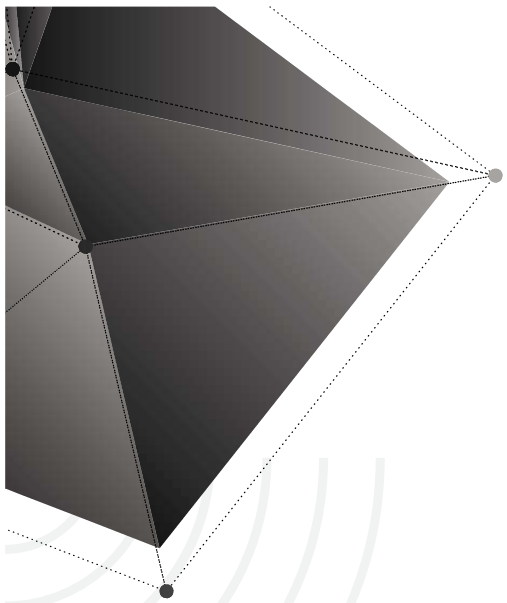
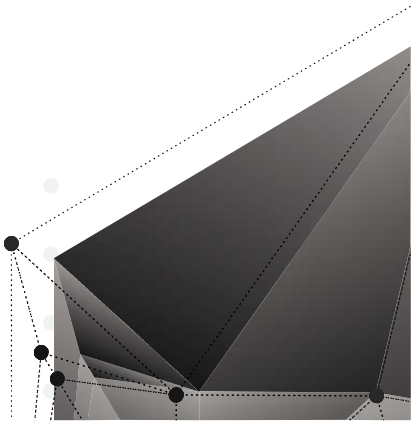


FOURTH SEMESTER

'ARTIFICIAL INTELLIGENCE
&
MACHINE LEARNING'



ALGORITHMS

Course Code:	434001
Course Title	Algorithms
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Understand the fundamental concepts of data structures and their relevance in programming, including sets, multisets, stacks, queues, and graphs.
2. Develop proficiency in implementing and analyzing various sorting algorithms, such as bubble sort, selection sort, insertion sort, merge sort, and quicksort.
3. Acquire knowledge of searching techniques using symbol tables, binary search trees, balanced search trees, and hash tables.
4. Gain expertise in working with graphs, including identifying paths, cycles, spanning trees, and applying algorithms like topological sorting, minimum spanning trees, and shortest path algorithms (such as Dijkstra's algorithm).
5. Familiarize yourself with string manipulation techniques, including string sorting, substring search, regular expressions, and elementary data compression methods.
6. Apply problem-solving skills by designing and implementing efficient algorithms using the learned data structures and algorithms for real-world scenarios.

COURSE CONTENTS

UNIT - 1 : Fundamentals

Programming Models. Data Abstraction. Sets, Multisets, Stacks, Queues. Overview of various analysis techniques of algorithms.

UNIT - 2 : Sorting

The sorting problem. Bubble sort, Selection sort, Insertion sort, Merge sort, Quicksort.

UNIT - 3 : Searching

Symbol Tables, Binary Search Trees, Balanced Search Trees. Hash Tables.

UNIT - 4 : Graphs

Definition of a directed and undirected graph. Paths, Cycles, spanning trees.

Topological Sorting. Minimum Spanning Tree algorithms. Shortest Path algorithms.

UNIT - 5 : Strings

String Sort. Tries. Substring Search. Regular Expressions. Elementary Data compression.

Practical Outcomes: Upon completion of the course, student will be able to:

1. Understand and implement various searching algorithms.
2. Develop proficiency in implementing and analyzing different sorting algorithms.
3. Acquire knowledge and skills in implementing advanced sorting algorithms.
4. Gain expertise in implementing graph algorithms.
5. Apply problem-solving skills by implementing algorithms.
6. Analyze the time complexity, efficiency, and performance of the implemented algorithms, and evaluate their suitability for different problem scenarios.

List of Practicals:

1. Write a program to implement Sequential Search.
2. Write a program to implement Binary Search.
3. Write a program to implement Insertion Sort.
4. Write a program to implement Bubble Sort.
5. Write a program to implement Selection Sort.
6. Write a program to implement Merge Sort.
7. Write a program to implement Quick Sort.
8. Write a program to implement Counting Sort.

Reference Books :

1. Algorithms, Sedgewick and Wayne, Pearson
2. Introduction to Algorithms, Cormen, Leiserson, Rivest and Stein. MIT Press
3. Introduction to Theory of Computation, Sipser Michael, Cengage Learnng.
4. Design & Analysis of Algorithms, Gajendra Sharma, Khanna Publishing House

*****

OBJECT ORIENTED PROGRAMMING using 'C++'

Course Code:	434002
Course Title	Object oriented Programming using 'C++'
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. To know about data abstraction
2. Explain the concepts of Object-oriented programming.
3. Write simple applications using C++.
4. Discuss the different methods of organizing large amount of data.

COURSE CONTENTS

1. Introduction :

- Understanding the concept of object-oriented programming (OOP)
- Exploring the need for object-oriented programming
- Characteristics of object-oriented languages (with a focus on C++)

2. C++ Programming Basics:

- Using cout for output and cin for input operations
- Formatting output using directives like endl, setw, setprecision, and fixed
- Working with bool type and performing input/output operations with boolean values
- Utilizing setw manipulator to control the width and alignment of output fields
- Type conversions & their application in C++ programs
- Handling common I/O errors using exception handling techniques

3. Functions :

- Returning values from functions
- Working with reference arguments
- Overloaded functions
- Understanding inline functions
- Utilizing default arguments
- Returning values by reference

4. Objects and Classes:

- Understanding core object concepts.
- Classes in C++
- Exploring the role of objects as physical objects and data types
- Working with constructors
- Using objects as function arguments
- Understanding the default copy constructor
- Structures and classes
- Objects, memory, and static class data
- Working with const and classes

5. Arrays and Strings:

- Fundamentals of arrays
- Arrays as class member data
- Working with arrays of objects
- Exploring the standard C++ String class

6. Operator Overloading:

- Overloading unary operations
- Overloading binary operators
- Data conversion and pitfalls of operator overloading
- Exploring explicit and mutable keywords

7. Inheritance:

- Concept of inheritance
- Derived classes and base classes
- Working with derived class constructors & member functions
- Aggregation: Classes within classes

8. Pointers:

- Introduction to Addresses and pointers
- Using the address of operator and pointers with arrays
- Pointers and fractions
- Pointers and C-style strings
- Pointers to objects
- Debugging pointers

9. Virtual Functions:

- Elementary idea of various virtual functions
-

10. Streams and Files:

- Streams classes
- Disk file I/O with streams
- File pointers
- Error handling in file I/O with member functions
- Overloading the extraction and insertion operators
- Printer output

11. Templates and Exceptions:

- Function templates
- Class templates
- Exception handling

12. The Standard Template Library (STL): Overview of standard template library.

PRACTICAL OUTCOMES: Upon completion of the course, student will be able to:

1. To know about data abstraction
2. Explain the concepts of Object oriented programming.
3. Write simple applications using C++.
4. To demonstrate different linearity in data structures.
5. Discuss the different methods of organizing large amount of data.

C ++ Practicals: (List of Programs) :-

1. Programs related to basic input/output.
2. Programs related to variables, strings, numbers
3. Programs related to conditions
4. Programs related to switch statement
5. Programs related to While Loops and For loop
6. Programs related to Break/Continue statement
7. Programs related to create references and pointers
8. Programs related to Functions
9. Programs related to Classes
10. Case study of application areas of C++.

References:

1. BhushanTrivedi, “Programming with ANSI C++, A Step-By-Step approach”, Oxford University Press, 2010.
2. Goodrich, Michael T., Roberto Tamassia, David Mount, “Data Structures and Algorithms in C++”, 7th Edition, Wiley. 2004.
3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms", Second Edition, McGraw Hill, 2002.
4. BjarneStroustrup, “The C++ Programming Language”, 3rd Edition, Pearson Education, 2007.
5. Ellis Horowitz, SartajSahni and Dinesh Mehta, “Fundamentals of Data Structures in C++”, Galgotia Publications, 2007
6. Big C++ - Wiley India
7. C++: The Complete Reference- Schildt, McGraw-Hill Education (India)
8. C++ and Object Oriented Programming – Jana, PHI Learning.
9. Object Oriented Programming with C++ - Rajiv Sahay, Oxford
10. Mastering C++ - Venugopal, McGraw-Hill Education (India)

*****

INTRODUCTION TO OPERATING SYSTEMS

Course Code:	434003
Course Title	Introduction to Operating System
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Understand the basic concepts and architecture of UNIX/LINUX operating systems.
2. Gain knowledge of process management, including process concepts, operations, and inter-process communica-tion.
3. Learn about different process scheduling algorithms and their implementations.
4. Develop an understanding of memory management techniques such as allocation, swapping, paging, and segmentation.
5. Explore file management concepts, including file access methods, directory structure, and file system implementation.
6. Acquire knowledge of OS security principles, including authentication, access control, and system logs.

COURSE CONTENTS

UNIT - 1:

1.1 Overview of Operating System

1.2 Basic concepts

1.3 UNIX/LINUX Architecture

1.4 Kernel

1.5 Services and systems calls

1.6 System programs.

UNIT - 2 :

2.1 Overview of Process management Concept

2.2. Process scheduling Methods

2.3 Multi- threaded programming

2.4 Memory management Techniques

2.5 Virtual memory

UNIT - 3 :

3.1 File management

- 3.1.1 Concept of a file
- 3.1.2 Access methods

3.2 Directory structure

3.3 Overview of File system structure and implementation

3.4 Different types of file systems

UNIT - 4 :

4.1 I/O system

4.2 Mass storage structure

4.2.1 Overview

4.2.2 Disk structure

4.2.3 Disk attachment

4.3 Basic Idea of Disk scheduling algorithms

4.4 Swap space management

4.5 Raid.

UNIT - 5:

5.1 OS Security

5.2 Authentication

5.3 Access Control

5.4 Access Rights

5.5 System Logs

Practical Outcomes: Upon completion of the course, student will be able to:

1. Simulate CPU scheduling algorithms (FCFS, SJF, Round Robin, Priority) using a C program.
2. Implement the producer-consumer problem simulation using semaphores in C.
3. Simulate the Dining-philosophers problem to understand synchronization and resource allocation.
4. Implement memory allocation techniques (MVT, MFT) in a simulation.
5. Simulate contiguous memory allocation techniques (Worst Fit, Best Fit, First Fit) using a C program.
6. Implement page replacement algorithms (FIFO, LRU, Optimal) in a simulation.
7. Simulate different file organization techniques (Single Level Directory, Two Level Directory).

List of Practicals:

1. Simulate the following CPU scheduling algorithms.

   (a) FCFS (b) SJF (c) Round Robin (d) Priority.

2. Write a C program to simulate producer-consumer problem using Semaphores

3: Write a C program to simulate the concept of Dining-philosophers problem.

4. Simulate MVT and MFT.

5. Write a C program to simulate the following contiguous memory allocation Techniques

   (a) Worst fit (b) Best fit (c) First fit.

6. Simulate all page replacement algorithms

   (a) FIFO (b) LRU (c) OPTIMAL

7. Simulate all File Organization Techniques

8. Simulate all file allocation strategies

   (a) Sequential (b) Indexed (c) Linked.

11. Write a C program to simulate disk scheduling algorithms.

Reference Books

1. Operating System Concepts, Silberschatz and Galvin, Wiley India Limited
2. UNIX Concepts and Applications, Sumitabha Das, McGraw-Hill Education
3. Operating Systems, Internals and Design Principles, Stallings, Pearson Education, India
4. Operating System Concepts, Ekta Walia, Khanna Publishing House
5. Modern Operating Systems, Andrew S. Tanenbaum, Prentice Hall of India
6. Operating systems, Deitel & Deitel, Pearson Education, India

*****

ARTIFICIAL INTELLIGENCE

Course Code:	434004
Course Title	Artificial Intelligence
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Understand the definition, goals, and history of Artificial Intelligence (AI).
2. Explore the applications of AI in various domains.
3. Gain knowledge of different types of agents, such as simple reflex agents, model-based reflex agents, and goal-based agents.
4. Understand the nature of environments and their properties.
5. Develop proficiency in search algorithms, including brute-force strategies like breadth-first search and depth-first search, as well as heuristic search algorithms and local search techniques.
6. Acquire knowledge of fuzzy logic systems, including membership functions, fuzzification/defuzzification processes, and their applications.
7. Gain an understanding of neural networks, including their basic structure, the concept of perceptron, and the back-propagation algorithm.

COURSE CONTENTS

Unit - 1: Introduction to Artificial Intelligence

- Artificial Intelligence (AI) definition
- Goals of AI
- History of AI
- Applications of AI

Unit - 2 : Agents and Environments

- Agent Terminology, Types of Agents – Simple Reflex Agents, Model Based Reflex Agents, Goal Based Agents
- Nature of Environments, Properties of Environments

Unit - 3 : Search Algorithms

- Brute Force Search Strategies – Breadth First Search, Depth First Search.
- Heuristic Search Strategies, Local Search Algorithms.

Unit - 4 : Fuzzy Logic Systems

Introduction to Fuzzy Logic and Fuzzy systems,

- Membership functions,
- Fuzzification/Defuzzification

Unit - 5 : Neural Networks

Basic structure of Neural Networks

- Perceptron
- Back-propagation

PRACTICAL OUTCOMES: Upon completion of the course, student will be able to:

1. Select a problem statement, formulate it, and provide a PEAS description relevant to AI applications.
2. Implement PROLOG programs for logical reasoning and problem-solving.
3. Develop proficiency in search algorithms by implementing BFS, DFS, and informed A* search methods.
4. Write programs to implement Hill Climbing/Minimax search algorithms for optimization or game-playing scenarios.
5. Apply AI techniques to predict housing prices, investigate the Enron scandal, and predict stock prices.
6. Implement intelligent game-playing programs, such as Tic-Tac-Toe and the Water-Jug problem, using AI algorithms.

List of Practicals:

1. Select a problem statement relevant to AI, formulate the problem and give PEAS Description
2. Implement simple PROLOG programs.
3. Implement any search strategy algorithm to reach goal state
4. Write a Program to implement BFS/DFS search method.
5. Write a Program to implement informed A* search method.
6. Write a Program to implement Hill Climbing/Minimax search algorithm.
7. Write a Program to implement for Stock Price Prediction

Reference Books

1. "Artificial Intelligence: A Hands-On Approach" by Amit Konar
2. "Artificial Intelligence for Engineering Applications" by T. Veerakumar
3. "Practical Artificial Intelligence Programming with Java" by Mark Watson

Suggested Learning Resources:

1. Artificial Intelligence By Example: Develop machine intelligence from scratch using real artificial intelligence use cases Denis Rothman Packt Publishing ISBN – 978-1788990547

*****

MATHEMATICAL & STATISTICAL FOUNDATIONS

Course Code:	434005
Course Title	Mathematics & Statistical Foundation
No. of Credits	4 (TH:4,T:0,P:0)

COURSE OUTCOMES: At the end of the course, the student will be able to:

1. Demonstrate an understanding of greatest common divisors, prime factorization, and congruences, including the application of the Euclidean algorithm and the Chinese remainder theorem.
2. Apply simple linear regression and correlation analysis techniques, including least squares estimation and inferences concerning regression coefficients.
3. Analyze and apply probability distributions, including discrete and continuous distributions such as the normal distribution and its applications.
4. Understand the concepts of random sampling, sampling distributions, and the central limit theorem, and perform estimation and hypothesis testing for means, variances, and proportions.
5. Gain knowledge of stochastic processes and Markov chains, including transition probabilities, Markov chains, and steady state conditions.
6. Apply statistical inference techniques, including classical methods of estimation, prediction intervals, tolerance limits, and maximum likelihood estimation for comparing means and proportions.

COURSE CONTENTS

1. **Overview of Greatest Common Divisors and Prime Factorization**
   - 1.1 Greatest common divisors.
   - 1.2 Congruences.

2. **Simple Linear Regression and Correlation**
   - 2.1 Introduction to Linear Regression, The Simple Linear Regression Model (Overview Only)
   - 2.2 Simple Linear Regression Case Study

3. **Continuous Probability Distributions**
   - 3.1 Normal Distribution, Areas under the Normal Curve,
   - 3.2 Applications of the Normal Distribution.
   - 3.3 Random Sampling.

4. **Estimation & Tests of Hypotheses:**
   - 4.1 Introduction, Statistical Inference, Classical Methods of Estimation.
   - 4.2 Estimating the Mean, Standard Error of a Point Estimate.
   - 4.3 Limits, Estimating the Variance.

5. **Stochastic Processes and Markov Chains**
   - 5.1 Introduction to Stochastic processes, Markov Chains.

REFERENCES

1 Kenneth H. Rosen, Elementary number theory & its applications, sixth edition, Addison Wesley, ISBN 978 0-321-50031-1

2 Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, Keying Ye, Probability & Statistics for Engineers & Scientists, 9th Ed. Pearson Publishers.

3 S.D. Sharma, Operations Research, Kedarnath and Ramnath Publishers, Meerut, Delhi

*****

INTRODUCTION TO DBMS

Course Code:	434006
Course Title	Introduction to DBMS
No. of Credits	5 (TH:4,T:0,P:2)

COURSE OUTCOMES: At the end of the course, the student will be able to:

1. Understand the fundamental concepts of databases, advantages of DBMS, and the structure of a standard DBMS.
2. Conceptualize, design, and implement data structures using prominent data models, emphasizing the Entity-Relationship Model.
3. Acquire skills to implement and manage relational databases, apply security measures, and proficiently use query languages.
4. Grasp the principles of database normalization, functional dependencies, and apply them to design efficient databases.
5. Develop proficiency in SQL and PL/SQL for effective data manipulation, control structures, error handling, and understanding advanced features like cursors and triggers.
6. Understand and apply foundational concepts in database security, forensics, and auditing, ensuring the integrity and security of database applications.

COURSE CONTENTS

Unit-1: Introduction to Database Systems and Data Modelling:

- Fundamental concepts of databases and advantages over traditional file systems.
- Structure and components of a standard Database Management System (DBMS).
- Introduction to prominent Data Models with a focus on the Entity-Relationship (E-R) Model.
- Overview of the Client-Server Architecture in databases.
- Detailed study of the E-R Model: Entities, Attributes, Relationships, and their degrees.

Unit-2: The Relational Data Model: Concepts and Security Measures

- Introduction to the Relational Model: Attributes, domains, and key concepts.
- Understanding of primary and candidate keys and associated integrity constraints.
- Study of data security, access control, and authorization in databases.
- Introduction to foundational Query Languages: Relational Algebra and Views.

Unit-3: Structured Query Language (SQL) and Procedural Language/SQL (PL/SQL)

- **Basic SQL operations:** Creating, updating, deleting tables, and implementing constraints.
- **Introduction to advanced SQL concepts:** Set operations, aggregate functions, and joins.

- **Basics of PL/SQL:** Structure, variables, control mechanisms, and error handling.
- **Overview of specialized PL/SQL features:** Cursors, triggers, and packages.

Unit-4: Principles of Relational Database Design and Storage Systems

- Importance of database normalization and associated challenges.
- Basics of Functional Dependencies and steps to normalization.
- Introduction to file organization in databases.
- Fundamental concepts of indexing, hashing, and query optimization.

Unit-5: Fundamentals of Query and Transaction Processing

- Strategies for efficient query processing.
- Basics of transactions in databases and their properties.
- Understanding concurrency in transactions and related concepts.

Unit-6: Case Study on Database Security and Forensics

- Study of security models applicable to databases.
- Introduction to database auditing, activity monitoring, and forensic measures

Practical Outcomes: At the end of the course:

1. Student will be proficient in defining, altering, and managing database structures using Data Definition Language (DDL) commands.
2. Student will grasp the importance of data integrity through constraints like PRIMARY KEY, FOREIGN KEY, etc., and will leverage group functions to derive key insights from datasets.
3. Student will master data manipulation using DML commands, set operators, and essential string functions to optimize and analyse table data.
4. Student will become competent in managing date and time data types, utilizing SQL's specialized functions for time-sensitive queries and analyses.
5. Student will understand and apply conditional constructs like if-then-else, ensuring dynamic and responsive database operations.
6. Student will gain foundational skills in PL/SQL, especially in variable and constant declarations, facilitating efficient temporary data storage and manipulation in their programs.

List of Practicals:

1. Creating and Executing DDL in SQL.
2. Creating and Executing Integrity constraints in SQL.
3. Creating and Executing DML in SQL.
4. Executing relational, logical and mathematical set operators using SQL.
5. Executing group functions.
6. Executing string operators & string functions.
7. Executing Date & Time functions.

8. Executing Data Conversion functions.
9. Program using if-then-else in PL/SQL.
10. Program for declaring and using variables and constant using PL/SQL.

Text and Reference Books:

1. Fundamentals of Database Systems, Elmasri & Navathe, Pearson Education
2. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata McGraw Hill.
3. Database System Concepts, Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw- Hill, New Delhi, India.
4. Introduction to Database Systems, C .J. Date, Pearson Education.
5. Introduction to SQL, Rick F. VanderLans, Pearson Education

*****

'Elective 1-1'
SYSTEM SOFTWARE

Course Code:	434007
Course Title	System Software
No. of Credits	4 (TH:4,T:0,P:0)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Understand the primary concepts and functionalities of system software, recognizing the significance of language processors in system software development.
2. Develop a foundational understanding of Assembly Language Programming and the basic operational mechanisms of assemblers.
3. Acquire knowledge about macros, their definitions, expansion processes, and the importance of macro preprocessors in optimizing software development.
4. Gain insights into the functionalities and differences between interpreters and compilers, understanding the primary phases of compilation like scanning and parsing.
5. Understand the roles and functionalities of linkers, comprehending the key principles behind software relocation and linking.
6. Familiarize with essential software development tools, including editors, debug monitors, and user interfaces, enhancing the efficiency and effectiveness of the software development process.

COURSE CONTENTS

1. **System Software and Tools Overview:**

- Introduction to Language Processors and their fundamental activities.
- Basic principles behind Language Specification & Processing.
- Essential data structures in language processing.
- Introduction to foundational software tools: editors, debug monitors, and user interfaces.

2. **Understanding Assemblers:**

- Basics of Assembly Language Programming.
- Introduction to the structure of assemblers.
- Simplified design concepts of a two-pass assembler.
- Overview of a single-pass assembler tailored for common systems.

3. **Macros and Their Processing:**

- Fundamental concepts of Macro Definitions and Calls.
- Basics of Macro Expansion.
- Introduction to nested macro calls and advanced macro functionalities.
- Overview of designing a simple macro preprocessor.

4. **Introduction to Interpreters and Compilers:**

- Basics of Interpreters and their usage.
- Introduction to the primary functions of compilers: scanning and parsing.
- General aspects and overview of the compilation process.

5. Basics of Linkers and Loaders:

- Introduction to the roles and functionalities of linkers.
- Foundational concepts of relocation and linking in software.
- Basic design principles of a linker.
- Overview of self-relocating programs and linkers suited for common operating systems.

Text & Reference Books:

1. D. M. Dhamdhere, "Systems Programming and Operating Systems", Second Revised Edition, Tata McGraw-Hill, 1999.
2. Leland L. Beck, "System Software – An Introduction to Systems Programming", 3rd Edition, Pearson Education Asia, 2000.
3. Santanu Chattopadhyay, "System Software", Prentice-Hall India, 2007
4. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, "Compilers: Principles, Techniques, and Tools",2nd Edition, Pearson Education Asia

*****

‘Elective 1-2’
COMPUTER NETWORKS

Course Code:	434008
Course Title	Computer Networks
No. of Credits	4 (TH:4,T:0,P:0)

COURSE OUTCOMES: Upon completion of the course, student will be able to:

1. Understand the fundamental concepts and principles of computer networks.
2. Design and configure local area networks (LANs) and wide area networks (WANs).
3. Implement and troubleshoot network protocols and services.
4. Apply security measures in computer networks.
5. Collaborate effectively in teams to design and implement network solutions.
6. Analyze and optimize network performance.

COURSE CONTENTS

Unit - 1

1.1 Introduction to computer networks

1.2 Network Models

1.3 OSI Reference Model

1.4 TCP/IP Model

Unit - 2

2.1 Transmission media

- 2.1.1 Principles
- 2.1.2 Issues and examples

2.2 Wired media – coaxial, utp, stp, fiber optic cables

2.3 Wireless media – hf, vhf, uhf, microwave, ku band

2.4 Network topologies

2.5 Data link layer

- 2.5.1 Design issues
- 2.5.2 Example protocols (ethernet, wlan, bluetooth)
- 2.5.3 Switching techniques

Unit - 3

3.1 Network layer

- 3.1.1 Design issues
- 3.1.2 Example protocols (ipv4)

3.2 Routing

- 3.2.1 Principles/issues
- 3.2.2 Algorithms (distance-vector, link-state) and protocols (rip, ospf)

Unit - 4

4.1 Transport layer

4.1.1 Design issues

4.1.2 Example protocols (tcp)

4.2 Application layer protocols (smtp, dns)

Unit - 5

5.1 Functioning of Network Devices

5.1.1 NIC, Hub, Switch, Router, WiFi Devices

5.2 Network Management System.

References :

1. Computer Networks, 4th Edition (or later), Andrew S. Tanenbaum, PHI
2. TCP/IP Illustrated, Volume-1, W. Richard Stevens, Addision Wesley
3. Data and Computer Communications, William Stallings, PHI
4. An Engineering Approach to Computer Networking, S. Keshav, Addision Wesley/Pearson
5. An Integrated Approach to Computer Networks, Bhavneet Sidhu, Khanna Publishing House

*****

'Audit Course'
ESSENCE OF INDIAN TRADITIONAL KNOWLEDGE

Course Code	AS401
Course Title	Essence of Indian Traditional Knowledge
No. of Credits	0 (TH:2,T:0,P:0)

COURSE OUTCOMES: After completion of this course, student will be able to:

1. Develop a comprehensive understanding of the essence of Indian knowledge and tradition.
2. Explore the rich philosophical systems of ancient India and their relevance today.
3. Gain familiarity with the Vedic literature and scriptures, and appreciate their wisdom.
4. Analyze Indian epics and mythology to understand their cultural and spiritual significance.
5. Learn and apply principles of yoga, meditation, and mindfulness for personal well-being.
6. Discover the principles and practices of Ayurveda and natural healing for holistic health.

COURSE CONTENTS

1. Introduction to Indian Knowledge and Tradition
2. Ancient Indian Philosophical Systems
3. Vedic Literature and Scriptures
4. Indian Epics and Mythology
5. Yoga, Meditation, and Mindfulness Practices
6. Ayurveda and Natural Healing Systems
7. Indian Classical Arts and Music
8. Indian Architecture and Sculpture
9. Indian Festivals and Rituals
10. Ethical and Moral Values in Indian Culture

References /Suggested Learning Resources:

1. "Indian Philosophy: A Very Short Introduction" by Sue Hamilton
2. "The Vedas: An Introduction to Hinduism's Sacred Texts" by Roshen Dalal
3. "The Ramayana: A Shortened Modern Prose Version of the Indian Epic" by R.K. Narayan
4. "The Upanishads" translated by Eknath Easwaran
5. "Autobiography of a Yogi" by Paramahansa Yogananda
6. "Ayurveda: The Science of Self-Healing" by Dr. Vasant Lad.

*****

MINOR PROJECT WORK

Course Code:	AS402
Course Title	Minor Project Work
No. of Credits	2 (TH:0,T:0,P:4)

OBJECTIVE:

The Minor Project work is an integral part of the Engineering Diploma program, designed to provide students with an opportunity to apply theoretical knowledge gained throughout their studies to real-world engineering challenges. This module aims to foster creativity, problem-solving abilities, and practical skills essential for successful engineering professionals.

PRACTICAL OUTCOMES: After undergoing the minor project work, the student will be able to:

1. Understand the practical applications of engineering concepts in real-world scenarios.
2. Develop hands-on experience in designing, implementing, and testing engineering projects.
3. Enhance problem-solving and critical thinking skills through project execution.
4. Improve documentation and presentation skills for effective project communication.

GENERAL GUIDELINES:

1. Introduction to Minor Projects

- Overview of the module's purpose and objectives
- Importance of practical application in engineering
- Understanding the project life cycle and its stages

2. **Project Ideation and Proposal Development**

- Identifying engineering problems and project ideas
- Formulating clear project objectives and scope
- Developing a comprehensive project proposal

3. **Project Planning and Management**

- Creating a project plan with defined milestones and timelines
- Resource allocation and budgeting for the project
- Risk assessment and mitigation strategies

4. **Engineering Design and Analysis**

- Principles of engineering design and problem-solving
- Conducting feasibility studies and simulations (if applicable)
- Engineering analysis techniques and tools

5. **Prototyping and Implementation**

- Hands-on development of project prototypes
- Conducting experiments and data collection
- Troubleshooting and problem-solving during implementation

6. **Project Documentation and Reporting**

- Techniques for effective project documentation
- Writing comprehensive project reports and design documentation
- Organizing and presenting project data

7. **Project Presentation and Communication**

- Principles of effective communication in engineering
- Preparing engaging & informative project presentations
- Addressing questions & feedback during the presentation

8. **Project Evaluation and Assessment**

- Criteria for evaluating project success and achievement of objectives
- Conducting fair and unbiased project assessments
- Peer evaluations and constructive feedback.

ACTIVITIES AND EXECUTION GUIDELINES

1. **Project Proposal Submission:**

Students will submit their project proposals to the assigned mentors. The proposals should be well-structured, indicating the project's significance, expected outcomes, resources required, and a preliminary plan of action.

2. **Project Execution:**

During this period, students will work on their projects under the guidance of their mentors. They are encouraged to employ innovative techniques and apply engineering principles to achieve project objectives successfully.

3. **Project Documentation:**

Students will submit their final project reports and related documentation. The documentation should encompass all project phases, methodologies, experimental data, analysis, and outcomes.

4. Project Presentation:

Each student will deliver a comprehensive presentation to a panel of evaluators, showcasing their project's key aspects, results, and conclusions.

ASSESSMENT CRITERION

1. Project Proposal and Objective (10%)

Students are required to submit a comprehensive project proposal outlining the problem statement, objectives, scope, and methodology of the project. This component will account for 10% of the total marks.

2. Project Implementation (60%)

The core of the assessment will be based on the successful implementation of the project. Students will be evaluated on their ability to execute the project plan, adhere to timelines, and demonstrate practical engineering skills. This segment will carry 60% of the total marks.

3. Documentation (15%)

Proper documentation is vital to effective project management and communication. Students will be evaluated on the clarity, completeness, and organization of their project reports, design diagrams, code (if applicable), and any other relevant material. This component will contribute 15% of the total marks.

4. Project Presentation (15%)

Communication and presentation skills are crucial for engineers to articulate their ideas effectively. Students will be assessed based on their ability to present their project's

objectives, methodology, results, and conclusions in a clear and concise manner. This segment will be worth 15% of the total marks.

The Minor Project module is a pivotal component of the Engineering Diploma program that provides students with hands-on experience, encourages critical thinking, and prepares them for real-world engineering challenges. By adhering to the module guidelines and distribution of marks, students can excel in their projects and demonstrate their engineering prowess effectively.

*****