the IS curve left. Conversely, a lower interest rate increases investment and shifts the IS curve right. • **LM Curve (Money Market):** The LM curve shows combinations of interest rates and income levels where the money market is in equilibrium (i.e., where money supply equals money demand). An increase in income leads to higher demand for money, pushing up interest rates and shifting the LM curve upward. Conversely, a decrease in income reduces money demand, shifting the LM curve downward. **Effect of Policies:** • **Fiscal Policy:** An increase in government spending or a tax cut shifts the IS curve to the right, increasing income and output. This leads to higher interest rates in the short run. • **Monetary Policy:** An increase in the money supply shifts the LM curve to the right,	10	2	5

		lowering interest rates and stimulating investment, which boosts output and income.			

BT-Blooms Taxonomy, CO-Course Outcomes, M-Marks

Marks Distribution	**Particulars**		**CO1**	**CO2**	**CO3**	**CO4**	**CO5**	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**
	Quiz	**Max Marks**	-	01	04	01	04	10	-	-	-	-	-
	Test		-	-	30	10	10	-	20	20	10	-	-

RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Go, change the world

Academic year 2024-2025 (Odd Sem)

Date	27/01/2025	Maximum Marks	50+10
Course Code	IS353IA	Duration	90+30
Sem	V	CIE III (Scheme)	
UG/~~PG~~	UG	Faculty: MEM/AS/VH/VG/JS/SHRS/ARA	

Artificial Intelligence and Machine Learning

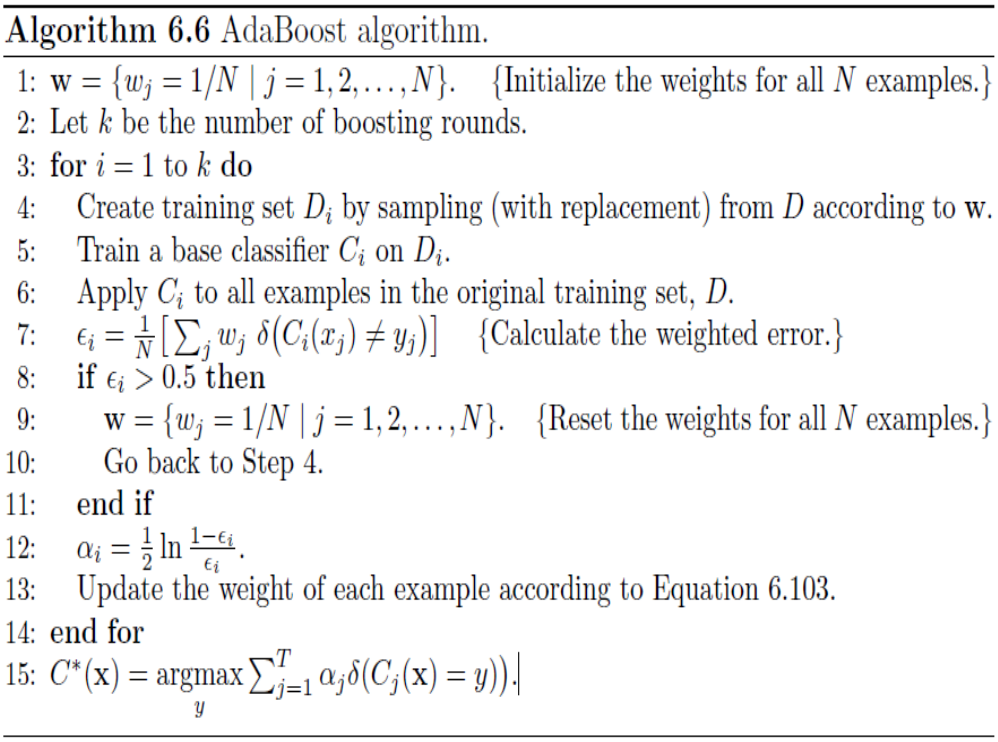
(Common to AIML/CSE/CD/CY/ISE)

Note: - Students need to add comments to their answers wherever required.

Q. No.	Questions	M
1. A	The **utility function** represents the payoff each player receives in a game. UTILITY(s, p) A game is called a **zero-sum game** because the total payoff is constant (usually zero), meaning one player's gain is exactly balanced by the other player's loss.	**5**
B	Alpha-Beta Pruning is an optimization technique for the Minimax algorithm. It reduces the number of nodes evaluated in the search tree by eliminating branches that cannot affect the final decision. The algorithm uses two parameters: - **Alpha (α):** The best value that the maximizing player is guaranteed. - **Beta (β):** The best value that the minimizing player is guaranteed. **Efficiency of Alpha-Beta Pruning** 1. **Reduction in Nodes:** The number of nodes explored is significantly reduced, especially in deeper trees. ○ Without pruning: $O(b^d)$, where b is the branching factor, and d is the depth. ○ With pruning: Best-case $O(b^{d/2})$. 2. **Preserves Correctness:** The same optimal decision as Minimax is guaranteed. 3. **Improves Practicality:** Pruning allows deeper exploration within the same time constraints. By intelligently skipping unnecessary branches, Alpha-Beta Pruning ensures faster decision-making while maintaining the integrity of the Minimax result.	**5**
2. A	Generalization error Gen.error(m) = c_1 * Noise + Bias (m) + c_2 * Variance (m) Ensemble Method increase classification accuracy: Rational for ensemble method $e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06,$	**05**
B	**Bagging** (Bootstrap Aggregating) is an ensemble learning technique that improves model accuracy by training multiple models on different random subsets of the data (created	**05**

through sampling with replacement) and then aggregating their predictions. For classification, it uses majority voting, and for regression, it averages the predictions. Bagging reduces variance and helps prevent overfitting,

3. A



Algorithm 6.6 AdaBoost algorithm.

```
1:  $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$.   {Initialize the weights for all $N$ examples.}
2:  Let $k$ be the number of boosting rounds.
3:  for $i = 1$ to $k$ do
4:     Create training set $D_i$ by sampling (with replacement) from $D$ according to $\mathbf{w}$.
5:     Train a base classifier $C_i$ on $D_i$.
6:     Apply $C_i$ to all examples in the original training set, $D$.
7:     $\epsilon_i = \frac{1}{N}\left[\sum_j w_j\, \delta(C_i(x_j) \neq y_j)\right]$   {Calculate the weighted error.}
8:     if $\epsilon_i > 0.5$ then
9:        $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$.   {Reset the weights for all $N$ examples.}
10:       Go back to Step 4.
11:    end if
12:    $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13:    Update the weight of each example according to Equation 6.103.
14: end for
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.
```

Adjust weights

Error rate

$$\epsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} w_j\ I\Big(C_i(\mathbf{x}_j) \neq y_j\Big)\right],$$

Importance of classifier

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right).$$

Weight adjustment

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} e^{-\alpha_j} & \text{if } C_j(\mathbf{x_i}) = y_i \\ e^{\alpha_j} & \text{if } C_j(\mathbf{x_i}) \neq y_i \end{cases},$$

10

4. A

Instance	Actual Label	Majority Vote	Correct ?
1	1	1	1
2	0	0	1
3	1	1	0

5

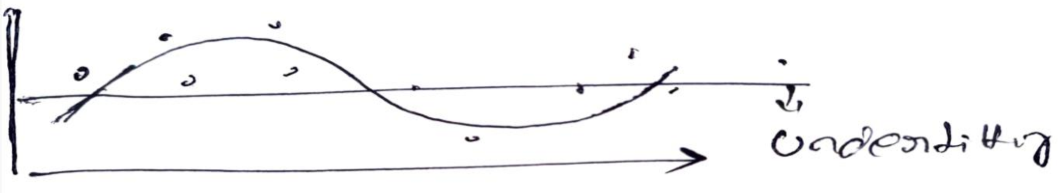
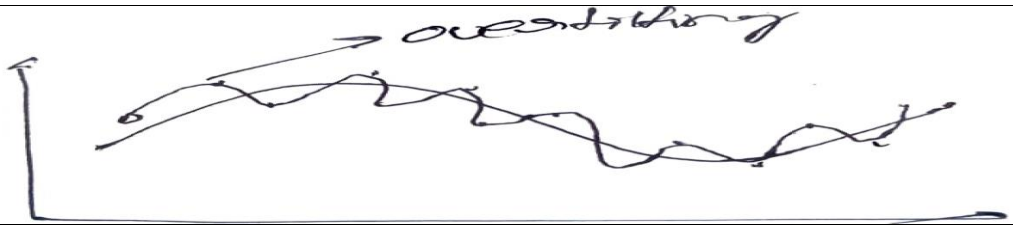
4	0	0	0
5	1	1	1

Ensemble Accuracy = Correct Predictions/Total Instances=5/5=100%

B	**Cluster Analysis?** Cluster analysis is a technique to group data points into clusters where points in the same cluster are similar, and points in different clusters are dissimilar. **Types of Clustering Techniques** 1. **Partitioning Clustering**: Divides data into non-overlapping clusters (e.g., K-means). 2. **Hierarchical Clustering**: Builds a tree of clusters using agglomerative or divisive approaches. 3. **Density-Based Clustering**: Forms clusters based on dense regions of data (e.g., DBSCAN). 4. **Grid-Based Clustering**: Divides data into grids and clusters cells with dense data (e.g., STING).	5
5. A	**Steps of the Algorithm:** 1. **Initialize Centroids**: Randomly choose k initial cluster centroids. 2. **Assign Points to Clusters**: Assign each data point to the nearest centroid. 3. **Update Centroids**: Recalculate centroids as the mean of all points in a cluster. 4. **Repeat**: Reassign points and update centroids until convergence (no changes in cluster assignments). One Numerical Example	**05**
B	**1. Cohesion (Intra-cluster Compactness)** • **Definition**: Measures how closely related the data points within the same cluster are. • **Formula**: Cohesion = $\sum_{i=1}^{k} \sum_{x \in C_i} distance\ (x, centroid\ (C_i))$ • **Goal**: Minimize cohesion to ensure data points within a cluster are similar. **2. Separation (Inter-cluster Distinction)** • **Definition**: Measures how well-separated different clusters are from each other. • **Formula**: Separation = $\sum_{i=1}^{k} \sum_{j=i+1}^{k} distance(centroid\ of\ C_i\ , centroid\ of\ C_j)$ • **Goal**: Maximize separation to ensure clusters are distinct from one another. **Combined Evaluation** The quality of clustering improves when cohesion is minimized, and separation is maximized. Metrics like the **Silhouette Coefficient** combine these concepts for practical evaluation.	**05**

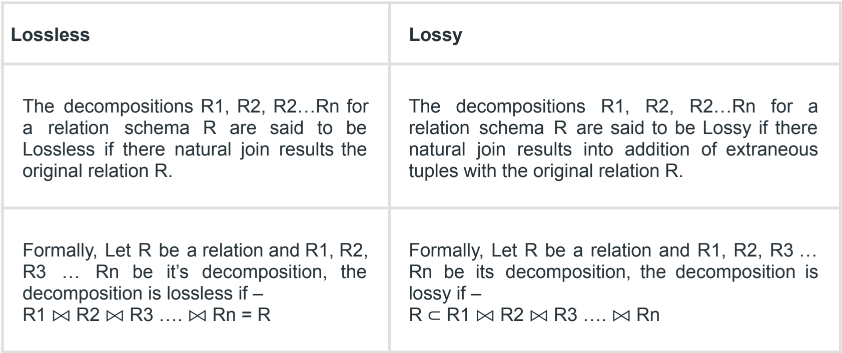
Academic year 2024-2025 (Odd Sem)

QUIZ

Q. No.	Questions	M
1.		2
2.		2
3.	Cohesion (C_i) = $\sum_{\substack{x \in C_i \\ y \in C_i}} proximity\ (x, y)$ Separation (C_i, C_j) = $\sum_{\substack{x \in C_i \\ y \in C_j}} proximity\ (x, y)$	2
4.	Parallel, Serial	2
5.	The performance of Random Forests is reduced as the fraction of irrelevant features increases because the irrelevant features introduce noise and reduce the quality of splits in the decision trees.	2

****

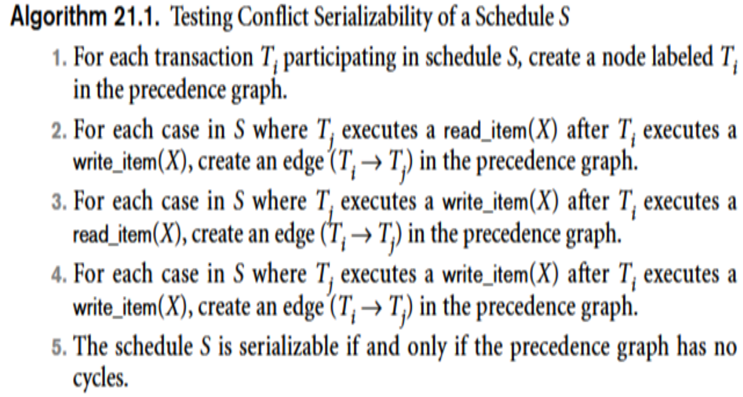
<table>
<tr><td colspan="2"></td><td colspan="4"><b>R V College of Engineering
Department of Computer Science and Engineering
CIE - III(Improvement): Scheme</b></td></tr>
<tr><td colspan="2"><b>Subject :
(Code)</b></td><td><b>Database Management Systems (CD252IA)</b></td><td colspan="3"><b>Semester : 5^{TH} BE</b></td></tr>
<tr><td>S.N</td><td colspan="2"><b>PART-A</b></td><td><b>M</b></td><td><b>BT</b></td><td><b>Co</b></td></tr>
<tr><td>1.</td><td colspan="2">What is the difference between lossless and lossy decomposition in DBMS?



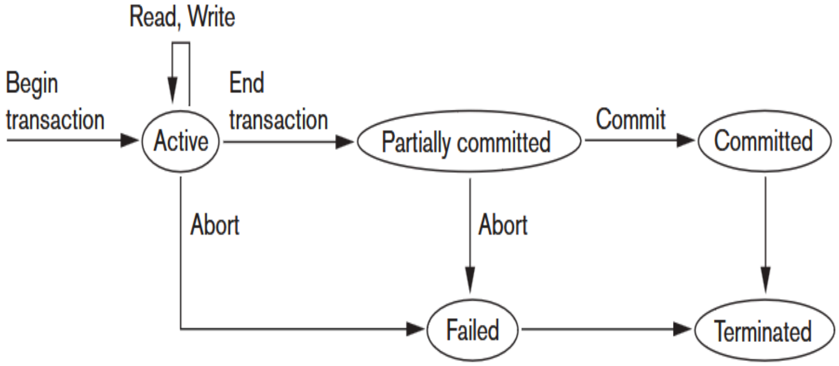
<table>
<tr><th>Lossless</th><th>Lossy</th></tr>
<tr><td>The decompositions R1, R2, R2…Rn for a relation schema R are said to be Lossless if there natural join results the original relation R.</td><td>The decompositions R1, R2, R2…Rn for a relation schema R are said to be Lossy if there natural join results into addition of extraneous tuples with the original relation R.</td></tr>
<tr><td>Formally, Let R be a relation and R1, R2, R3 … Rn be it’s decomposition, the decomposition is lossless if –
R1 ⋈ R2 ⋈ R3 …. ⋈ Rn = R</td><td>Formally, Let R be a relation and R1, R2, R3 … Rn be its decomposition, the decomposition is lossy if –
R ⊂ R1 ⋈ R2 ⋈ R3 …. ⋈ Rn</td></tr>
</table>
</td><td>2</td><td>L2</td><td>3</td></tr>
<tr><td>2.</td><td colspan="2">List the two conditions for checking the Binary decomposition?
▪ The FD ((R1 ∩ R2) → (R1 – R2)) is in F+,
▪ The FD ((R1 ∩ R2) → (R2 – R1)) is in F+</td><td>2</td><td>L1</td><td>2</td></tr>
<tr><td>3.</td><td colspan="2">Define the Condition of 3NF?
Third normal form (3NF) is based on the concept of transitive dependency. A functional dependency X → Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold.</td><td>2</td><td>L1</td><td>2</td></tr>
<tr><td>4.</td><td colspan="2">Define a Transaction with example.
A transaction is an executing program that forms a logical unit of database processing. A transaction includes one or more database access operations—these caninclude insertion, deletion, modification, or retrieval operations.
Ex: airline reservation systems</td><td>2</td><td>L1</td><td>1</td></tr>
<tr><td>5.</td><td colspan="2">Elaborate and Define ACID properties:
Atomicity, Consistency,Isolation, Durability----0.5m each</td><td>2</td><td>L1</td><td>1</td></tr>
<tr><td></td><td colspan="2"><b>PART-B</b></td><td></td><td></td><td></td></tr>
<tr><td>1a</td><td colspan="2">Discuss the condition for two functional dependencies to be equivalent? Check whether relation R(A,B,C,D) having two FD sets FD1 = {A->B, B->C, AB->D} and FD2 = {A->B, B->C, A->C, A->D} are equivalent or not ?

Condition: Two sets of functional dependencies E and F are equivalent if E+ = F+. Therefore, equivalence means that every FD in E can be inferred from F, and every FD in F can be inferred from E; that is, E is equivalent to F if both the conditions—E covers F and F covers E—hold---1m
Step 1: Checking whether all FDs of FD1 are present in FD2
● A->B in set FD1 is present in set FD2.
● B->C in set FD1 is also present in set FD2.</td><td>5</td><td>L3</td><td>2</td></tr>
</table>

	● AB->D is present in set FD1 but not directly in FD2 but we will check whether we can derive it or not. For set FD2, (AB)+ = {A, B, C, D}. It means that AB can functionally determine A, B, C, and D. So AB->D will also hold in set FD2. As all FDs in set FD1 also hold in set FD2, FD2 ⊃ FD1 is true. Step 2: Checking whether all FDs of FD2 are present in FD1 ● A->B in set FD2 is present in set FD1. ● B->C in set FD2 is also present in set FD1. ● A->C is present in FD2 but not directly in FD1 but we will check whether we can derive it or not. For set FD1, (A)+ = {A, B, C, D}. It means that A can functionally determine A, B, C, and D. SO A->C will also hold in set FD1. ● A->D is present in FD2 but not directly in FD1 but we will check whether we can derive it or not. For set FD1, (A)+ = {A, B, C, D}. It means that A can functionally determine A, B, C, and D. SO A->D will also hold in set FD1. As all FDs in set FD2 also hold in set FD1, FD1 ⊃ FD2 is true. As FD2 ⊃ FD1 and FD1 ⊃ FD2 both are true **FD2 =FD1** is true. These two FD sets are semantically equivalent. -----4m			
1b	Explain any 5 reasons for failure of transaction. A transaction or system error Local errors or exception conditions detected by the transaction, A computer failure (system crash). Concurrency control enforcement. Disk failure. Physical problems and catastrophes.-----any 5 from above -----1m each	5	L2	
2a	Explain the steps for finding Minimal Cover for Functional Dependencies. For the given set of FDs {A->C, AC->D, E->H, E->AD} find the minimal cover. ● Steps: we define a set of functional dependencies F to be minimal if it satisfies the following conditions: ● 1. Split the right-hand attributes of all FDs:Every dependency in F has a single attribute for its right-hand side. ● 2. Remove all redundant FDs. ● 3. Find the Extraneous attribute and remove it.. ● Example: Minimize {A->C, AC->D, E->H, E->AD} ● **Step 1**: {A->C, AC->D, E->H, E->A, E->D} ● **Step2**:{A->C,AC->D,E->H,E->A} Here Redundant FD : {E->D} ● **Step3**:{AC->D}{A}+={A,C} Therefore C is extraneous and is removed. {A->D} ● Minimal Cover = {A->C, A->D, E->H, E->A}	7	L3	2
2b	Write the algorithm for Testing whether a schedule is serializable or not.	3		

<table>
<tr><td></td><td>
<b>Algorithm 21.1.</b> Testing Conflict Serializability of a Schedule S
1. For each transaction T_i participating in schedule S, create a node labeled T_i in the precedence graph.
2. For each case in S where T_j executes a read_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
3. For each case in S where T_j executes a write_item(X) after T_i executes a read_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
4. For each case in S where T_j executes a write_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
5. The schedule S is serializable if and only if the precedence graph has no cycles.</td><td></td><td></td><td></td></tr>
<tr><td>3a</td><td>Explain the properties of Attribute preservation and dependency preservation?
Attribute preservation---2.5m
Dependency preservation-2.5m</td><td>5</td><td>L2</td><td>3</td></tr>
<tr><td>3b</td><td>Given a relational schema R = { SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS } and the decomposed table R1 = { ENAME, PLOCATION } and R2 = { SSN, PNUMBER, HOURS, PNAME, PLOCATION } and FD = { SSN → ENAME, PNUMBER → { PNAME, PLOCATION}, { SSN, PNUMBER } → HOURS }. Identify whether the given decomposition of R, R1 and R2 is lossless or lossy decomposition ?
Matrix:
<table>
<tr><th></th><th>SSN</th><th>ENAME</th><th>PNUMBER</th><th>PNAME</th><th>PLOCATION</th><th>HOURS</th></tr>
<tr><td>R1</td><td></td><td>a</td><td></td><td></td><td>a</td><td></td></tr>
<tr><td>R2</td><td>a</td><td></td><td>a</td><td>a</td><td>a</td><td>a</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
Final matrix
<table>
<tr><th></th><th>SSN</th><th>ENAME</th><th>PNUMBER</th><th>PNAME</th><th>PLOCATION</th><th>HOURS</th></tr>
<tr><td>R1</td><td>b</td><td>a</td><td>b</td><td>b</td><td>a</td><td>b</td></tr>
<tr><td>R2</td><td>a</td><td>b</td><td>a</td><td>a</td><td>a</td><td>a</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
\
It’s a lossy decomposition</td><td>5</td><td>L3</td><td></td></tr>
<tr><td>4a</td><td>Given a relation R(A, B, C, D) and Functional Dependency set FD = { AB → CD, B → C }, determine whether the given R is in 2NF? If not convert it into 2 NF.
Solution: No non-prime attribute should be partially dependent on Candidate Key Since R has 4 attributes: - A, B, C, D, and Candidate Key is AB, Therefore, prime attributes (part of candidate key) are A and B while a non-prime attribute are C and D
a) FD: <b>AB → CD</b> satisfies the definition of 2NF, that non-prime attribute(C and D) are fully dependent on candidate key AB
b) FD: <b>B → C</b> does not satisfy the definition of 2NF, as a non-prime attribute(C) is partially dependent on candidate key AB(i.e. key should not be broken at any cost)
<b>As FD B → C, the above table R(A, B, C, D) is not in 2NF---3M</b>

<b>Conversion to 2NF:</b>
a) R1(B, C)
b) R2(A, B, D)-----2M</td><td>5</td><td>L3</td><td>1</td></tr>
</table>

b	With a transition diagram explain the states for transaction execution  Read, Write Begin transaction Active End transaction Partially committed Commit Committed Abort Abort Failed Terminated Diag-2m Expl-3m	5	L2	2
5.	List and explain with examples the types of problems that can be encountered if two transactions are executing concurrently. ● The Lost Update Problem. ● The Temporary Update (or Dirty Read) Problem. ● The Incorrect Summary Problem. ● The Unrepeatable Read Problem. List -2m Explanation- 8m	10	L2	1

CIE-II

Theory of computation (CS354TA)

1.1 Definition of left recursion - 1M
Elimination of left recursion - 1M

$S \rightarrow (L) \mid a$

~~S'~~ $L' \rightarrow , SL' \mid \epsilon$, $L = SL'$

1.2 CFG to PDA.

(q_1, B)

$\delta(q_0, \epsilon, Z) = (q_1, SZ)$, $\delta(q_1, \epsilon, S) = \{(q_1, a), (q_1, aA)$

$\delta(q_1, \epsilon, A) = \{(q_1, aB), (q_1, \epsilon)\}$, $\delta(q_1, \epsilon, B) = \{q_1, Aa)$

1.3 $S \rightarrow aSbb \mid A \mid B$. 1.4. one 1.5 $L = R = P$.

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$.

1.6 A PDA is deterministic iff

a) $\delta(q, a, x)$ has atmost one member for any
$a \in \{\Sigma \cup \epsilon\}$ — 1M

b) If $\delta(q, a, x)$ is non empty for some $a \in \Sigma$, then $\delta(q, \epsilon, x$
must be empty. - 1M

2a) PDA → 3 M , Graphical representation - 2M, ID-2

$\delta(q_0, a, 2) = (q_0, a2)$, $\delta(q_0, a, b) = (q_0, \epsilon)$ $\delta(q_0, b, a) = (q_0, b$

$\delta(q_0, b, 2) = (q_0, b2)$, $\delta(q_0, a, c) = (q_0, \epsilon)$ $\delta(q_0, b, b) = (q_0, bb$

$\delta(q_0, c, 2) = (q_0, c2)$, $\delta(q_0, a, a) = (q_0, aa)$ $\delta(q_0, b, c) = (q_0, \epsilon$

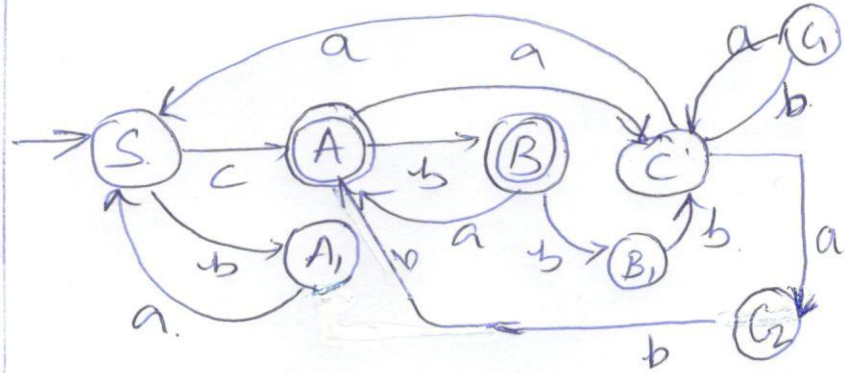
$\delta(q_0, \epsilon, 2) = (q_f, 2)$

$\delta(q_0, c, a) = (q_0, \epsilon)$

$\delta(q_0, c, b) = (q_0, cb)$

$\delta(q_0, c, c) = (q_0, cc)$

2b)



3a) (i) Reverse the automata. — 2M

(ii) Obtain right linear grammar - 2M

(iii) Reverse productions of right linear grammar - 2M.

3b) (i) Languages accepted by final state - 2M

(ii) Languages accepted by empty stack - 2M.

4a) Definition of CNF. — 2M

(i) NO useless production.

(ii) A and B are nullable variable.

$S \rightarrow abAB \mid abB \mid abA \mid ab$

$A \rightarrow a \mid Aa$.

$B \rightarrow b$.

} 2M.

(iii) No Unit productions.

Converting to CNF - 2M.

$S \rightarrow A_1A_2 \mid A_1B \mid A_1A \mid A_3B$.

$A_2 \rightarrow AB$, $A \rightarrow a \mid AA_3$.

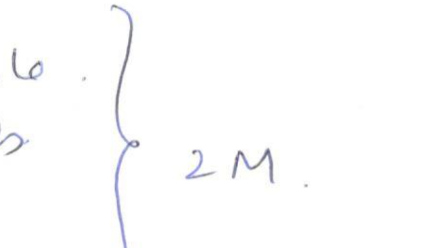
$A_1 \rightarrow A_3B$. $B \rightarrow b$.

$A_3 \rightarrow a$.

4b) Definition of ambiguity - 1M.

Proving for ambiguity - 2M.

Unambiguous grammar - 1M.



LMD1 $S \rightarrow aaB$

$S \rightarrow aab$

LMD2

$S \rightarrow AB$

$A \rightarrow AaB$

$A \rightarrow aaB$

$A \rightarrow aab$.

2 parse trees hence ambiguous.

Unambiguous grammar

$S \rightarrow AB$

$A \rightarrow a \mid Aa$. $B \rightarrow b$.

5a) Algorithm — 2 M.

Conversion — 7M.

Language accepted — 1M $L = \{ n_0(w) = n_1(w) \mid w \in (0,1)$

Start symbol. $q_0 Z q_1$ — 1M.

$\delta(q_0, 0, Z) = (q_0, AZ)$. $4 * 1 = 4$ M.

$q_0 Z q_0 \rightarrow 0 (q_0 A q_0)(q_0 Z q_0) \mid 0 (q_0 A q_1)(q_1 Z q_0)$

$q_0 Z q_1 \rightarrow 0 (q_0 A q_0)(q_0 Z q_1) \mid 0 (q_0 A q_1)(q_1 Z q_1)$

|||ly for other transitions. writing CFG.

$\delta(q_0, 0, B) = (q_0, \epsilon)$ $\quad q_0 B q_0 \rightarrow 0$

$\delta(q_0, 1, A) = (q_0, \epsilon)$ $\quad q_0 A q_0 \rightarrow 1$ $\quad \}$ 2M.

$\delta(q_0, \epsilon, Z) = (q_1, \epsilon)$ $\quad q_0 Z q_1 \rightarrow \epsilon$.

6a) CFG for L_1 $\quad S_1 \rightarrow a S_1 b \mid \epsilon$. $\quad$ CFG for L_2 $\quad S_4 \rightarrow S_5 S_6$.

$S_0 \rightarrow S_1 S_2$. $\quad S_2 \rightarrow Cc \mid \epsilon$. $\quad S_5 \rightarrow a S_7 \mid \epsilon$

└ 1.5M $\quad S_6 \rightarrow b S_6 c \mid \epsilon$.

└ 1.5M.

This proves that L1 & L2 are CFLs

L_3 $\quad S \rightarrow S_0 \mid S_4$ — 1M.

Proof for $L_1 \cap L_2 \notin$ CFL — 2M.

6b) Applications of CFGs.

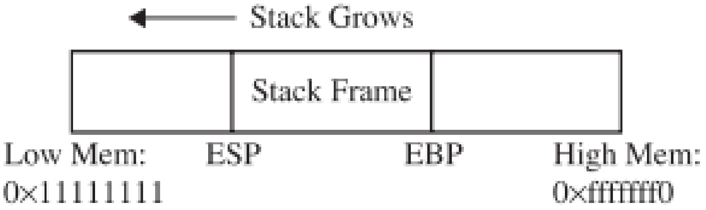
(i) In compiler design during parsing for syntactic checking.

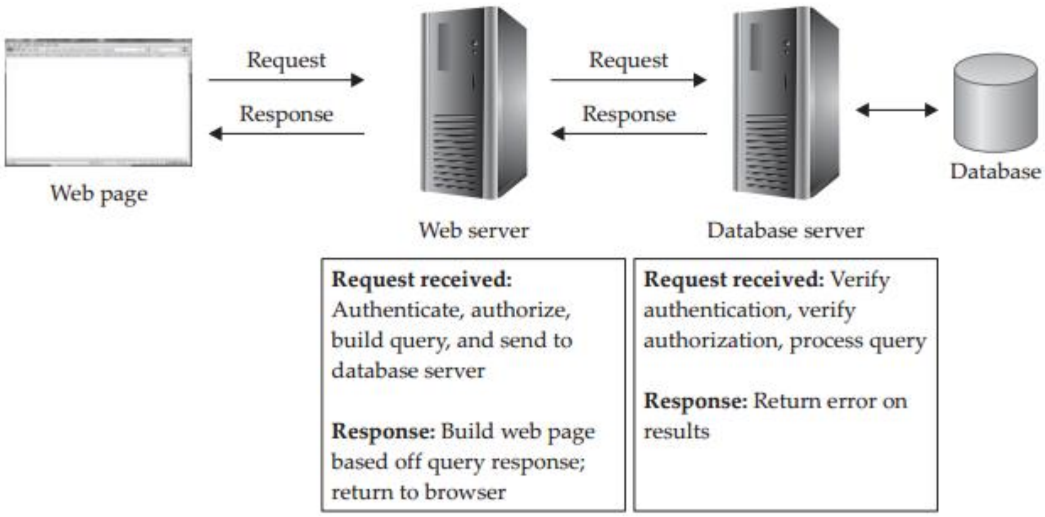
(ii) XML and Document-type definition.

(iii) Markup languages.

	RV College of Engineering® **Department of Computer Science and Engineering** **CIE - III**		
Course & Code	**Vulnerability Assessment & Penetration Testing (CY255TBD)** **Professional Core Course Elective-I**		**Semester: V**
Date: 29/01/2025	**Duration:**120 minutes	**Max.Marks** : 10 Marks (Quiz) + 50 Marks (Test)	
Scheme and Solution			

Sl.no.	**Part A Quiz**	**Marks**
1	Scope	01
2	Stack	01
3	Payload creation	01
4	Cross-site scripting (XSS)	01
5	Server-side	01
6	Passive	01
7	Ghidra	01
8	Codebase	01
9	A buffer overflow exploit works by overwriting adjacent memory (e.g., the return address) with malicious data, enabling an attacker to execute arbitrary code.	01
10	Injection vulnerabilities.	01

Sl.no.	Part B Test	Marks
1.a	Definition of buffer **overflow** and explanation for how it occurs. stack operations: **push** and **pop**, and their importance in memory management. Buffer overflow exploits manipulate the stack, such as overwriting the return address. Example or diagram of a buffer overflow on the stack. Implications of buffer overflows and measures to prevent them (e.g., stack canaries, address space randomization).  Stack Grows Stack Frame Low Mem: 0×11111111 ESP EBP High Mem: 0×fffffff0	06
1.b	○ **Passive Analysis:** ▪ **Advantages:** 1. Non-intrusive and stealthy. 2. Ideal for monitoring systems in real-time or long-term. ▪ **Limitations:** 1. May miss vulnerabilities that require active exploitation. 2. Limited in scope and may not cover all potential attack vectors. ▪ **Use Cases:** 1. Continuous monitoring of networks for suspicious behavior. 2. Detecting passive vulnerabilities like weak encryption or misconfigured security settings. ○ **Source Code Analysis:** ▪ **Advantages:** 1. Allows early detection of vulnerabilities during the development phase. 2. Comprehensive review of all code, ensuring thorough vulnerability assessment. ▪ **Limitations:** 1. Requires access to the source code, which may not always be available (e.g., closed-source software). 2. May generate false positives or require manual validation. ▪ **Use Cases:** 1. Reviewing application code for security flaws before deployment. 2. Identifying insecure coding practices and logic errors. ○ **Binary Analysis:**	06

	▪ **Advantages:** 1. Can be applied to closed-source software (no access to source code required). 2. Useful for detecting vulnerabilities in compiled code that may not be present in source code. ▪ **Limitations:** 1. Requires expertise in reverse engineering and understanding machine-level code. 2. May not identify all vulnerabilities, especially in highly obfuscated or protected binaries. ▪ **Use Cases:** 1. Analyzing third-party software or closed-source applications for vulnerabilities.	
2.	 Web page Request Response Web server Request Response Database server Database **Request received:** Authenticate, authorize, build query, and send to database server **Response:** Build web page based off query response; return to browser **Request received:** Verify authentication, verify authorization, process query **Response:** Return error on results The OWASP Top Ten is a list of the most critical security risks for web applications, compiled by the Open Web Application Security Project (OWASP). These vulnerabilities represent common and impactful security flaws that developers and organizations should address to ensure robust web application security. Addressing OWASP Vulnerabilities Enhances Application Security 1. **Prevention of Data Breaches**: Securing vulnerabilities like injection and broken access control prevents unauthorized data access or exposure. 2. **Increased User Trust**: A secure application builds trust among users, improving reputation and customer retention. 3. **Compliance with Standards**: Mitigating these risks helps organizations comply with regulations like GDPR, HIPAA, and PCI-DSS. 4. **Cost Reduction**: Early detection and remediation of vulnerabilities reduce costs associated with post-breach incident handling. 5. **Improved Application Reliability**: Secure systems experience fewer disruptions from cyberattacks, ensuring continuous service availability.	10
3	**Binary Analysis** is the process of inspecting and analyzing the compiled binary code of an application to identify potential security flaws. Unlike **source code analysis**, which examines human-readable code, **binary analysis** involves working with the compiled version of an application, which may not include the original source code. This method is particularly useful for security assessments when the source code is unavailable or when vulnerabilities exist at the compiled level.	10

Process of Binary Analysis

1. **Obtaining the Binary**:
   The first step in binary analysis is to acquire the compiled binary of the application. This can be an executable (.exe, .elf) or a library (.dll, .so) that contains the program's logic. For web applications, it could also involve analyzing compiled scripts like those written in Java or .NET.
2. **Disassembly**:
   The next step is to **disassemble** the binary. Disassembly converts machine code (or bytecode) back into assembly language, which is more readable to humans. This allows security analysts to examine the structure of the program, including functions, calls, and memory handling.
3. **Reverse Engineering**:
   After disassembly, **reverse engineering** is used to interpret the logic of the program. Analysts attempt to reconstruct the original program logic, identify critical operations, and locate areas where vulnerabilities could be present (such as buffer overflows or improper memory handling). Reverse engineering may involve identifying strings, functions, and system calls that can indicate potential vulnerabilities.
4. **Dynamic Analysis**:
   In addition to static disassembly, **dynamic analysis** is often performed. This involves running the binary in a controlled environment (such as a sandbox or virtual machine) and observing its behavior during execution. Analysts monitor runtime behavior, input validation, system calls, and network communication to detect anomalies, vulnerabilities, or malicious actions.
5. **Vulnerability Identification**:
   During binary analysis, analysts look for common vulnerabilities such as:
   - **Buffer Overflows**: Identifying areas where input data can overflow and overwrite memory.
   - **Use-After-Free**: Detecting instances where memory is used after it has been freed, leading to potential exploits.
   - **Code Injection**: Looking for areas where an attacker might inject malicious code.
   - **Privilege Escalation**: Finding ways the binary might allow unauthorized users to gain higher privileges.
   - **Uninitialized Memory**: Identifying areas where memory is not initialized properly, which can lead to unpredictable behavior.
6. **Patching and Recommendations**:
   After detecting vulnerabilities, the next step is to **recommend fixes**. These might involve altering memory handling, using safer libraries, implementing proper bounds checking, or other methods to mitigate identified risks.

Detecting Security Flaws in a Compiled Application

Binary analysis helps in detecting flaws that may not be visible through source code inspection. Key methods used to detect security flaws in a compiled application include:

1. **Memory Corruption Vulnerabilities**:
   Analyzing how the application handles memory helps to detect issues like buffer overflows, heap corruption, and use-after-free errors that can be exploited for arbitrary code execution.
2. **Control Flow Hijacking**:
   Attackers can manipulate the control flow of a program, for instance, by exploiting stack-based buffer overflows to overwrite return addresses. By analyzing how the program's control flow is structured, these vulnerabilities can be identified.
3. **Improper Input Handling**:
   Binary analysis also detects vulnerabilities related to poor input validation, which can lead to SQL injection, command injection, and other similar attacks.
4. **Exploiting Third-Party Libraries**:
   Compiled applications often rely on external libraries, which may contain

vulnerabilities. Binary analysis can help identify issues in these libraries, such as known exploits or misconfigurations.

Commonly Used Tools for Binary Analysis

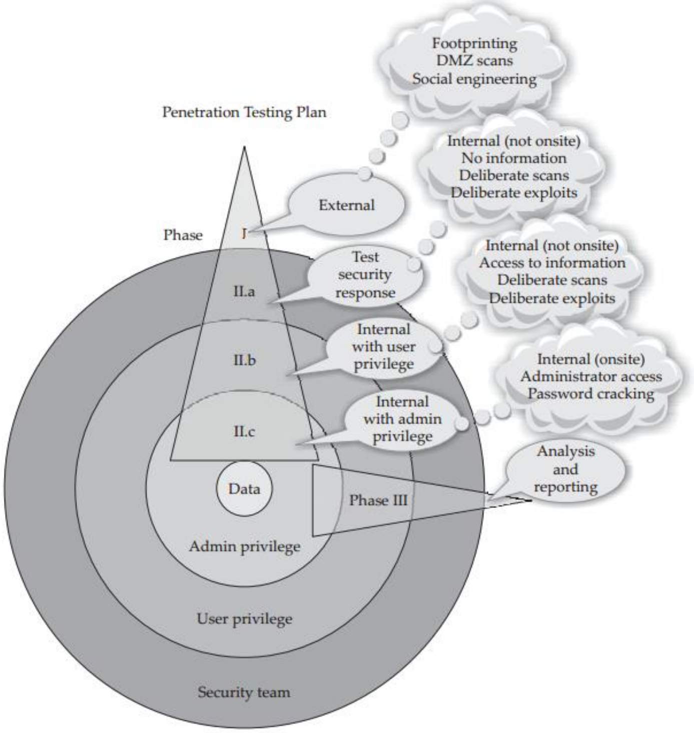
Several tools are employed in binary analysis to automate and facilitate the process of inspecting compiled applications. These tools help reverse engineers and security professionals to disassemble, debug, and analyze binaries for security flaws.

1. **IDA Pro (Interactive Disassembler)**:
   IDA Pro is one of the most popular disassemblers and debuggers, allowing analysts to reverse engineer a binary into a human-readable assembly language. It also offers features like function graphing and debugging capabilities.
2. **Ghidra**:
   Ghidra is a free, open-source reverse engineering tool developed by the NSA. It provides powerful disassembling, decompiling, and debugging capabilities for analyzing binaries.
3. **Radare2**:
   Radare2 is an open-source framework for reverse engineering, offering a wide range of tools for disassembly, debugging, and exploiting vulnerabilities in binaries.
4. **OllyDbg**:
   OllyDbg is a popular 32-bit debugger used to analyze the runtime behavior of a program. It's useful for dynamic analysis, such as detecting buffer overflows and code injection.
5. **Binary Ninja**:
   Binary Ninja is a reverse engineering platform that automates parts of the analysis, such as identifying vulnerabilities like control-flow hijacking or memory corruption.
6. **Frida**:
   Frida is a dynamic instrumentation toolkit that allows analysts to interact with a running process, enabling the monitoring of function calls, system calls, and memory regions in real time.

4

List of three main phases along with sub phases, Diagram- 4 marks, brief explanation- 4 marks

08



5.a	Dradis Server: The Dradis framework is an open source system for information sharing Scope of a Penetration Test *03 marks each*	06
5.b	○ **Server-Side Defenses:** 1. **Input Validation and Sanitization:** Sanitize user inputs by removing dangerous characters like <, >, and `&`, ensuring they can't be executed as code. 2. **Output Encoding:** Encode data before rendering it in the browser, ensuring that potentially harmful characters are displayed safely. For example, encoding < as `&lt;` in HTML output. 3. **Content Security Policy (CSP):** Implement a CSP to restrict the types of content that can be loaded, preventing the execution of malicious scripts. ○ **Client-Side Defenses:** 1. **HTTP-Only and Secure Cookies:** Ensure cookies are marked as `HTTPOnly` and `Secure` to prevent JavaScript from accessing sensitive data. 2. **JavaScript Frameworks:** Use secure frameworks like AngularJS and React, which automatically escape user inputs and protect against XSS attacks. 3. **Subresource Integrity (SRI):** Implement SRI to ensure that external scripts haven’t been tampered with.	04