

**Revision – R-2026 (Suggestive Courses for 1st Year MCA)
Course Structure**

Induction Program													
Sl. No.	Course Code	Course Title	Credit Distribution					Assessment					Teaching Department (TD) and Question Paper and Setting Board (PSB)
			Lecture	Tutorial	Practical	Self-Learning	Examination in Hours	Formative Assessment		Summative Assessment	Total Marks	Credits	
	Theory	Practical											
								L	T				
1	26IND001	Orientation Session (Induction Program)	Three Days Programme					100		-	100	1	CA

I - Semester

Sl. No.	Course Code	Course Title	Teaching Hours per Week					Examination					Teaching Department (TD) and Question Paper Setting Board (PSB)
			Lecture	Tutorial	Practical	Self-Learning	Exam Duration in Hours	Formative Assessment		Summative Assessment	Total Marks	Credits	
			L	T	P	SL		Theory	Practical				
1	26MCA110	Mathematical Foundation for Computer Applications	2	1	0	2	2½	70		30	100	3	BSC
2	26MCA120	Data Structures & Applications	2	1	1	2	2½	70		30	100	4	CA
3	26MCA130	Python Programming	2	1	1	2	2½	70		30	100	4	CA
4	26MCA140	Web Stack Development through PHP	2	1	1	2	2½	70		30	100	4	CA
5	26MCA150	Unix & Shell Programming	2	1	1	2	2½	70		30	100	4	CA
6	26MCA160	Programming in C	2	1	0	2	2½	70	-	30	100	3	CA
7	26CEE170	Communicative English	0	0	1	1	-	100	-	-	100	1	CA
Total Marks											700	23	

Department of Computer Applications**II - Semester**

Sl. No.	Course Code	Course Title	Teaching Hours per Week					Examination					Teaching Department (TD) and Question Paper Setting Board (PSB)
			Lecture	Tutorial	Practical	Self-Learning	Duration in Hours	Formative Assessment		Summative Assessment	Total Marks	Credits	
			L	T	P	SL		Theory	Practical				
1	26MCA210	Database Management System	2	1	1	2	2½	70		30	100	4	CA
2	26MCA220	Object Oriented Programming with JAVA	2	1	1	2	2½	70		30	100	4	CA
3	26MCA230	Software Engineering & Project Management	2	1	1	2	2½	70		30	100	4	CA
4	26MCA24X	Professional Elective-1	2	1	-	2	2½	70		30	100	3	CA
5	26MBAOE20X	Open Elective-1	2	-	-	2	2½	70		30	100	3	MBA
6		Online Course (Swayam/Moocs)	2	-	-	-	1½	30	-	70	100	2	CA
7	26PWS260	Professional Writing Skills in English	1	-	2	1	-	70	-	30	100	2	H & S
Total Marks											700	22	

Professional Elective-1		Open Elective-1	
Course Code	Course Title	Course Code	Course Title
26MCA241	Data Communication	26MBAOE206	Financial Management & Investment
26MCA242	Foundation of Data Science	26MBAOE207	Supply Chain Management
26MCA243	Cyber Security	26MBAOE208	Entrepreneurship Development

**Abbreviations:**

BSC	Basic Science Course
CA	Computer Applications
ESC	Engineering Science Course
SEC	Skill Enhancement Course
H&S	Humanities & Science
B	Binary Graded Course
TD	Teaching Department
QPSB	Question Paper Setting Board
FA	Formative Assessment
SA	Summative Assessment
EH	Examination in Hours

Credit Definition:

Structure of Four Credit Course: Lecture (L), Tutorial (T) and Practice (P) are

L: T:P is 2:1:1 and

One Tutorial session = Two hours, One Practice session = Two hours

Two hours of Tutorial / Practice per week = 1Credit

Department of Computer Applications**Course Name: Mathematical Foundation for Computer Applications**

Course Code	26MCA110	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:0:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	(30:30:0:30)	Total Marks	100
Credits	03	Exam Hours	03

Pre-requisites:

1. Matrices
2. Basics of statistics
3. Basics of Graph Theory

Course objectives: Enable the students to

1. Solve system of linear equations
2. To model, analyze, and interpret data using curve fitting, correlation, and regression techniques for solving real-world problems.
3. Apply probability distribution in problems arising in different disciplines
4. Test the hypothesis using t-distribution and chi-square distribution.
5. Determine optimal prefix code

Module-1

	Page No.	From: 134 to 160 (Engineering Mathematics by Dr. K. S. C. Sudha Publications)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Topic : System of Linear Equations	Elementary row transformation of a matrix, Rank of a matrix, Consistency and solution of system of linear equations, Gauss elimination method, Gauss-Seidel method.					
Team work and self - learning	Assignment	1. Construct the real life problem into system of linear equations 2. Identify the nature of solution (unique, infinite or none) 3. Solve the equations				
	Seminar	1. Procedure of solving the system of linear equations				
	Self-learning	Cayley Hamilton theorem.				
	Applications	❖ Machine Learning & AI: Linear algebra is fundamental in algorithms like Principal Component Analysis (PCA), neural networks, and support vector machines.				

Department of Computer Applications

		❖ Computer Graphics: Used for transformations (scaling, rotation, translation) in 3D graphics and image processing.				
Module-2						
	Page No.	From: 183 to 238 (Complex Analysis, Probability & Statistics by Dr. K. S. C., Sudha Publications)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Topic: Curve Fitting, Correlation and Regression	Curve Fitting, Correlation and Regression: Principles of least squares, Curve fitting by the method of least squares: fitting a straight line in the form $y=ax+b$, fitting a parabola in the form $y=ax^2+bx+c$, and fitting a geometrical curve in the form of $y=ax^b$. Correlation, Co-efficient of correlation, Lines of regression, Rank correlation.					
Team work and self-learningP1-4	Assignment	1. Explain the importance of curve fitting in Artificial Intelligence and Machine Learning. 2. Differentiate between correlation and regression.				
	Seminar	1. Importance of correlation and regression in Software Engineering.				
	Self-learning	1. Fitting of curve in the form $y=ae^{bx}$ and $y=ab^x$.				
	Applications	Machine Learning Model Training: Fitting a curve to training data to minimize error in prediction models. Feature Selection in Machine Learning: Identifying features that are highly correlated with the target variable to improve model accuracy. Software Engineering: Estimating project effort or time using historical data (e.g., COCOMO models).				
Module-3						
	Page No.	From: 111 to 182 (Complex Analysis, Probability & Statistics by Dr. K. S. C., Sudha Publications)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Topic: Random Variables and Probability Distribution	Random Variables and Probability Distribution Concept of random variable, Discrete and continuous probability distributions, Binomial distribution, Poisson’s distribution, Mean and variance with proof, Problems.					

Department of Computer Applications

Team work and self-learning	Assignment	1. Conduct a small survey and write random variable values with frequencies 2. Calculate mean, variance and standard deviation 3. Write any 2 real-world examples each of discrete and continuous variables and justify.				
	Seminar	1. Procedure to calculate mean and variance using Binomial and Poisson distribution.				
	Applications	Simulation and Modeling Decision Theory and Game Theory Machine Learning and Artificial Intelligence				
	Self-learning	1. Joint Probability Distribution				
Module-4						
	Page No.	From: 262 to 299 (Complex Analysis, Probability & Statistics by Dr. K. S. C., Sudha Publications)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Topic: Sampling Theory:	Sampling Theory: Introduction to sampling Theory, type-1, type-II errors, Test of hypothesis - Student's t-distribution and Chi-square distribution as a test of goodness of fit.					
Team work and self-learning	Assignment	1. Identify the real world problems of hypothesis testing. 2. Apply the suitable method in the problem. 3. Justify the conclusion				
	Seminar	1. Explain the steps involved in hypothesis testing.				
	Self-learning	Point estimation and interval estimation				
	Applications	Machine Learning, Data Analysis, Network Security, Quality Control				
Module-5						
	Page No.	From: 415 to 459 (Discrete Mathematical Structures by Dr. D. S. C. Prism Books Pvt. Ltd.)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Topic: Trees	Trees: Definition of a tree, Properties of trees, minimally connected graphs. Theorems on Trees (with proof) such as. “A tree with n vertices has (n-1) edges”. “Any connected graph with n vertices and (n-1) edges is a tree”. “A connected graph is a tree if and only if it is minimally connected” and Examples, Routed trees, Weighted Trees and Prefix Codes.					

Department of Computer Applications

Team work and self-learning	Assignment	1. Understand the properties of trees 2. Write proofs of theorems 3. Construct Huffman tree step by step
	Seminar	Procedure of finding an optimal prefix code
	Self-learning	Sorting technique
	Applications	Hierarchical Data Representation Search engine XML/HTML/JSON Parsing Tries (Prefix Trees)

Course Outcomes: At the end of the course, student will be able to**CO-1:** Apply matrix theory for solving system of linear equations and compute eigen values and eigen vectors.**CO-2:** Apply curve fitting, correlation and regression to build predictive models across various engineering and scientific fields.**CO-3:** Apply binomial, poisson and normal distributions in problems arising in different disciplines.**CO-4:** Apply sampling methods, to formulate hypotheses, and perform appropriate statistical tests to support data-driven decision-making**CO-5:** Apply prefix codes and determine optimal prefix code.**Text Books and References:**

1. Kenneth H Rosen, Discrete Mathematics and its Applications, Tata Mcgraw-Hill, 7th edition, 2012.
2. Narsingh Deo, Graph Theory: With Application to Engineering and Computer Science, Prentice Hall of India, 16th, Edition.2003.
3. B S Grewal, Higher Engineering Mathematics, Khanna Publications, 44th edition, 2021.
4. Dr. D.S. Chandrasekharaiah, Discrete Mathematical structures and Graph theory, Prism Books Pvt Ltd, 6th Edition , 2019.
5. Dr. K. S. Chandrasekharaiah, Engineering Mathematics Sudha Publications.

E-References:

<https://nptel.ac.in/>
<https://epgp.inflibnet.ac.in/>
<https://www.coursera.org/>

Department of Computer Applications**DATA STRUCTURES AND APPLICATIONS**

Course Code	26MCA120	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:2:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	120 (30:30:30:30)	Total Marks	100
Credits	04	Exam Hours	2 ½

Pre-requisites:

1. Proficiency in C programming
2. Understanding of basic mathematics (arithmetic and logic)
3. Understanding of control statements, loops, functions, arrays, pointers, and basic problem-solving techniques
4. Basic understanding of time and space efficiency

Course Learning Objectives (CLOs):

1. Understand the fundamentals of data structures and their abstract models.
2. Apply algorithmic techniques for time and space-efficient problem-solving.
3. Analyze the performance of different data structures and algorithms.
4. Design programs using linear and non-linear data structures such as stacks, queues, lists, trees, and graphs.
5. Apply appropriate data structures for developing efficient solutions to real-world applications.

MODULE – I

Pedagogy	Page No.	From:1-30				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	Week	Hour		Hour		
	1	Algorithm Specification		Recursive Algorithms (Basics)		
	2	Recursive Algorithms (Advanced)		Data Abstraction & ADTs.		
	3	Performance Analysis (Time and Space Complexity		Arrays and Structures (Abstract Data Types, Sparse Matrix)		
Transactions	1	Search Optimization Scenario				
	2	Recursion in Real Life				
	3	Memory Efficiency Problem (Sparse Matrix)				
Practice	1	Why is binary search used in apps like phone contacts instead of linear search? Explain with reasoning.				
	2	How can recursion be used to navigate through nested folders in a computer system?				
	3	In Google Maps or image processing, why is it inefficient to store all values in a matrix? What is a better approach?				

MODULE – II

	Page No.	From: 101 to 132				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Stack ADT			Stack Implementation & Applications	
	2	Expression Conversion (Infix → Postfix/Prefix)			Understanding Maze Problem	
	3	Queue ADT			Multiple stacks and queues	
Transactions	1	Browser Back/Forward System (Stack)				
	2	Expression Evaluation in Calculators (Stack)				
	3	Food Delivery / Ride Queue System (Queue)				

Department of Computer Applications

Practice	1	Browser Navigation				
	2	Calculator Design				
	3	Call Center Handling System (Queue)				
MODULE – III						
	Page No.	From: 135 to 165				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Limitations of Arrays		Memory Management (Static vs Dynamic)		
	2	Singly Linked List		Doubly & Circular Linked Lists		
	3	Applications of Linked Lists		Linked Representation of Polynomials		
Transactions	1	Train Coach Linking System (Linked List)				
	2	Memory Fragmentation Concept (Dynamic Memory)				
	3	Ticket Booking Cancellation System (Linked List)				
Practice	1	Feed Optimization				
	2	Memory Usage				
	3	Real-Time Updates				
MODULE – IV						
	Page No.	From: 186 to 231				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Introduction to Trees & Binary Trees		Tree Representations & Traversals		
	2	Binary Search Trees (BST)		Threaded Binary Trees & Abstract Operations		
	3	Heaps		Priority Queues		
Transactions	1	File Folder Structure (Tree)				
	2	Online Shopping Search (BST)				
	3	Emergency Task Handling (Priority Queue)				
Practice	1	Why are trees more suitable than arrays for representing hierarchical data like file systems?				
	2	Why is BST preferred over linear structures for searching large datasets?				
	3	In which situations is a priority queue more useful than a normal queue? Give a real-world example.				
MODULE – V:						
	Page No.	From: 257 to 295				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Graph Basics & Graph ADT		Graph Representations		
	2	Breadth First Search (BFS)		Depth First Search (DFS)		
	3	Spanning Trees		Shortest Path Algorithms		
Transactions	1	Social Network Connections (Graph)				
	2	Navigation System (Shortest Path)				
	3	Web Crawling (BFS/DFS)				
Practice	1	Why are graphs more suitable than trees for representing social networks?				
	2	In a navigation system, why is BFS preferred for finding the shortest path in an unweighted graph?				

**Department of Computer Applications**

	3	How does a GPS system determine the shortest route between two locations?
Course Outcomes: CO-1. Analyze the efficiency of algorithms and apply appropriate data structures for problem-solving. CO-2. Implement linear data structures such as stacks, queues, and arrays. CO-3. Design programs using linked list-based structures for various applications. CO-4. Construct and traverse different types of trees and apply tree-based data structures. CO-5. Model and solve real-world problems using graph algorithms.		
Text Books: 1. Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed, Fundamentals of Data Structures in C, University Press. 2. Mark Allen Weiss Pearson, Data Structures and Algorithm Analysis in C, 4th Edition		
References 1. Gilberg, F Richard & Forouzan, A Behrouz, Data Structures A Pseudocode approach with C, 2nd Edition, Cengage		

Department of Computer Applications
PYTHON PROGRAMMING

Course Code	26MCA130	CIE Marks	70
Teaching Hours/Week (L:T:P:SL))	2:2:2:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	120 (30:30:30:30)	Total Marks	100
Credits	04	Exam Hours	2 ½

Pre-requisites: Basic knowledge of computer programming concepts and problem-solving skills.**Course Learning Objectives (CLOs)**

1. Understand the fundamentals of Python programming and problem solving.
2. Apply functions, data structures, and object-oriented concepts in Python programs.
3. Develop applications using file handling, regular expressions, and exception handling.
4. Analyze and visualize data using Python libraries such as NumPy, Pandas, and Matplotlib.
5. Build database, GUI, and web-based applications using Python technologies.

Unit-1

Pedagogy	Page No.	From: 1,7,35-58,67-73				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	Week	Hour		Hour		
	1	Python Introduction: History	Features,	Python Fundamentals: Identifiers, keywords		
	2	Statement and expression, Variables		Indentation, comments, Program output and input statement		
	3	Operators, associativity, Data types	Precedence and	Control Flow Statements: Conditional and Branching Statements		
Transactions	1	Programs on operators that implement real world problems				
	2	A presentation where students analyze the diffrence between static vs. dynamic typing in large-scale system development				
	3	Evaluate the impact of mandatory white-space on code readability and collaborative software engineering.				
Practice	1	Programs demonstrating valid/invalid identifiers, keyword usage, first Python program execution				
	2	Experiments on dynamic typing and building interactive scripts using formatted input and output statements.				
	3	programs on operator precedence and multi-level conditional branching to solve logical decision-making problems.				

Unit-2

	Page No.	Unit 2 From: 74-85, 95-112, 119-142, 149-169, 175-193, 201-221				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Looping statements, Loop control statements		Functions		
	2	Recursion & Lambda Functions		Data structures: List, Tuple, List comprehensions		
	3	Strings		Dictionaries, sets		
Transactions	1	Analysis of looping techniques				
	2	Discussion on text processing methods				
	3	Understanding dictionary and set operations				
	1	Programs focused on loop control (break/continue) and creating modular code using				

**Department of Computer Applications**

Practice		user-defined functions.				
	2	Hands-on practice with memory-efficient data storage using tuples and high-speed data generation through list comprehensions.				
	3	Projects involving text parsing and building unstructured data maps using dictionaries and unique-value sets.				
Unit-3						
	Page No.	From: 229-257, 267-274, 289-328				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Files		Regular expressions		
	2	Object-Oriented Programming – classes and objects, Constructors		Inheritance, Encapsulation		
	3	Polymorphism		Exception Handling – try/except/finally, custom exceptions		
Transactions	1	Design secure pattern-matching tools.				
	2	Defend a plan to protect sensitive data within code.				
	3	Design a fault-tolerant error-handling architecture using specialized exception hierarchies for specific system failures.				
Practice	1	Laboratory tasks on persistent storage, including reading/writing complex text patterns using regular expressions.				
	2	Case studies on blueprint modeling, focusing on real-world hierarchies through inheritance and data protection via encapsulation.				
	3	Programs demonstrating runtime flexibility through polymorphism and building fault-tolerant software using custom exception handling.				
Unit-4						
	Page No.	From: 341 - 380				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Introduction to data science		Functional Programming in python		
	2	JSON and XML in python		Numpy with Python		
	3	Pandas		Matplotlib		
Transactions	1	Study of structured vs. semi-structured data formats and how Python parses hierarchical data from web APIs.				
	2	Case studies on real-world datasets to understand the "Data frame" concept and indexing strategies.				
	3	presentations on "Visual Analytics"—how to choose the right graph (Bar, Scatter, Histogram) to tell a data story.				
Practice	1	Programs implementing functional paradigms to process and clean raw data sequences efficiently.				
	2	Laboratory tasks on numerical computing involving matrix manipulations and statistical operations on large arrays.				
	3	Projects on visual data representation, creating customized, publication-quality plots from analyzed data.				
Unit-5						
	Page No.	From:T2=397-393,397-421				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Database connectivity – sqlite3		CRUD operations with Python & SQLite		
	2	MySQL with Python		Tkinter Module: window and Widgets		
	3	Web Frameworks (FastAPI/Flask)		Introduction to Flask Applications		

Department of Computer Applications

Transactions	1	Designing normalized schemas with complex integrity constraints.
	2	Implement a Flask/FastAPI server that acts as a secure bridge between a web client and a MySQL database
	3	Designing GUI applications with Tkinter
Practice	1	Programs on transaction control and multi-table data operations.
	2	Developing a Web-based CRUD system connecting Python to a database.
	3	Building Tkinter interfaces that manage remote MySQL records.

Course Outcomes:

- CO-1. Develop Python programs using syntax, operators, and control statements.
- CO-2. Implement functions, data structures, and object-oriented concepts.
- CO-3. Design applications using files, regular expressions, and exception handling.
- CO-4. Perform data analysis and visualization using Python libraries.
- CO-5. Develop database, GUI, and web-based applications using Python.

TextBooks:

1. Text book 1: "Introduction to Python Programming", Gowrishankar S, Veena A
2. Python Crash Course A Hands On Project Based Introduction to Programming by Eric Matthes

References

1. Learning Python
2. Python Crash Course
3. Flask Web Development
4. Python for Data Analysis

Department of Computer Applications
UNIX AND SHELL PROGRAMMING

Course Code	26MCA150	Formative Assessment Lecture/ Practice	70
Teaching Hours/Week (L: T:P:SL)	2:2:2:2	Summative Assessment	30
Total Hours of Pedagogy (L: T:P:SL)	120 (30:30:30:30)	Total Marks	100
Credits	04	Exam Hours	2 ½

Pre-requisites:

1. Basic Computer Knowledge — understanding CPU, memory, and storage functions
2. Operating System Concepts — processes, memory management, and file system fundamentals
3. Command-Line Awareness — experience with CLI environments (Windows CMD, PowerShell, or equivalent)
4. Basic file handling — creating, deleting, and renaming files and folders

Course Objectives:

1. Evaluate the architecture, features, and operational design of Unix to assess its efficiency in multiuser and multitasking environments
2. Critically apply Unix file management, directory handling, and system commands to design solutions for complex administrative scenarios
3. Evaluate Unix file system architecture, access control mechanisms, and user management strategies for security and efficiency
4. Design and develop shell scripts that automate real-world tasks, integrating control structures, signal handling, and modular design patterns
5. Evaluate and manage process lifecycles, scheduling strategies, and system utilities to diagnose and resolve Unix environment issues

MODULE – I

Pedagogy	Page No.	From: 4 to 62				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	Week	Hour				
	1	• Unix Operating System — Evaluate the architectural layers: kernel, shell, and utilities in supporting multiuser, multitasking operations • POSIX and UNIX Specifications — Assess the significance of POSIX standards for system portability and compatibility across platforms				
	2	• Command Structure and Flexibility — Analyse internal vs. external commands, evaluate command execution pathways using which, whereis, and type				
	3	• General-Purpose Utilities — Design command sequences using cal, date, echo, printf, bc, script, passwd, who, uname, tty, and stty to solve real-world scenarios				
Transactions	1	• Evaluate the Design Philosophy and Operational Efficiency of the Unix Command-Line Interface				
	2	• Areas to critically analyze: Layered Unix architecture and its scalability merits POSIX standards — evaluating cross-system compatibility Internal vs. external command overhead — performance implications				
	3	• Pipes and redirection — evaluating data flow architectures CLI relevance in contemporary ecosystems (Linux, macOS, cloud servers) Design trade-offs in				

**Department of Computer Applications**

		Unix command flexibility				
Practice	1	- Evaluate Unix architectural components (kernel, shell, utilities) by executing uname and environment variable commands; justify each component's role based on observed output - Distinguish internal from external commands using which, whereis, and type; critically compare execution behavior and resolve observed discrepancies				
	2	- Design command pipelines incorporating I/O redirection; evaluate how pipes and redirection alter data flow versus standalone execution - Construct and assess formatted outputs using echo, printf, and bc; evaluate the precision and flexibility of each utility across different use cases				
	3	- Examine multiuser environment dynamics using who, tty, and stty; simulate and diagnose terminal misconfiguration scenarios using the script utility - Display logged-in users using who, identify terminal using tty, display terminal settings using stty, record a session using script and save it				
Self-Learning	1	Evaluate and classify commands as internal or external; justify classification criteria based on execution mechanism and shell behavior				
	2	- Design a comparative analysis of piped vs. direct command execution — measure and assess differences in performance and output integrity - Critically examine how I/O redirection strategies influence system resource usage and output management in scripts vs. interactive shells				
	3	Formulate a taxonomy of Unix utilities by function and evaluate their combined effectiveness in automating repetitive administrative tasks				
MODULE – II						
	Page No.	From: 64 to 116, From 212 to 220				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	File System Architecture, File Handling Commands				
	2	Compression and Archiving,				
	3	File Permissions and Security, File Discovery				
Transactions	1	You are a newly appointed system administrator at a university lab. Evaluate the current unstructured file system state and design an optimized directory hierarchy for student records (CSE, ECE, MECH departments).				
	2	Your submission must: (i) Critically evaluate the security implications of the existing permission structure, (ii) Design and implement a robust permission scheme using chmod and ownership settings, (iii) Create archiving and retrieval workflows using tar and gzip, (iv) Evaluate link strategies (hard vs. symbolic) for shared resources and justify recommendations				
	3	Deliverable: Implemented directory structure + annotated command log + justified permission policy document				
Practice	1	Design the prescribed ~/unix_lab/ directory structure; critically evaluate the efficiency of absolute vs. relative pathname navigation using pwd and ls variants				

**Department of Computer Applications**

		Perform and evaluate file operations (cp, mv, rm, cat, more) within dir1; assess the operational risks of irreversible deletions and propose safeguards				
	2	- Construct and critically analyze permission schemes using chmod and ls -l; evaluate directory-level access control implications and ownership change workflows - Compare file content divergence using cmp, diff, and comm on structured datasets; evaluate which tool is best suited for each comparison scenario and justify				
	3	- Design a compression and archiving pipeline using gzip, zip, and tar; evaluate storage efficiency and retrieval speed across tools - Create and evaluate hard links vs. symbolic links; critically assess link behavior after original file deletion and implications for system integrity				
Self-Learning	1	Design and evaluate a comparative study of UNIX file system organization vs. modern file systems (ext4, NTFS); assess structural strengths and limitations				
	2	- Critically examine how file permission models prevent or enable privilege escalation in multi-user environments - Evaluate the operational impact of hard link vs. symbolic link strategies on backup, recovery, and shared resource management				
	3	Formulate a production-grade file search strategy using find with compound criteria; evaluate precision vs. performance trade-offs				
MODULE – III						
	Page No.	From: 173 to 190, 120 to 140				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	Process Fundamentals, Process Creation Mechanisms				
	2	Job and Signal Control				
	3	Task Scheduling, vi Editor Mastery				
Transactions	1	A multiuser Unix system is degrading in performance due to unregulated process creation and inefficient text editing. Critically evaluate the root causes and design a remediation plan.				
	2	Your submission must: (i) Analyse running process states and distinguish user vs. system processes using ps, (ii) Design a job control and scheduling framework using at, batch, and cron, (iii) Evaluate priority management using nice and time with quantitative measurements, (iv) Create and evaluate a vi-based configuration editing protocol that minimises errors via structured search-and-replace workflows				
	3	Deliverable: Process analysis report + scheduling design + vi workflow documentation with justified rationale				
Practice	1	Evaluate process states using ps; critically distinguish system processes from user processes and assess each state's impact on resource usage				

Department of Computer Applications

	2	- Design a multi-process scenario: run commands in foreground and background; evaluate job control commands (jobs, fg, bg) for effective workflow management - Schedule tasks using at and batch; evaluate their execution behavior under different system load conditions and compare their suitability
	3	- Apply nice to modify process priorities; evaluate the effect of priority adjustment on scheduling fairness and overall system performance using time - Design a vi-based text editing workflow for a configuration file; critically evaluate undo, pattern search, and substitution strategies for accuracy and efficiency
Self-Learning	1	Design a real-world server health monitoring automation system; evaluate how cron-based scheduling improves operational reliability over manual execution
	2	- Critically evaluate the Unix signal architecture; design a process termination strategy that prioritizes graceful shutdown over forced kills - Create a mini log-file processing pipeline using vi substitutions; evaluate the trade-offs compared to sed-based automated alternatives
	3	Evaluate zombie and orphan processes — design detection and prevention strategies for production Unix environments

MODULE – IV

	Page No.	From: 145 to 170 & 227 to 260				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	Shell Internals, Redirection and Pipes				
	2	Shell Variables and Environment, Filters				
	3	Pattern Matching with grep, Stream Editing with sed				
Transactions	1	A university Unix server generates large server.log files processed inefficiently using independent commands, causing high resource overhead. Evaluate the current approach and design an optimized solution.				
	2	Your submission must: (i) Critically evaluate the limitations of processing logs without integrated pipelines, (ii) Design an optimized grep (BRE/ERE) + sort + uniq + cut + tr + sed pipeline, (iii) Justify the use of tee, pipes, command substitution, and redirection for performance gains, (iv) Provide a quantitative comparison of traditional vs. optimized approaches (execution time, CPU usage, scalability), (v) Recommend the best approach for heavy-load scenarios with full justification				
	3	Deliverable: Benchmarked command pipelines + comparative analysis + design rationale document				
Practice	1	Design wildcard-based file management operations (*,?,[]) on a multi-extension directory; evaluate the risk of over-matching and propose safeguards				

**Department of Computer Applications**

		Construct a redirection and tee pipeline; critically evaluate how data routing via /dev/null and file redirection affects debugging and logging workflows
	2	- Create and evaluate shell variable ecosystems; assess the security and performance implications of PATH and alias configurations - Design a student record processing pipeline using head, tail, cut, paste, sort, and uniq; evaluate each tool's contribution to a complete data workflow
	3	- Construct grep queries using BRE and ERE on structured datasets; critically evaluate regex design for precision vs. recall trade-offs - Design multi-instruction sed scripts for log file transformation; evaluate the use of line vs. context addressing for targeted editing
Self-Learning	1	Design a shell script pipeline that integrates grep, sort, uniq, tee, and sed for log analysis; evaluate its performance against equivalent Python-based solutions
	2	- Critically examine PATH and environment variable security risks; design a hardening strategy for production shell environments - Evaluate BRE vs. ERE expressiveness across five real-world text matching scenarios; justify regex selection criteria
	3	Create a sed substitution script for automated report formatting; evaluate its robustness against edge cases and unexpected input formats

MODULE – V:

	Page No.	From 271 to 320				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	Shell Scripting Foundations, Control Structures				
	2	Advanced Shell Techniques, AWK Fundamentals,				
	3	Advanced AWK, Shell-AWK Integration				
Transactions	1	Design and implement a complete Shell-based Data Processing System integrated with AWK for analyzing structured records (student, employee, or log data).				
	2	- Part A — Shell Programming: Design a menu-driven script with read, \$0/\$1 argument, if/test/[] validation, case menus, while/for loops, set/shift argument handling, here documents, expr arithmetic, exit status, trap signals, and set -x debugging - Part B — AWK Programming: Create an AWK script file with record filtering, field splitting and processing, printf formatted output, comparison operators, built-in variables (NR, NF), BEGIN/END sections, arrays for data storage, user-defined functions, control flow, and numerical analysis (totals, averages, counts)				

Department of Computer Applications

	3	- Part C — Integration: Critically evaluate the integration design — dynamic file passing from shell to AWK, formatted report generation, modularity, error handling, and scalability - Deliverable: Fully documented, working system with design rationale, test cases, debugging log, and evaluation of design decisions
Practice	1	Design a Student Record Validation System: shell script accepts/validates input (Name, USN, Marks) via if/test[]; AWK script calculates totals, averages, and grades — evaluate input robustness and output accuracy
	2	- Create a Command-Line File Analyzer: shell processes multiple filenames via for/shift; AWK counts words/lines/characters with formatted printf output — evaluate argument-handling strategies and error recovery - Design a Log File Monitoring Tool: while loop continuously monitors a log; trap handles SIGINT gracefully; AWK arrays count occurrences; BEGIN/END generate summaries — evaluate monitoring reliability under load
	3	- Construct a Menu-Driven Data Processing System using case, expr, and modular AWK files for Add/View/Delete/Analyze operations — critically evaluate modular design vs. monolithic scripting - Design a Payroll Processing Application with if/logical operators for validation, expr for salary calculations, AWK for tax-rule application and formatted report generation — evaluate the script using set -x debugging and assess identified defects
Self-Learning	1	Design advanced shell scripts with robust error handling; critically evaluate exit codes and trap-based recovery strategies for production reliability
	2	- Evaluate shell script performance: assess the trade-offs between subshells, pipelines, and in-process computation in time-critical scripts - Design reusable, modular AWK libraries; evaluate function encapsulation strategies for enterprise-scale data processing
	3	Create a live log monitoring and analytics system integrating AWK dynamic filtering with shell control flow; evaluate its real-time performance and accuracy

Department of Computer Applications

Course Outcomes:

- CO-1. Explain the architecture, features, and basic concepts of the Unix operating system, including kernel, shell, and utilities.
- CO-2. Apply Unix commands to perform file management, directory handling, and general system operations effectively.
- CO-3. Analyze Unix file systems, permissions, and user management mechanisms to ensure secure and efficient system usage.
- CO-4. Develop and execute shell scripts to automate routine tasks and solve real-world problems.
- CO-5. Evaluate and manage processes, system utilities, and troubleshooting techniques in Unix environments.

Text Books and References:

- 1. Sumitabha Das., Unix Concepts and Applications., 4th Edition., Tata McGraw Hill
- 2. UNIX: The Complete Reference, Second Edition by Kenneth H. Rosen et al. McGraw-Hill/Osborne 2007

E-References:

- 1. <https://www.tutorialspoint.com/unix/index.htm>
- 2. <https://www.geeksforgeeks.org/linux-unix/introduction-to-unix-system/>

**PROGRAMMING IN C**

Course Code	26MCA160	Formative Assessment Lecture/ Practice	70
Teaching Hours/Week (L: T:P:SL)	2:2:0:2	Summative Assessment	30
Total Hours of Pedagogy (L: T:P:SL)	90 (30:30:0:30)	Total Marks	100
Credits	3	Exam Hours	2 ½

Pre-requisites:

1. Basic C Programming Knowledge – Understanding of variables, data types, operators, control structures (if, loops), and basic functions.
2. Familiarity with Arrays and Strings – Ability to work with one-dimensional arrays, strings, and basic input/output operations (printf, scanf).
3. Fundamentals of Pointers and Functions – Basic understanding of pointers, function calls, and parameter passing in C.
4. Unix – Basic Understanding of Unix Environment, Basic Unix Commands, vi editor.

Course Learning Objectives (CLOs): At the end of the course, the student will be able to

1. To provide a strong foundation in Linux environment, C programming concepts, modular programming, debugging techniques, and efficient program development using GCC, GDB, Makefiles, and preprocessor directives.
2. To develop problem-solving skills using arrays, pointers, dynamic memory allocation, structures, unions, and advanced data manipulation techniques in C programming.
3. To enable students to understand function execution mechanisms, recursion, file handling, system calls, and low-level memory organization in Linux-based systems.
4. To familiarize students with process management, inter-process communication mechanisms, debugging strategies, and program optimization techniques used in system-level programming.
5. To equip students with the knowledge and practical skills required for multithreaded programming using POSIX threads, synchronization techniques, thread communication, and concurrent programming best practices in Linux systems.

MODULE – I

Pedagogy	Page No.	From 9 to 57 (Book 1)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	Week	Hour				
	1	Overview on Linux Environment, C Programming & Execution in Linux, Data Types, Operators and Expressions, Writing C program using multiple files, Debugging and Error Handling (gcc, gdb)				
	2	Preprocessor Directives, Header files and modular programming Makefile, Storage Classes, Scope Rules				
	3	Bitwise operators (&, , ^, ~, <<, >>), Bit manipulation techniques, Applications in embedded systems				
Transactions	1	Evaluate the Role of Modular Programming and Header File Management in Developing Scalable Linux-Based C Applications				
	2	Analyze and Critique the Effectiveness of GCC, GDB, and Makefiles in Large-Scale Software Debugging and Build Automation				

**Department of Computer Applications**

	3	Assess the Impact of Bit Manipulation Techniques and Bitwise Operators in Optimizing Embedded System Applications				
Practice	1	- Develop and debug a multi-file C application in Linux using GCC, GDB, header files, and Makefile to perform student record management with modular programming concepts. (RBTL-4) - Analyze the memory behavior of different storage classes (auto, static, extern, register) by implementing a Linux-based C program and interpreting the program output using debugging tools. (RBTL-4)				
	2	Design a Linux-based embedded-system simulation program using bitwise operators and bit manipulation techniques to control virtual hardware components such as LEDs, switches, or sensors. (RBTL-5)				
	3	- Evaluate the efficiency of modular programming by comparing execution, maintainability, and debugging complexity between single-file and multi-file C applications in Linux. (RBTL-5) - Construct and test a command-line utility in Linux that uses preprocessor directives, macros, and error-handling mechanisms to perform configurable data processing operations. (RBTL-5)				
Self-Learning	1	- Introduction to Shell Scripting and Automation in Linux for C Programmers - Static and Dynamic Library Creation in Linux using ar, ld, and Shared Objects				
	2	- Memory Layout of C Programs in Linux (Stack, Heap, Data, and Text Segments) - Compare and evaluate the use of preprocessor macros and inline functions in C programming with respect to execution efficiency, maintainability, and debugging complexity in Linux environments.				
	3	- Introduction to System Calls and Process Management in Linux - Cross-Compilation Techniques for Embedded Linux Development				
MODULE – II						
	Page No.	From 77 to 100(Book 1)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	1-, 2-& 3-dimensional Arrays, Pointers, Pointer to pointer, Function pointers, Pointer arithmetic				
	2	Dynamic memory allocation, Memory leaks and dangling pointers, void Pointer,				
	3	Pointers with arrays and structures, Nested and arrays of structures, Self-referential structures, Unions				
Transactions	1	Analyse the impact of improper pointer handling in dynamic memory allocation by tracing the execution of a C program that produces memory leaks and dangling pointers. Suggest modifications to ensure safe and efficient memory management. (RBTL Level 4 – Analyse)				
	2	Design a transaction processing application in C using structures, arrays of structures, pointers, and function pointers to manage operations such as adding, updating, searching, and deleting transaction records. Evaluate the efficiency of the chosen data structures and memory management techniques. (RBTL Level 5 – Create/Evaluate)				
	3	Compare and evaluate the use of normal pointers, pointer to pointer, void pointers, and function pointers in implementing generic data processing functions in C. Develop				

**Department of Computer Applications**

		sample programs demonstrating their applications and justify the advantages and limitations of each approach. (RBTL Level 5 – Evaluate)				
Practice	1	1. Write a C program to perform addition and multiplication of 2-dimensional matrices using arrays and pointers. Display the memory addresses of array elements and explain the effect of pointer arithmetic. (RBTL Level 4 – Analyze) 2. Develop a C program to dynamically allocate memory for a 3-dimensional array using malloc () and perform operations such as insertion, traversal, and deallocation. Identify possible memory leaks and demonstrate proper memory management. (RBTL Level 5 – Evaluate/Create)				
	2	3. Implement a menu-driven program using arrays of structures and nested structures to store employee details. Use pointers to structures for accessing and modifying records efficiently. (RBTL Level 4 – Apply/Analyze)				
	3	4. Write a C program demonstrating the use of function pointers to perform arithmetic operations (addition, subtraction, multiplication, division). Compare the implementation with normal function calls and analyze the flexibility achieved. (RBTL Level 4 – Analyze) 5. Construct a linked list application using self-referential structures and pointer-to-pointer concepts for insertion and deletion operations. Extend the program using unions or void pointers to store multiple data types and evaluate the memory efficiency of the implementation. (RBTL Level 5 – Create/Evaluate)				
Self-Learning	1	• Applications of Function Pointers in Callback Mechanisms and Event-Driven Programming in C				
	2	• Implementation of Generic Data Structures Using Void Pointers and Dynamic Memory Allocation				
	3	• Memory Optimization Techniques Using Unions and Self-Referential Structures in System Programming				
MODULE – III						
	Page No.	From 170 to 245(Book 3)				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	Recursion (advanced problems), Inline functions, passing structures to functions, returning pointers from functions				
	2	Function Stack, Frame Pointer, Stack Frame, Function Call Mechanism in Linux, what is System Call, System calls for file handling				
	3	File modes and operations, Binary file handling, Random access in files, Error handling in files				
Transactions	1	• Comparative Analysis of Standard File Handling Functions and Linux System Calls in C Programming (RBTL Level 5)				
	2	• Design and implement a Linux-based file management system in C that uses recursive directory traversal to analyse file structures, perform binary file operations, and handle errors using system calls. Compare the performance and reliability of system calls versus standard library functions for file handling. (RBTL Level 5 – Evaluate)				

**Department of Computer Applications**

	3	Create a binary database management program in C using random file access and Linux system calls for file operations. The program should support insertion, deletion, modification, and retrieval of records with robust file error handling mechanisms. Justify the design choices for memory management, stack utilization, and file access techniques.				
Practice	1	- Write a C program to implement recursive solutions for advanced problems such as Tower of Hanoi, N-Queens, or maze traversal, and trace the function stack and stack frame changes during execution using Linux debugging tools. (RBTL Level 5) - Implement a Linux-based file handling program using system calls (open, read, write, lseek, close) to perform file creation, reading, writing, and random-access operations on binary files with proper error handling. (RBTL Level 5)				
	2	Develop a program to analyze the function call mechanism in Linux by displaying stack frame information, frame pointer values, and recursion depth for nested function calls. Use debugging tools such as gdb to validate the execution flow. (RBTL Level 6)				
	3	- Implement a recursive file search utility in C that traverses directories and subdirectories in Linux, identifies files based on extensions or size, and logs the results into a binary file using system calls. (RBTL Level 5) - Design and implement a mini banking record management system in C using binary files and random-access techniques. The application should support secure record insertion, transaction updates, balance inquiry, and robust error handling for invalid file operations. (RBTL Level 5)				
Self-Learning	1	Explore how recursion is used with linked lists, trees, and graphs along with memory management using malloc (), calloc(), and free ()				
	2	Study how Linux manages files internally, including inode structure, file descriptors, buffering, and low-level I/O operations.				
	3	Learn debugging using tools such as gdb, strace, and valgrind for tracing system calls, memory leaks, and runtime errors.				
MODULE – IV						
	Page No.	From 280 to 325				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	Process Creation, Process Table, Inter-process communication - pipes				
	2	Message Queues, Semaphore, Shared Memory, signals basics				
	3	Debugging techniques (gdb, assertions), Program optimization techniques				
Transactions	1	Design and develop a Linux-based process management system that demonstrates process creation using fork() and implements communication among multiple child processes using shared memory and semaphores. Integrate signal handling for				

Department of Computer Applications

		process synchronization and graceful termination. Optimize the program for minimum CPU and memory usage, and document the debugging and optimization techniques used with tools such as gdb and assertions. (RBTL Level 6)				
	2	• Create a comparative experimental framework to study program optimization techniques in process-intensive applications. The framework should: ○ generate multiple concurrent processes, ○ monitor process table entries and resource usage, ○ identify bottlenecks using debugging tools (gdb, assertions), and ○ implement optimizations such as reduced context switching, efficient IPC usage, and memory optimization. Prepare a detailed report analyzing performance improvements before and after optimization. (RBTL Level 6)				
	3	• Modern IPC Mechanisms in Linux: Comparison of Pipes, Sockets, Shared Memory, and Message Queues				
Practice	1	• Write a Linux C program to create multiple child processes using fork() and display the Process ID (PID), Parent Process ID (PPID), and process hierarchy information using the process table. (RBTL Level 3) • Implement inter-process communication between two processes using: (RBTL Level 4) ○ unnamed pipes, and ○ named pipes (FIFO) ○ Compare their behavior for bidirectional communication				
	2	• Develop a producer–consumer application using shared memory and semaphores to synchronize data exchange between processes.(RBTL Level5)				
	3	• Write a program to demonstrate signal handling in Linux using signals such as SIGINT, SIGKILL, and SIGUSR1. Implement custom handlers and observe process behavior during signal reception. (RBTL Level 3) • Debug and optimize a CPU-intensive C program using gdb and assertions. Identify runtime errors, memory-related issues, and optimize the program execution time using suitable program optimization techniques. (RBTL Level 5)				
Self-Learning	1	• Efficient Resource Sharing in Concurrent Systems using Semaphores, Shared Memory, and Message Queues				
	2	• Process Scheduling and Context Switching Techniques in Multitasking Operating Systems				
	3	• Performance Profiling and Code Optimization Tools in Linux (gprof, valgrind, perf)				
MODULE – V:						
	Page No.	From 300 to 358				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
	Week	Hour				
Learning	1	Introduction to POSIX Threads (Pthreads), Thread Synchronization, Mutexes in Pthreads, Condition Variables				
	2	Semaphores and Thread Coordination, Thread Communication and Shared Memory				
	3	Thread Scheduling and Performance, Debugging and Best Practices with pthreads				

Department of Computer Applications

Transactions	1	Design and implement a multithreaded file-processing system using POSIX threads, where multiple worker threads read, process, and update shared data concurrently. Use mutexes, condition variables, and semaphores to avoid race conditions and ensure synchronization. Analyse thread scheduling behaviour and evaluate the impact of synchronization mechanisms on performance under varying workloads. RBTL Level: 6 – Create
	2	Performance Optimization and Debugging of Multithreaded Applications using Pthreads, Study thread scheduling, synchronization overhead, race condition detection, profiling tools, and best practices for developing efficient and reliable pthread-based systems. RBTL Level: 6 – Create
	3	Create a parallel task-execution framework in Linux using POSIX threads, where tasks are dynamically assigned to threads based on priority and scheduling policies. Implement thread coordination, shared resource management, and synchronization primitives to maintain correctness and efficiency. Evaluate thread scheduling algorithms, measure execution performance, and document best practices for debugging and optimizing pthread-based applications. RBTL Level: 6 – Create
Practice	1	- Write a POSIX threads (Pthreads) program to create multiple threads that perform parallel computation and display thread IDs, execution order, and shared data access details. RBTL Level: 3 – Apply - Develop a multithreaded program using mutexes to synchronize access to a shared counter variable and demonstrate the prevention of race conditions. RBTL Level: 4 – Analyze
	2	Implement the producer-consumer problem using Pthreads, semaphores, and condition variables with a bounded shared buffer. RBTL Level: 5 – Evaluate
	3	- Write a Pthreads-based application to demonstrate thread communication through shared memory, where multiple threads read and update common data structures safely. RBTL Level: 4 – Analyze - Create a multithreaded scheduling simulation that measures execution time and compares thread performance under different scheduling policies and synchronization techniques. RBTL Level: 5 – Evaluate
Self-Learning	1	Thread Safety in Standard Libraries and Reentrant Functions in Linux Study how thread-safe functions and reentrant programming techniques improve reliability in concurrent applications.
	2	CPU Affinity and Processor Binding in Multicore Systems Explore how threads can be assigned to specific CPU cores to improve cache utilization and application performance in Linux systems.
	3	Hybrid Parallel Programming using POSIX Threads and OpenMP Learn how pthreads can be integrated with OpenMP for scalable parallel application development in scientific and high-performance computing.
Course Outcomes: CO-1. Apply Linux-based C programming concepts to develop, compile, debug, and manage modular applications using GCC, GDB, Makefiles, and preprocessor directives. CO-2. Demonstrate proficiency in using arrays, pointers, dynamic memory allocation, structures, unions, and bitwise operations for efficient problem solving and memory management. CO-3. Analyze function execution mechanisms, recursion, file handling techniques, and Linux system calls to design		



Department of Computer Applications

robust system-level applications.

- CO-4. Implement process management and inter-process communication techniques such as pipes, message queues, semaphores, shared memory, and signals in Linux environments.
- CO-5. Develop multithreaded applications using POSIX threads with proper synchronization, thread communication, debugging, and performance optimization techniques.

Text Books and References:

1. The C Programming Language by Brain W. Kernighan, Dennis M. Ritchie – 2nd Edition
2. Expert C Programming: Deep C Secrets - Peter van der Linden
3. Let Us C - Yashavant Kanetkar

COMMUNICATIVE ENGLISH

Course Code	26CEE170	CIE Marks	50
Teaching Hours/Week (L: T:P:SL)	01:00:01:01	SEE Marks	0
Total Hours of Pedagogy	30 (15:00:15:15)	Total Marks	50
Credits	01	Exam Hours	01

Pre-requisites:

- Basic understanding of English grammar.
- Familiarity with sentence construction and vocabulary.
- Ability to comprehend simple English texts and communicate ideas in basic English.

Course objectives:

- To develop effective communication skills in English for academic and professional use.
- To improve fluency and accuracy in spoken and written English.
- To develop an understanding of grammatical structures and vocabulary for everyday communication.
- To enhance listening, speaking, reading, and writing (LSRW) skills through interactive and experiential learning.
- To prepare students for interviews, presentations, and workplace communication

Module-1

Topic	Number of hours of engagement	L	T	P	SL	Total
		3	0	3	3	6
Lecture	Introduction to Communication and Grammar • Basics of Communication • Elements of communication • Types of communication: verbal, non-verbal, written, and visual • Barriers to communication • Grammar essentials: Tenses, Sentence formation, Subject-verb agreement, Articles, Prepositions					
Tutorial	NIL					
Practice / Activity	Lab: Wordsworth Lab Prep - 1 Beginners, Sessions 36 Activity: Ice-breaking group tasks Grammar games and quizzes Role-play on communication breakdown scenarios					
Self-Learning	Assignment	Prepare a report (300–400 words) on any innovative technical product.				
	Seminar	"The Role of Non-Verbal Communication in the Digital Age" – Discuss how body language, facial expressions, and gestures are still relevant despite virtual interactions.				
	Self-Reading	Reading articles from magazines and newspapers.				

Department of Computer Applications

Reference						
1. Communication Skills (Sanjay Kumar, Pushpalata)					Page No. 1-20	
Module-2						
Topic	Number of hours of engagement	L	T	P	SL	Total
		3	0	3	3	6
Lecture	Phonetics Introduction to phonetics: Phonetics Symbols Pronunciation Guidelines to Vowels: Monophthongs and Diphthongs Pronunciation Guidelines: Consonants sounds Syllables: Structure and Phonetic Transcription. Pronunciation of Letters: Silent and Non-Silent Letters Word Stress and Word Accent, Stress Shift					
Practice / Activity	Lab: Wordsworth Prep -2 Intermediate, Sessions 30 Activity: Phonetics related videos and Practicing vowel and consonants sounds in class.					
Self-Learning	Assignment	Identify and classify 20 English words based on their vowel and consonant sounds (IPA symbols).				
	Seminar	The Importance of Phonetics in Spoken English Mother Tongue Influence (MTI) and how phonetics can help reduce It.				
	Self-Reading	Practice with online phonetic transcription tools Watch videos/tutorials on place and manner of articulation.				
Reference						
1. Communication Skills (Sanjay Kumar, Pushpalata)					Page No- 189-210	
Module-3						
Topic	Number of hours of engagement	L	T	P	SL	Total
		3	0	3	3	6
Lecture	Grammar for Communication Why Grammar? Parts of Speech: Noun, Pronoun, Verb, Adjective, Adverb, Preposition, Conjunction and Interjection. Determiners and Articles Question Tags: Rules for Question Tags Modals: Modals and its types Transformation of sentences: Reported Speech, Active & Passive Voice Using Tenses: Present, Past and Future Tense					
Practice / Activity	Lab: Wordsworth Prep 5 – Grammar, Sessions – 6 Activity: Sentence Relay Game: Teams transform given sentences quickly and accurately Transformation Challenge: Turn a story from active to passive voice					

Department of Computer Applications

Self-learning	Assignment	Rewrite 15 sentences by changing their voice, form (assertive to interrogative, etc.), and degree of comparison. Correct and transform reported speech in a short paragraph				
	Seminar	Mastering Sentence Transformation: A Key to Clear Communication Voice and Tense: How Sentence Structure Affects Meaning				
Reference						
(Wren and Martin)		Page No- 11- 142				
Module-4						
Topic:	Number of hours of engagement					
		L	T	P	SL	Total
		3	0	3	3	6
Lecture	Mastering Communication Skills Listening: Dialogues & Conversations, Debate and Discussions & TED Talks Speaking: Self – Introduction, Role plays, speech on Friends/Hobbies/Current Affairs, MTI (Mother Tongue Influence) Reading: Comprehension, making inference on reading passage, Article, Magazine, Newspaper. Writing: Free writing, Types of Paragraph Writing, Essay Writing, Formal & Informal Letters and E-mail writing, Show Cause and Blog Writing Vocabulary for Competitive Exams					
Practice / Activity	Lab: Wordsworth Prep – 4, Language Enrichers, Sessions - 10 Activity: Reading news articles and summarizing. Vocabulary games like word ladders, crossword puzzles. Book club: read and discuss a short story.					
Self-Learning	Assignment	Choose a tech article or short story and write a detailed summary, including at least 15 new vocabulary words with meanings and usage in your own sentences.				
	Seminar	"Reading Habits of Successful Tech Professionals" – Explore how continuous reading contributes to communication and innovation in the tech world.				
	Self-Reading	Word of the day (team-based dictionary wall). Create vocabulary flashcards (physical or digital).				
Reference						
Technical Communication- (Meenakshi Raman)				Page No – 48- 139, 206-340		
Module-5						
Topic	Number of hours of engagement	L	T	P	SL	Total
		3	0	3	3	6

Department of Computer Applications

Presentations & Soft Skills Development:	Public speaking and confidence building. Presentation tools and techniques (PowerPoint, Canva, etc.) Time management and team collaboration. Non-verbal communication: body language, eye contact. Visual Aids and Presentation Tools Techniques for managing presentation anxiety.	
Practice/Activity	Lab: Wordsworth Lab Prep – 3 Proficient, Sessions -28 Activity: Group presentations on tech topics. Soft skill role-plays (conflict resolution, negotiation) Feedback sessions using peer assessment rubrics.	
Team work and self-learning	Assignment	In a group of 3–5 students, create a presentation on an emerging technology (e.g., AI, Blockchain, Quantum Computing) and deliver it in class using PowerPoint or Canva.
	Seminar	"Soft skills: every techie must master" – Present on teamwork, adaptability, emotional intelligence, and their growing importance in the IT industry
	Self-Reading	Create and deliver a team presentation Watch and reflect on great speeches (e.g., Steve Jobs, Obama)
Reference		
Functional English (Gajendra Singh Chauhan)		Page No-309-333
Course Outcomes: By the end of this course, students will be able to:		
CO-1: Demonstrate proficiency in English communication for academic and workplace settings.		
CO-2: Identify the nuances of phonetics, intonation and enhance pronunciation skills.		
CO-3: To impart basic English grammar and essentials of language skills as per present requirement.		
CO-4 To impart effective listening, speaking, reading, and writing in English, demonstrating improved comprehension, communication, and writing skills in various contexts.		
CO-5 Collaborate with peers effectively and present ideas confidently using appropriate tools.		
Text Books and References:		
1. Communication Skills by Sanjay Kumar and Pushp Lata		
2. Functional English by Wren and Martin		

Department of Computer Applications**DATABASE MANAGEMENT SYSTEM**

Course Code	26MCA210	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:2:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	120 (30:30:30:30)	Total Marks	100
Credits	04	Exam Hours	2 ½

Pre-requisites:

1. Basic knowledge of programming skills, fundamentals of computer, mathematics and file systems.

Course Learning Objectives (CLOs):

1. Comprehend fundamental database principles, including Database approach, Data Models, Schemas, Instances, DBMS architecture.
2. Demonstrate proficiency in Relational Data Model concepts, Relational Algebra operations, Relational Calculus.
3. Acquire skills in Function Dependencies, Normalization techniques (1NF, 2NF, 3NF, BCNF), informal design guidelines.
4. Understand transaction basics, implement atomicity and durability, grasp concurrent executions, serializability, recovery techniques, failure classification, storage structure, and recovery methods.
5. Insights into Distributed databases, Multimedia databases, Object-oriented database management systems

MODULE – I

Pedagogy	Page No.	From: 1 to 50				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	Week	Hour		Hour		
	1	Introduction to Database, Database Users		Characteristics of DBMS, Advantages of DBMS		
	2	Data Models, Schemas & Instances		DBMS Architecture, Data Independence		
	3	System Architecture, Data Dictionary		Database Languages & Interfaces		
Transactions	1	Introduction to ER Model – Entity Types, Entity Sets, Attributes, Keys				
	2	Relationship Types, Weak Entity Types, Structural Constraints				
	3	Enhanced ER Model – Specialization, Generalization, Constraints				
Practice	1	Identify entities, attributes, and keys from real-world scenarios				
	2	Identify entities, attributes, and keys from real-world scenarios				
	3	Design Enhanced ER models with specialization and generalization				

MODULE – II

	Page No.	From: 51 to 90				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Relational Data Model Concepts		Constraints		
	2	Relational Algebra – Select		Project, Set Operations		
	3	Join Operations		Overview of Relational Calculus		
Transactions	1	Introduction to SQL, Data Definition Commands				
	2	Queries, Views, Transaction Control Commands				
	3	Constraints, Indexes in SQL				
Practice	1	Solve relational algebra queries				
	2	Write SQL queries (DDL & DML)				

Department of Computer Applications

	3	Create tables, apply constraints and indexes				
MODULE – III						
	Page No.	From: 91 to 124				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Functional Dependencies		Design Guidelines		
	2	Normalization – 1NF, 2NF, 3NF		BCNF, 4NF & 5NF		
	3	BCNF, Lossless Join		Dependency Preservation		
Transactions	1	Multivalued Dependencies				
	2	Join Dependencies, 4NF & 5NF				
	3	Denormalization				
Practice	1	Identify functional dependencies				
	2	Normalize relations up to 3NF/BCNF				
	3	Perform decomposition and check lossless property				
MODULE – IV						
	Page No.	From: 125 to 155				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Transaction concepts and system structure		ACID properties and transaction states		
	2	Schedules: serial & non-serial		Recoverability and serializability		
	3	Concurrency control techniques		Two-Phase Locking (2PL)		
Transactions	1	Locking mechanisms and multiple granularity locking				
	2	Concurrency issues: lost update, dirty read, phantom problem				
	3	Basic SQL transaction support (COMMIT, ROLLBACK, SAVEPOINT)				
Practice	1	SQL transactions (COMMIT, ROLLBACK) implementation				
	2	Concurrency control case studies / problems				
	3	Recovery techniques demonstration: Log-based recovery (UNDO/REDO) Shadow paging (basic idea)				
MODULE – V:						
	Page No.	From: 156 to 190				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Active databases and triggers (concept, types, use cases)		Overview of temporal, spatial, and multimedia databases		
	2	Information Retrieval (IR): concepts and applications Retrieval models (Boolean, Vector – basic idea)		Text preprocessing and inverted indexing		
	3	Evaluation of IR (precision, recall) Web search basics		Database security fundamentals (overview, threats)		
	1	Access control mechanisms: DAC, RBAC, basic MAC				

**Department of Computer Applications**

Transactions	2	SQL Injection: concept, examples, prevention
	3	Encryption basics and Public Key Infrastructure (PKI)
Practice	1	QL triggers implementation Text preprocessing (tokenization, stop word removal)
	2	SQL Injection demonstration and prevention techniques Role-Based Access Control (RBAC) implementation
	3	Basic encryption demonstration Case study on database security and privacy issues

Course Outcomes:

- CO-1. Understand fundamental concepts of database systems, data models, DBMS architecture, and ER modeling.
- CO-2. Apply relational data model concepts and perform queries using relational algebra and SQL.
- CO-3. Design relational database schemas using normalization techniques and functional dependencies.
- CO-4. Analyse transaction management, concurrency control, and recovery mechanisms in DBMS.
- CO-5. Explore advanced database concepts including distributed databases, multimedia databases, and data warehousing & mining.

Text Books:

1. Elmsari and Navathe Fundamental of Database System 7th Edition Addison Wesley, New York
2. H. Korth & A. Silberschatz DATABASE SYSTEM CONCEPTS 6th Edition Tata McGraw-Hill (TMH)

References

1. Date. CJ An Introduction to Database System 8th Edition Narosa Publishing House, New Delhi
2. Ullman. J.D Principles of Database Systems 3rd Edition Galgotia Publications, New Delhi

OBJECT ORIENTED PROGRAMMING USING JAVA

Course Code	26MCA220	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:2:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	120 (30:30:30:30)	Total Marks	100
	04	Exam Hours	2 ½

Pre-requisites:

1. Basic knowledge of programming concepts and logic development
2. Familiarity with any one high-level programming language like C or C++
3. Understanding of fundamental computer science concepts such as variables, data types, and control structures

Course objectives:

1. Students will understand the fundamentals of Java programming and Object-Oriented Programming concepts.
2. Students will develop problem-solving skills using Java language constructs and control statements.
3. Students will design and implement reusable classes, interfaces, and packages.
4. Students will demonstrate advanced Java concepts such as exception handling, multithreading, collections, and file handling.
5. Students will build database-driven and event-driven applications using JDBC, Servlets, and GUI programming.

MODULE – I

Pedagogy	Page No.	From: 1 to 180				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	Week	Hour		Hour		
	1	Introduction to Java and Features of Java		Operators and Input/Output Operations		
	2	JVM, JDK, JRE and Structure of Java Program		Control Statements in Java		
	3	Data Types and Variables		Arrays, Methods and Command Line Arguments		
Transactions	1	Smart Inventory Billing System:-A supermarket wants to automate billing and inventory management for thousands of products sold every day. Can you create a Java program that accepts product details, calculates discounts, applies taxes, and generates the final bill automatically? The system should also identify low-stock products and display alerts whenever stock reaches a minimum level. How can arrays, loops, conditional statements, and methods help in developing an efficient billing solution?				
	2	University Result Analytics System:-A university examination department wants to analyse student performance across multiple subjects and semesters. Can you design a Java application that calculates grades, class averages, toppers, pass percentages, and subject-wise statistics? The system should generate different performance reports automatically for faculty members and administrators. How can nested loops, arrays, and decision-making statements improve large-scale				

Department of Computer Applications

		academic result processing?				
	3	Employee Payroll and Tax Processing System:-A multinational company wants to automate salary processing for employees working in different departments and locations. Can you create a Java program that calculates salary using allowances, overtime, bonuses, insurance deductions, and income tax rules? The application should generate monthly salary slips and identify employees with exceptional performance incentives. How can arithmetic operators, conditions, and modular programming improve payroll management efficiency?				
Practice	1	Why is Java considered platform independent? Explain with JVM architecture?				
	2	How are arrays useful in managing student marks or employee records?				
	3	Why are control statements important in real-time applications?				
MODULE – II						
	Page No.	From: 181 to 360				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Classes and Objects			Method Overriding and Dynamic Binding	
	2	Constructors and Method Overloading			Packages and Access Specifiers	
	3	Inheritance and Types of Inheritance			Interfaces and Abstract Classes	
Transactions	1	A national bank wants to create software for handling savings, current, and loan accounts efficiently. Can you create Java classes and objects to store customer details, account balance, and transaction history? The application should use constructors for account initialization and method overloading for different transaction operations. How can inheritance and dynamic binding improve flexibility and code reuse in banking applications?				
	2	A university wants to manage details of students, teaching staff, and non-teaching staff in a single application. Can you design a Java program using inheritance where common features are shared and specialized behaviors are implemented separately? The application should use method overriding to display different roles and responsibilities dynamically. How can packages and access specifiers improve security and organization in large academic systems?				
	3	An e-commerce company wants to manage products, customers, and payment methods using object-oriented programming concepts. Can you create interfaces for different payment options such as UPI, Credit Card, and Net Banking? The system should use abstract classes for common shopping operations and implement different payment behaviors dynamically. How can interfaces and abstract classes improve scalability and maintainability in large software systems?				
Practice	1	Why is inheritance useful in software development?				
	2	Differentiate between abstract classes and interfaces with examples.				
	3	How do packages help in large-scale application development?				

Department of Computer Applications

MODULE – III						
	Page No.	From: 361 to 520				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Exception Handling Basics			Thread Lifecycle and Synchronization	
	2	try-catch-finally, throw and throws			String Handling and Wrapper Classes	
	3	Introduction to Multithreading			File Handling in Java	
Transactions	1	A bank ATM application must safely handle situations like insufficient balance, invalid PIN entry, and incorrect withdrawal amounts. Can you create a Java program that uses try-catch-finally, throw, and throws to manage transaction errors effectively? The system should display meaningful error messages and ensure that the program continues execution without crashing. How can exception handling improve reliability and security in banking applications?				
	2	food delivery company wants to track multiple customer orders simultaneously in real time. Can you create a multithreaded Java application where separate threads manage order preparation, delivery updates, and payment confirmation? The application should synchronize shared resources to avoid conflicts when multiple orders are processed together. How does thread lifecycle management and synchronization improve efficiency in real-time systems?				
	3	A college administration wants to store student details such as name, registration number, marks, and attendance permanently in files. Can you create a Java program that performs reading, writing, searching, and updating student records using file handling concepts? The application should validate numeric and text data using wrapper classes and string handling methods before storing information. How can file handling improve long-term storage and retrieval of academic records?				
Practice	1	Why is exception handling important in software applications?				
	2	How does multithreading improve application performance?				
	3	Why are files used for permanent data storage?				
MODULE – IV						
	Page No.	From: 521 to 720				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Introduction to Collections Framework			Map Interface, HashMap and TreeMap	
	2	Array List, LinkedList and Vector			GUI Programming using AWT	
	3	Set Interface and HashSet			Swing Components and Event Handling	
Transactions	1	A university wants students to register for courses using an interactive graphical application. Can you create a Java Swing application with labels, text fields, buttons, and combo boxes for collecting student details? The system should validate user inputs and display confirmation messages whenever registration is completed successfully. How does event handling improve interaction between users and graphical applications?				
	2	A hospital wants to manage patient appointments using a GUI-based software				

Department of Computer Applications

Practice		application. Can you design a Java program with forms for entering patient details, doctor selection, and appointment timing? The application should respond to button clicks and user actions dynamically using event-driven programming concepts. How can GUI applications improve usability and efficiency in healthcare management systems?
	3	A finance company wants a desktop application that performs arithmetic calculations and currency conversion operations. Can you create a Java GUI application using Swing components and event handling mechanisms? The application should process user inputs through buttons and display accurate results instantly. How does event-driven programming improve responsiveness and user experience in desktop applications?
	1	Why are collections preferred over arrays in dynamic applications?
	2	Explain real-world applications of HashMap.
	3	Why are GUI applications more user friendly?

MODULE – V:

	Page No.	From: 721 to 900				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	JDBC Architecture and Drivers			Event Handling and Delegation Event Model	
	2	Database Connectivity and SQL Operations			Java Utility Packages and Collections Review	
	3	Java Networking Basics and URL Handling			Servlets and session management	
Transactions	1	A university portal allows students and faculty members to access attendance, assignments, examination results, and notices securely. Can you create a Java application that validates login credentials using JDBC connectivity and database queries? The system should prevent invalid login attempts and securely retrieve user information from the database. How does JDBC improve secure communication between Java applications and relational databases?				
	2	A competitive examination platform wants to conduct online tests and evaluate thousands of students automatically. Can you design a Java application that stores questions, answers, scores, and examination details in a database? The application should generate scorecards automatically and maintain examination history for future reference. How can database connectivity improve scalability and efficiency in online examination systems?				
	3	A multinational company wants to maintain employee records across multiple departments and branch offices digitally. Can you create a Java application that inserts, updates, deletes, and retrieves employee information using JDBC operations? The system should generate department-wise reports, salary summaries, and employee performance details dynamically. How does database management improve enterprise-level data processing and				

Department of Computer Applications

		accuracy?
Practice	1	Why is JDBC important for enterprise applications?
	2	How does Java networking help in client-server communication?
	3	Explain the importance of database connectivity in real-world systems.

Course Outcomes:

- CO-1. Understand the fundamentals of Java programming and object-oriented concepts.
- CO-2. Develop Java programs using classes, inheritance, interfaces, and packages.
- CO-3. Apply exception handling, multithreading, and file handling concepts in Java applications.
- CO-4. Design GUI-based applications and utilize Java Collections Framework effectively.
- CO-5. Build database-driven, networking, and event-driven Java applications using JDBC and Java utility packages.

Text Books:

1. Herbert Schildt, Java: The Complete Reference, McGraw Hill.
2. E. Balagurusamy, Programming with Java, McGraw Hill.
3. Herbert Schildt, Java 2: The Complete Reference, McGraw Hill

References

1. Cay S. Horstmann, Core Java Volume I – Fundamentals, Pearson.
2. Deitel & Deitel, Java How to Program, Pearson.
3. Bruce Eckel, Thinking in Java, Prentice Hall.
4. The syllabus topics are aligned with concepts covered in Herbert Schildt's Java: The Complete Reference.

**Software Engineering & Project Management**

Course Code	26MCA230	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:2:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	120 (30:30:30:30)	Total Marks	100
Credits	04	Exam Hours	2 ½

Pre-requisites:

1. Students should have a foundational understanding of programming concepts and at least one programming language (e.g., Java, C++, or Python).
2. A general idea of the stages involved in developing software—from requirements to deployment—will help students grasp advanced models and methodologies.
3. Since software design and component development often rely on efficient data handling and logic, prior knowledge in this area is essential.

Course objectives:

1. Learn what software is and how software engineers build it using different stepby-step methods.
2. Understand how agile development works and how teams can quickly and effectively build software.
3. Know how to gather, organize, and check software requirements using tools like diagrams and models.
4. Learn how to design the structure of software and test it to make sure it works well.
5. Gain basic project management skills like planning, time estimation, handling risks, and managing team work in software projects.

Unit-1

Pedagogy	Page No.	From: 67-75, 185-194				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	Week	Hour		Hour		
	1	Software and Software Engineering		The Nature of Software, Software Engineering, Software Engineering Practice,		
	2	Process Models		A Generic Process Model, Prescriptive Process Models: Waterfall, Incremental Process, Evolutionary Process & Concurrent Models,		
	3	Specialized Process Models		Component Based development, Formal Methods model, Aspect-oriented software development, The Unified Process, Personal and Team Process Models		
Transactions	1	Evaluate different software process models to identify the most suitable approach for developing large-scale software systems.				
	2	Analyze the effectiveness of traditional and modern software engineering practices in real-world software development.				
	3	Design a customized software development process integrating Agile and Component-Based approaches for an enterprise application.				
Practice	1	Develop a hybrid software process model for a Library Management System and justify the workflow and development phases.				
	2	Evaluate software project outcomes based on the selection and implementation of process models.				

**Department of Computer Applications**

	3	Construct a collaborative software development framework using PSP and TSP practices to improve productivity and quality.				
Unit-2						
	Page No.	From: 273-285				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	What is Agility? What is an Agile Process			Agility Principles, The politics of Agile Development, Human Factors,	
	2	Extreme Programming (XP):			XP values, XP process, Industrial XP, The XP debate,	
	3	Agile Process Models			Adaptive Software Development(ASD), Scrum, Dynamic Systems development method, Crystal, Feature Driven development, Lean Software Development, Agile Modeling, Agile Unified Process, A Toolset for Agile Process	
Transactions	1	Evaluate the effectiveness of Agile methodologies such as Scrum, XP, and Lean in managing dynamic software projects.				
	2	Design a complete Agile workflow including sprint planning, backlog management, and continuous integration for a software product.				
	3	Analyze the challenges and benefits of implementing Agile practices in distributed software development environments.				
Practice	1	Construct an Agile project dashboard using Jira or Trello to monitor sprint progress and backlog activities.				
	2	Develop and manage an Agile project lifecycle using Scrum practices and project management tools.				
	3	Evaluate sprint performance and team collaboration during Agile software development activities.				
Unit-3						
	Page No.	From: 485-507; 589-592				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Understanding Requirements			Requirements Engineering, Eliciting requirements, Developing Use cases, Building the Requirements model, Negotiating & Validating requirements,	
	2	Requirement Modeling			Requirement Analysis, Scenario based modelling, UML models, Data modelling concepts, Class based modelling, Flow oriented Modelling, Creating a behavioural model, Patterns for Requirements Modeling	
	3	The Design Process,				
Transactions	1	Analyze UML and scenario-based models to identify inconsistencies and improve requirement clarity in software systems.				
	2	Evaluate requirement elicitation and validation techniques for improving the quality of software requirements specifications.				
	3	Design a complete Software Requirements Specification (SRS) document with UML models for a real-world application.				
	1	Develop UML-based requirement models including use case, sequence, and activity				

**Department of Computer Applications**

Practice		diagrams for a software system.				
	2	Evaluate software requirement documents to ensure correctness, consistency, and completeness.				
	3	Construct behavioral and flow-oriented models for an enterprise-level software application.				
Unit-4						
	Page No.	From: 546:570				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	Architectural Design			Software Architecture, Architectural Style, Architectural Design, Architectural Mapping using Data Flow,	
	2	Component Level Design			What is a Component? Designing Class-Based Components.	
	3	Software Testing:			Verification & Validation, Testing Strategies for Conventional software, Validation testing, System Testing, White Box testing, Basis Path testing, Control Structure testing, BlackBox testing, Model Based testing	
Transactions	1	Analyze component-level software designs and testing reports to improve software reliability and quality.				
	2	Evaluate software architectural styles and testing strategies for developing scalable and maintainable software systems.				
	3	Design a complete software architecture and testing framework for a real-world application using standard design principles.				
Practice	1	Evaluate software testing results to identify defects and improve system reliability and performance.				
	2	Develop a layered software architecture and component interaction model for an Online Bookstore application.				
	3	Construct automated testing strategies integrating white-box, black-box, and model-based testing techniques.				
Unit-5						
	Page No.	From: 782-797				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	6	6	24
Learning	1	The Management Spectrum,			Metrics in the Process & Project Domains, Software Measurement, Metrics for Software Quality,	
	2	Estimation for Software Projects:			Resources, Decomposition Techniques, Empirical estimation models, Project Scheduling, Scheduling, Earned Value Analysis,	
	3	Risk Management:			Software Risks, Risk Identification, Risk Projection, Risk Refinement, Risk Mitigation, Monitoring & Management, The RMMM Plan	
Transactions	1	Design a comprehensive software project management plan including estimation, scheduling, and risk mitigation strategies.				

Department of Computer Applications

	2	Evaluate software estimation models and scheduling techniques for effective software project planning and execution.
	3	Analyze software project risks and quality metrics to improve project monitoring and management processes.
Practice	1	Develop a complete software project plan with cost estimation, scheduling, and resource allocation for a software application.
	2	Evaluate software project risks and propose optimized mitigation and monitoring strategies using an RMMM plan.
	3	Construct project tracking and reporting mechanisms using software metrics and earned value analysis techniques.

Course Outcomes:

- CO-1. Understand and explain the fundamental concepts and practices of software engineering, including various traditional and modern software process models.
- CO-2. Apply agile methodologies and principles to manage software development projects with a focus on collaboration, flexibility, and continuous improvement.
- CO-3. Analyze, model, and document software requirements using structured techniques such as use cases, UML, and scenario-based models.
- CO-4. Design software architectures and components using standard design principles and evaluate them through structured testing strategies.
- CO-5. Estimate, plan, and manage software projects using metrics, risk analysis, scheduling, and monitoring tools to ensure project success.

Text Books:

1. Roger S Pressman, Software Engineering: A Practitioners Approach, McGraw-Hill, 2010.

References

1. Ian Sommerville, Software Engineering, 10th Edition, Pearson Education, 2015.
2. Pankaj Jalote, An Integrated Approach to Software Engineering, 3rd Edition, Wiley publications, 2005

Data Communication

Course Code	26MCA241	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:0:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	(30:30:0:30) 90	Total Marks	100
Credits	3	Exam Hours	2 ½

Pre-requisites: 1. Basic understanding of computer networks, binary systems, and data representation. 2. Familiarity with networking models, communication protocols, and digital communication concepts.

Course Learning Objectives (CLOs)

1. To provide a comprehensive understanding of fundamental concepts in data communication and networking, including signal propagation and transmission media.
2. To familiarize students with various network topologies, communication protocols, standards, and the role of standard organizations like IEEE, ITU, and ISO.
3. To explain and differentiate between analog and digital communication, signal encoding, transmission modes, and error detection and correction techniques.
4. To introduce multiplexing techniques, data link layer protocols, network models (OSI and TCP/IP), and local to wide area networks (LAN, MAN, WAN).
5. To explore modern network technologies such as wireless communication standards (802.x series), IPv4/IPv6, cryptographic techniques, and secure transmission protocols like SSL/TLS.

MODULE – I

Pedagogy	Page No.	From: 4 to 24, 55-65				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	Week	Hour		Hour		
	1	Fundamental concepts of Data Communication, components of communication systems, data representation.		Communication models, and types of transmission media.		
	2	Protocols, standards, standard organizations, layered architecture.		Signal propagation, analog and digital signals, characteristics of signals, and noise.		
	3	Bandwidth of a signal and medium, Fourier Analysis, concepts of bandwidth.		Data transmission rate, channel capacity, and relationship between bandwidth and data rate.		
Transactions	1	Explanation of data communication concepts, communication process, protocols, standards, and roles of standard organizations using diagrams and examples.				
	2	Discussion on analog and digital signals, signal propagation techniques, signal characteristics, and transmission impairments.				
	3	Demonstration of Fourier Analysis, bandwidth calculations, transmission rate concepts, and numerical examples related to channel capacity and bandwidth.				
Practice	1	Draw and explain data communication models, layered architecture, and functions of protocols and standards.				
	2	Identify analog and digital signals, calculate bandwidth of signals and transmission media, and compare signal characteristics.				
	3	Solve numerical problems related to Fourier Analysis, bandwidth, data transmission				

Department of Computer Applications

		rate, and channel capacity calculations.				
MODULE – II						
	Page No.	From: 69 to 96,156-163				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Analog and Digital Signals, Analog and Digital Transmission.			Signal characteristics, Baud Rate, and Bits per Second.	
	2	Modes of Transmission: Parallel and Serial Communication.			Synchronous and asynchronous transmission concepts.	
	3	Multiplexing and Demultiplexing,			Types of Multiplexing: FDM, TDM, and basic applications.	
Transactions	1	Explanation of analog and digital signals, transmission methods, baud rate, and data transfer concepts with examples.				
	2	Discussion on serial and parallel communication methods and comparison of transmission techniques.				
	3	Demonstration of multiplexing and demultiplexing techniques with block diagrams and real-time communication examples.				
Practice	1	Differentiate analog and digital signals and calculate baud rate and bits per second.				
	2	Compare serial and parallel communication using suitable examples.				
	3	Draw multiplexing models and identify different multiplexing techniques used in communication systems.				
MODULE – III						
	Page No.	From: 187 to 201,326 to 331				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Guided and Unguided Transmission Media.			Characteristics and applications of transmission media.	
	2	Network Topology and its types.			OSI Model and functions of different layers.	
	3	LAN, MAN, and WAN concepts, Ethernet, VLAN, Token Ring.			Basic wireless networking concepts, Pure ALOHA and Slotted ALOHA.	
Transactions	1	Explanation of transmission media and comparison of wired and wireless communication methods.				
	2	Discussion on network topologies and detailed explanation of OSI model layers with examples.				
	3	Demonstration of LAN, MAN, WAN technologies and channel access methods including Pure ALOHA and Slotted ALOHA.				
Practice	1	Identify guided and unguided media and compare different transmission media.				
	2	Draw different network topologies and explain OSI model layers.				
	3	Differentiate LAN, MAN, and WAN technologies and solve basic problems related to ALOHA protocols.				
MODULE – IV						
	Page No.	From: 962-972,1081-1092				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Error Classification, Types of Errors.			Basic Error Detection and Correction techniques.	
	2	Simple Coding Schemes.			Fundamentals of Cryptography.	

Department of Computer Applications

	3	Symmetric and Asymmetric Key Encryption, Digital Signatures, Digital Certificates.	SSL/TLS concepts and secure communication.
Transactions	1	Explanation of transmission errors, error classification, and coding schemes with examples.	
	2	Discussion on cryptography concepts and comparison of symmetric and asymmetric encryption techniques.	
	3	Demonstration of digital signatures, digital certificates, SSL/TLS working process, and secure data transmission methods.	
Practice	1	Identify different types of transmission errors and apply simple coding techniques.	
	2	Differentiate symmetric and asymmetric encryption methods with suitable examples.	
	3	Explain the working of digital signatures, certificates, and SSL/TLS in secure communication systems.	

MODULE – V:

	Page No.	From: 436-439,451,528				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Overview of Wireless Networks.		IEEE standards for LAN, MAN, and WAN.		
	2	IEEE 802 standards: 802.3, 802.5, 802.11, Infrared Communication.		Bluetooth, and Wireless LAN concepts.		
	3	Introduction to IPv4 and IPv6 addressing, features.		Comparison between IPv4 and IPv6.		
Transactions	1	Explanation of wireless network concepts and IEEE networking standards with applications.				
	2	Discussion on Ethernet, Token Ring, Wireless LAN, Infrared, and Bluetooth communication technologies.				
	3	Demonstration of IPv4 and IPv6 addressing structure, packet format, and comparison of internet protocols.				
Practice	1	Identify different wireless communication technologies and IEEE standards.				
	2	Compare Bluetooth, Infrared, and Wireless LAN technologies with suitable examples.				
	3	Differentiate IPv4 and IPv6 addressing and solve basic IP addressing problems.				

Course Outcomes:

- CO-1. Explain the principles of data communication systems, including signal types, bandwidth, and transmission rates.
- CO-2. Compare and contrast networking standards and protocols used in LANs, MANs, and WANs, and identify their real-world applications.
- CO-3. Demonstrate understanding of transmission techniques (serial, parallel, analog, digital) and apply multiplexing and demultiplexing methods.
- CO-4. Analyze network errors and implement simple error detection and correction schemes, and describe security mechanisms like encryption and digital signatures.
- CO-5. Evaluate wireless technologies (Bluetooth, Infrared, 802.11) and IP addressing schemes (IPv4/IPv6), and explain their significance in modern communication systems



Department of Computer Applications

Text books:

1. Behrouz A. Forouzan, Data Communications and Networking, McGraw-Hill, 5th Edition

References:

1. James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach, Pearson, 7th Edition
2. William Stallings, Data and Computer Communications, Pearson, 10th Edition
3. William Stallings, Cryptography and Network Security, Pearson, 7th Edition
4. Andrew S. Tanenbaum and David J. Wetherall, Computer Networks, Pearson, 5th Edition.
5. <https://nptel.ac.in/courses/106105081>
6. <https://standards.ieee.org>
7. <https://www.geeksforgeeks.org/computer-network-tutorials/>
8. <https://www.cloudflare.com/learning/ssl/what-is-ssl/>

Foundation of Data Science

Course Code	26MCA242	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:0:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	90 (30:30:0:30)	Total Marks	100
Credits	03	Exam Hours	2 ½

Pre-requisites: Basic knowledge of programming and mathematics.**Course Learning Objectives (CLOs)**

1. To introduce fundamental principles of Data Science.
2. To develop understanding of data pre-processing, analysis, and visualization.
3. To introduce foundational statistics and probability relevant to data analysis.
4. To familiarize students with basic machine learning concepts.
5. To understand ethical concerns in data-driven decision-making.

Unit-1

Pedagogy	Page No.	From: 1-42				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	Week	Hour		Hour		
	1	Introduction to Data Science: The Ascendance of Data, What is Data Science, Motivating Hypothetical: Data Sciencecenter.		A Crash Course in Python: The Zen of Python, Getting Python, Virtual Environments, Whitespace Formatting.		
	2	Python Basics: Modules, Functions, Strings, Exceptions.		Python Data Structures: Lists, Tuples, Dictionaries.		
	3	Python for Data Handling: Error Handling, File I/O Operations.		Python for Data Science		
Transactions	1	Apply Python to implement programs covering strings, lists, dictionaries, and exception handling				
	2	Create a mini-project: Student Record System using Dictionaries and Functions.				
	3	Analyze a real-world case study where Python is used in data science pipelines.				
Practice	1	Setup Python and virtual environments for data science projects				
	2	Write Python programs for file handling (read/write CSV).				
	3	Compare performance of lists vs dictionaries using large datasets.				

Unit-2

	Page No.	From: 43- 73				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Visualizing Data: matplotlib, Bar Charts, Line Charts, Scatterplots.		Linear Algebra: Vectors, Matrices		
	2	Statistics: Describing a single set of data		Central Tendencies, Dispersion		
	3	Correlation, Simpson's Paradox		Some other correlational Caveats, Correlation and Causation		

Department of Computer Applications

Transactions	1	Apply visualization techniques to represent data using bar charts, line graphs, and scatter plots.
	2	Examine datasets using statistical measures such as central tendency and dispersion to understand data behavior.
	3	Analyze relationships between variables using correlation and interpret meaningful insights.
Practice	1	Implement data visualization using matplotlib and verify graphical representations.
	2	Calculate statistical measures using Python and verify the results.
	3	Generate scatter plots and verify correlation between variables.

Unit-3

	Page No.	From: 75 - 99				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Probability Fundamentals: Dependence, Independence		Theorems: Bayes' Theorem, Random Variables		
	2	Conditional Probability.		Distributions: Continuous Distributions, Normal Distribution, Central Limit Theorem		
	3	Statistical Hypothesis Testing, p-Values, Confidence Intervals.		Inference: p-Hacking, A/B Testing Example		
Transactions	1	Apply probability concepts to model uncertainty in real-world situations.				
	2	Use Bayes' Theorem to interpret probabilistic outcomes in data-driven problems.				
	3	Analyze datasets using distribution concepts and hypothesis testing to draw conclusions.				
Practice	1	Simulate probability experiments and verify theoretical outcomes.				
	2	Implement Bayes' Theorem using Python and validate predictions.				
	3	Perform hypothesis testing and verify results using sample datasets.				

Unit-4

	Page No.	From: 129 - 160				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Exploring One-Dimensional Data, Two Dimensions, Many Dimensions		Using NamedTuples, Dataclasses, Cleaning and Munging		
	2	Manipulating Data, Rescaling		An Aside: tqdm, Dimensionality Reduction		
	3	Machine Learning: Modeling, Overfitting/Underfitting,		Bias-Variance Tradeoff.		
Transactions	1	Examine datasets of different dimensions to understand their structure and characteristics.				
	2	Apply data cleaning and preprocessing techniques to prepare datasets for analysis.				
	3	Analyze a dataset to differentiate between overfitting and underfitting.				
Practice	1	Explore datasets using Python and verify data structure and dimensions.				
	2	Implement data cleaning and preprocessing techniques and verify results.				
	3	Train a simple model on a dataset and observe overfitting and underfitting behavior.				

Unit-5

	Page No.	From: 161-174, 215-220, 361-365				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18

Department of Computer Applications

Learning	1	Feature Engineering: Feature Extraction and Selection.	Algorithms: K-Nearest Neighbors, Simple Linear Regression.
	2	Decision Trees: Entropy The Entropy of a Partition Creating a Decision Tree	Data Ethics: Fairness, Bias, Data Protection.
	3	Ethical Implications: Trading off Accuracy and Fairness.	Interpretability.
Transactions	1	How does feature selection impact model performance in KNN and decision trees?	
	2	Compare the interpretability and fairness of KNN vs. decision trees in a classification task.	
	3	Justify when linear regression is more ethical than KNN for high-stakes decisions.	
Practice	1	Implement feature engineering to improve KNN and decision tree accuracy.	
	2	Train KNN and decision tree models and compare their fairness metrics.	
	3	Evaluate and justify the most ethical model (KNN, linear regression, or decision tree) for a high-stakes decision task.	

Course Outcomes:

- CO-1. Understand fundamental concepts of Data Science and its workflow.
 CO-2. Perform data cleaning, transformation and preprocessing.
 CO-3. Apply statistical methods and probability theory in data analysis.
 CO-4. Interpret data through visualizations and extract insights.
 CO-5. Understand basic machine learning models and ethical implications.

Text Books:

1. Joel Grus, Data Science from Scratch: First Principles with Python, O'Reilly Media.

References

1. Wes McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd Edition, O'Reilly Media, 2017.
2. Jake VanderPlas, Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media, 2016.

CYBER SECURITY

Course Code	26MCA243	CIE Marks	70
Teaching Hours/Week (L:T:P:SL)	2:2:0:2	SEE Marks	30
Total Hours of Pedagogy (L:T:P:SL)	(30:30:0:30) 90	Total Marks	100
Credits	3	Exam Hours	2 ½

Pre-requisites: 1.Basic Understanding of Computer Networks 2. Awareness of common cyber security threats and concepts

Course Learning Objectives (CLOs): 1. Students will gain foundational knowledge of cyber security principles, threats, and defensive techniques. 2. Students will learn hands-on skills in identifying, analysing, and mitigating various cyber-attacks. 3. Students will develop the ability to secure computer systems, networks, and applications using best practices and tools. 4. Students will explore cryptographic methods, access control, and authentication mechanisms to enhance data protection. 5. Students will understand legal, ethical, and compliance aspects of cyber security, including standards and policies.

MODULE – I

Pedagogy	Page No.	From: 1 to 38				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	Week	Hour		Hour		
	1	Cybercrime: definition, origin.		Basics of information security		
	2	Cybercriminals		Classification of cybercrime		
	3	Legal perspectives: Indian IT Act (overview)		Global cyber laws		
Transactions	1	Cybercrime concepts and types, Role of cybercriminals				
	2	Indian cyber laws and IT Act, 2000 (basics) ,Legal issues related to hacking				
	3	Global perspectives on cyber laws ,Introduction to cybersecurity tools				
Practice	1	Case study analysis of a real-world cybercrime incident				
	2	Identification of cybersecurity tools for threat detection and prevention				
	3	Report on cybercriminal roles and tracking methods				

MODULE – II

	Page No.	From: 45 to 77				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Introduction to cyber attack planning		Social engineering concepts and techniques		
	2	Cyber stalking		Cybercrimes in public access points (cyber cafés)		
	3	Botnets: concept, working, and role in cybercrime		Cloud computing vulnerabilities (basic idea)		
Transactions	1	Attack vectors and methods used by cybercriminals				
	2	Social engineering attacks and psychological manipulation				
	3	Botnets and their use in large-scale cyber attacks				

Department of Computer Applications

Practice	1	Simulation of social engineering attack (role-play / scenario-based)				
	2	Case study analysis: cyber stalking and botnet attacks				
	3	Case study on cyber stalking and legal actions				
MODULE – III						
	Page No.	From: 125 to 180				
	Number of hours of engagement	L	T	P	SL	Total
Learning	1	Introduction to cybercrime tools and techniques		Proxy servers and anonymizers ,Phishing attacks		
	2	Malware types: viruses, worms, trojans		Spyware, keyloggers , Password cracking concepts		
	3	Network attacks: DoS, DDoS, wireless attacks ,Web attacks: SQL		injection, buffer overflow (basics) ,Steganography (basic concept		
Transactions	1	Techniques of phishing and identity theft ,Use of anonymization tools				
	2	Malware behavior and attack mechanisms ,Password cracking methods				
	3	Network and web-based attack methods (DoS/DDoS, SQL Injection) ,Basic prevention techniques				
Practice	1	Simulation of phishing email (awareness-focused)				
	2	Network monitoring using Wireshark (packet analysis basics)				
	3	Controlled password testing using tools (John the Ripper / Hashcat – basic demo) ,Case study on DDoS attack and malware analysis				
MODULE – IV						
	Page No.	From: 185 to 220				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Introduction to phishing and identity theft		Methods of phishing and common phishing techniques		
	2	Types of phishing: spear phishing, email scams		Phishing toolkits (basic idea) and detection concepts		
	3	Identity theft: PII (Personally Identifiable Information)		Types, techniques, and countermeasures		
Transactions	1	Phishing attack process and social engineering aspects				
	2	Identification of phishing attacks (URLs, email headers, content clues)				
	3	Identity theft methods and prevention strategies				
Practice	1	Simulation of phishing scenarios (awareness-based)				
	2	Analysis of phishing emails and identification of red flags				
	3	Table of identity theft types and methods				
MODULE – V:						
	Page No.	From: 317 to 389				
	Number of hours of engagement	L	T	P	SL	Total
		6	6	0	6	18
Learning	1	Introduction to computer forensics and digital forensic science		Need and importance of cyber		

Department of Computer Applications

			forensics
	2	Digital evidence and forensic life cycle	Chain of custody concepts
	3	Network forensics (basic idea)	Challenges in computer forensics
Transactions	1	Steps in forensic investigation process	
	2	Collection and handling of digital evidence	
	3	Analysis techniques and reporting in forensics	
Practice	1++	Demonstration of data recovery using forensic tools (Autopsy / FTK Imager – basic)	
	2	Metadata extraction and analysis for forensic clues	
	3	Report on forensic life cycle	

Course Outcomes:

- CO-1. Describe the nature of cybercrime, identify various types of cybercriminals, and explain the legal perspectives with reference to Indian and global laws.
- CO-2. Analyze how cybercrimes are planned, including methods such as social engineering, cyber stalking, and botnet-based attacks, and examine the threats in cloud environments.
- CO-3. Identify and assess various cyber attack techniques including phishing, malware, DoS/DDoS attacks, SQL injection, and buffer overflow, and understand methods used to compromise wireless networks.
- CO-4. Evaluate different phishing strategies and identity theft techniques, and suggest appropriate countermeasures to safeguard personal and organizational data.
- CO-5. Apply the principles of computer forensics, digital evidence handling, forensic investigation life cycle, and network forensics to approach and analyze cybercrime incidents effectively.

Text Books:

1. Cyber Security: Understanding Cyber Crimes, Computer Forensics and Legal Perspectives
Authors: Nina Godbole and Sunit Belapure

References:

1. Computer Security: Principles and Practice
2. Authors: William Stallings & Lawrie Brown