NOTIFICATION

No. 104 /2025 Dated : 19 /07/2025

Subject : Implementation of new syllabus of the subjects of Semester III & IV of the new program Five Years Integrated Course of Master of Computer Application (MCA) as per NEP-2020 guidelines.

It is notified for general information of all concerned that the authorities of the University have accepted to implement the new syllabus of the subjects of Semester III & IV of the new program Five Years Integrated Course of Master of Computer Application (MCA) as per NEP-2020 guidelines to be implemented from the academic session 2025-2026 onwards as mentioned below:

Sd/-
(Dr.A.M.Asanare)
Registrar
Sant Gadge Baba Amravati University

Syllabus of Semester III & IV of MCA 5 Years Integrated Course

SEMESTER III

3IMC201 MATHEMATICAL FOUNDATION TO COMPUTER SCIENCE- I

Course Objectives:

1. To understand and apply basic set operations and relation properties.
2. To explore types of functions and their roles in modeling.
3. To apply the binomial theorem and recurrence relations in problem-solving.
4. To develop logical reasoning using propositions and truth tables.
5. To understand graph structures and their real-world applications.
6. To study trees and spanning trees for computing and network optimization.

Course Outcomes:

Upon completing this course, students will be able to:

1. Represent and analyze sets and relations using visual and algebraic tools.
2. Classify functions and perform operations like composition and inversion.
3. Solve binomial expansions and linear recurrence relations.
4. Construct truth tables and test logical equivalence.
5. Analyze graphs and identify Eulerian and Hamiltonian paths.
   Apply tree properties and Kruskal's algorithm to spanning trees.

SYLLABUS:

Unit 1: Sets, Relations, and Their Applications (7 Hours)

Introduction to Sets: Definitions, Subsets, and Set Notation Set Operations: Union, Intersection, Difference, Complement, Properties of Set Operations Venn Diagrams and Their Applications, Cartesian Product of SetsRelations: Definition and Types of Relations Properties of Relations: Reflexive, Symmetric, Transitive, Equivalence Relations and Their Properties Partition of a Set through Equivalence Relations, Closures of Relations. Example.

Unit II: Fundamentals of Functions (7 Hours)

Fundamentals of Functions: Definition of a function, Domain and range of a function, Types of mappings: One-to-one (Injective) functions, Onto (Surjective) functions, Bijective functions, Types of Functions,Constantfunctions,Identityfunctions,Polynomialfunctions,Exponentialfunctions,Logarithmicfunctions,Operations on Functions, Composite functions, Inverse of a function, example on functions.

Unit III: Binomial Expansion and Recurrences (8 Hours)

Binomial Coefficients Definition and Properties of Binomial Coefficients, Calculation of Binomial Coefficients using Pascal's Triangle, Statement and Proof of the Binomial Theorem for Positive Integers, General Term and Middle Term in a Binomial Expansion, Applications of the Binomial Theorem in Solving Algebraic Problems **Introduction to Recurrence Relations**, Modeling Real-World Problems Using Recurrence Relations, FibonacciNumbers,Tower of Hanoi Problem, Simple Economic and Algorithmic ModelsSolving Linear Recurrence Relations First-Order and Second-Order Linear Recurrence Relations with Constant Finding General and Particular Solutions, Coefficients, Solving Techniques Using the Characteristic Equation Method,Applications in Algorithm Analysis and Computer Programming.

Unit IV: Logic and Proofs (7 Hours)

Logic and Proofs ,Definition of a proposition, Logical operators: Conjunction (AND), Disjunction (OR),Negation (NOT), Compound Statements, Formation of compound propositions, Truth Value, Truth tables for compound statements, Logical Equivalences, De Morgan's Laws, Tautologies and contradictions, Principal of Duality ,Logical Equivalence Examples.

Unit V: Introduction to Graph Theory : (8 Hours)
Introduction to Graphs Definition and types of graphs: simple, multigraphs, pseudographs, Undirected and directed graphs (digraphs) ,Graph Terminologies, GraphTerminologies, Subgraphs, paths, cycles, circuits, Connected and disconnected graphs, Complete graphs, null graphs ,Graph Representation Adjacency matrix, Incidence matrix,Weighted graphs, Eulerian graphs: Euler paths and circuits, conditions, Hamiltonian graphs: Hamiltonian paths and cycles, Applications in network analysis.

Unit VI: Trees and Special Graph Structures (8 Hours)
Introduction to Trees , Definition, Properties of trees. Number of edges in a tree,n−1,Types of Trees,Rooted trees, parent, child, leaf, level, height Center of a tree. Binary Tree: Definition, Examples and properties.Spanning Tree: Definition, Properties, Shortest spanning tree, Kruskal's algorithm.

TEXT BOOKS AND REFERENCES**:**

Text Books:

1. Garg, Reena, Engineering Mathematics, Khanna Book Publishing Company, 2024.
2. Kolman B., Busby R. and Ross S., Discrete Mathematical Structures, 6th Edition, Pearson Education, 2015.
3. Deo Narsingh, Graph Theory with Application to Engineering and Computer Science, Prentice Hall, India, 1979.
4. Vasishtha A. R. and Vasishtha A. K., Matrices, Krishna Prakashan, 2022.

Reference Books:

1. Grimaldi Ralph P. and Ramana B. V., Discrete and Combinatorial Mathematics: An Applied Introduction, Fifth Edition, Pearson Education, 2007.
2. Rosen Kenneth H. and Krithivasan Kamala, Discrete Mathematics and its Applications, McGraw Hill, India, 2019.
3. West Douglas B., Introduction to Graph Theory, Second Edition, Pearson Education, 2015.

3IMC202 JAVA Programming

Course Objectives:

1. Understand the fundamentals of Java programming and object-oriented concepts.
2. Write, compile, and debug Java programs using standard development tools.
3. Develop reusable and maintainable object-oriented Java applications.
4. Apply exception handling and file I/O operations.
5. Implement GUI applications using Java libraries.
6. Use Java APIs and packages to build small-scale applications.

Course Outcomes (CO):

At the end of this course, students will be able to:

CO1: Demonstrate understanding of Java syntax and program structure.
CO2: Apply object-oriented programming concepts like classes, objects, inheritance, and polymorphism in Java.
CO3: Implement exception handling and manage input/output operations using Java.
CO4: Develop simple Java GUI applications using AWT and Swing.
CO5: Use Java Collections and multithreading to build responsive applications.
CO6: Develop, test, and debug Java applications for real-world scenarios.

Unit I: Introduction to Java (6 Hours)
History and features of Java, Java Virtual Machine (JVM) and Bytecode, Java Program Structure, Data types, variables, operators, Control structures (if, switch, loops), Input and output (Scanner class)

Unit II: Classes, Objects, and Methods (6 Hours)
Defining classes and objects, Constructors (Default, Parameterized), Method overloading and static methods, This keyword, Introduction to packages and access modifiers.

Unit III: Inheritance and Polymorphism (7 Hours)
Inheritance and types, Super keyword, Dynamic method dispatch, Abstract classes and interfaces.

Unit IV: Exception Handling and File Handling (7 Hours)
Types of exceptions, Try-catch blocks, Finally block and throws keyword, Custom exceptions, File I/O using FileReader, FileWriter, BufferedReader.

Unit V: GUI Programming with AWT & Swing (6 Hours)
Basics of AWT: components, containers, layout managers, Event handling: Action Listener, Item Listener, Swing components: JButton, JLabel, JTextField. Building basic GUI forms.

Unit VI: Multithreading and Collections (8 Hours)
Threads and Runnable interface, Thread life cycle, Java Collections: List, Set, Map, Iterators and generics

Text Books:
1. Java: The Complete Reference by Herbert Schildt, 12th edition, **Publisher:** McGraw-Hill Education.
2. Programming with Java by E. Balagurusamy, Publisher: McGraw-Hill Education.

Reference Books:
1. Java Programming: A Practical Approach" by C.X. Herbert & C. Muthu, Publisher: McGraw-Hill Education or other local publishers.
2. "Beginning Programming with Java For Dummies" by Barry Burd, Edition: 6th, Publisher: Wiley.

3IMC203 DATA STRUCTURE AND ALGORITHM

Course Objectives:
1. Understand basic concepts of data structures and algorithms.
2. Learn array and string operations with applications.
3. Study stacks and queues with real-life uses.
4. Explore types of linked lists and their operations.
5. Understand trees and graphs with traversal methods.
6. Learn searching, sorting, recursion, and basics of dynamic programming.

Course Outcomes:
At the end of the course, students will be able to:
1. Explain types of data structures and analyze algorithms using Big O.
2. Perform operations on arrays, strings, and apply string matching.
3. Use stacks and queues to solve programming problems.
4. Implement linked lists and compare with arrays.
5. Apply tree and graph algorithms like BFS and DFS.
6. Implement and compare searching and sorting techniques with recursion.

Unit I: Introduction to Data Structures and Algorithms (7hours)
Introduction to Data Structures, Classification of Data Structures: Linear and Non-linear, Introduction to Algorithms Characteristics and Properties of Algorithms, Steps in Algorithm Development, Algorithm Analysis: Time and Space Complexity, Big O Notation.

Unit II: Arrays and Strings (7hours)
Introduction to Arrays, Represntation of Arrays in Memory,Operations on Arrays: Insertion, Deletion, Traversal, Searching, Multi-dimensional Arrays, Introduction to Strings, Basic String Operations.

Unit III: Stack and Queue (8hours)
Concept of Stack, Stack Operations: PUSH, POP, Applications of Stack: Expression Evaluation,Parenthesis Matching Concept of Queue, Types of Queues: Simple Queue, Circular Queue, Deque, Priority QueueQueue Operations and Applications.

Unit IV: Linked list (7hours)
Introduction to Linked List, Types of Linked List: Singly, Doubly, Circular, Operations: Insertion, Deletion, Traversal, Applications of Linked Lists, Comparison with Arrays.

Unit V: Trees and Graphs (7hours)
Introduction to Trees,Binary Tree and Binary Search Tree, Tree Traversals: Inorder, Preorder, Postorder, Introduction to Graphs, Graph Terminology and Representation, Graph Traversals: BFS and DFS

UnitVI: Sorting and Sarching Algorithm (8hours)
Searching Techniques: Linear and Binary Search, Sorting Techniques: Bubble, Selection, Insertion, Merge, Quick Sort, Comparison of Sorting Algorithms, Recursive Algorithms: Factorial, Fibonacci, Tower of Hanoi,Introduction to Dynamic Programming.

Text Books:
1. Fundamentals of Data Structures in C **by** Horowitz, Sahni, and Anderson-Freed
2. Data Structures Through C by Yashavant Kanetkar
3. Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein (simplified reading)

Reference Books:

1. Data Structures and Algorithms Made Easyby Narasimha Karumanchi
2. Data Structures Using C by Reema Thareja
3. Algorithm Design by Jon Kleinberg and Éva Tardos
4. Data Structures and Algorithms in Python by Goodrich, Tamassia**,** and Goldwasser.

3IMC204 SOFT COMPUTING

Course Objectives:

- To introduce the fundamental concepts of soft computing and its difference from conventional computing.
- To provide understanding of fuzzy logic and its applications in control and reasoning.
- To explore neural networks and their architecture for classification and prediction.
- To understand evolutionary algorithms including Genetic Algorithms for optimization.
- To integrate fuzzy logic, neural networks, and genetic algorithms for hybrid intelligent systems.

Course Outcomes:

- Differentiate between soft and hard computing techniques.
- Apply fuzzy logic for reasoning and decision-making under uncertainty.
- Design and train simple neural networks using backpropagation.
- Implement genetic algorithms for solving optimization problems.
- Combine soft computing techniques for solving real-world complex problems.
- Use soft computing tools and libraries in Python for practical applications.

Syllabus Outline (Total 45 Hours):

Unit I: Introduction to Soft Computing (7 Hours)
Definition and scope of Soft Computing, Difference between Hard and Soft Computing, Components of Soft Computing: Fuzzy Logic, Neural Networks.

Unit II: Fuzzy Logic and Fuzzy Systems (8 Hours)
Fuzzy sets and fuzzy operations, Membership functions and linguistic variables, Fuzzy inference systems: Mamdani and Sugeno models, Fuzzy rules and reasoning.

Unit III: Artificial Neural Networks – Basics (7 Hours)
Structure and function of a neuron, Architecture of neural networks: Feedforward, Feedback, Recurrent, Learning rules: Hebbian, Delta, Perceptron.

Unit IV: Neural Networks – Training and Applications (8 Hours)
Backpropagation algorithm, Activation functions, Training, testing and validation of ANN, Applications: Classification and Pattern Recognition.

Unit V: Genetic Algorithms and Optimization (7 Hours)
Basic concepts of genetic algorithms, Genetic operators: Selection, Crossover, Mutation, Fitness function and reproduction.

Unit VI: Hybrid Systems and Recent Trends (8 Hours)
Neuro-Fuzzy systems and Fuzzy-Neural control, Genetic-Fuzzy and Genetic-Neural integration, Real-world hybrid applications

Text Books:
1. "Soft Computing and Intelligent Systems" by K. K. Aggarwal and Yogesh Singh Publisher: New Age International Publishers
2. "Soft Computing – Fundamentals and Applications" by V. K. Jain, Publisher: McGraw Hill Education

Reference Books:
1. "Soft Computing and Intelligent Systems: Theory and Applications" by F. O. Karray and Clarence De Silva, Publisher: Pearson Education
2. "Neural Networks, Fuzzy Logic, and Genetic Algorithms: Synthesis and Applications" By S. Rajasekaran and G. A. Vijayalakshmi Pai, Publisher: PHI Learning.

3IMC211 ENTREPRENEURSHIP DEVELOPMENT

Course Objectives: -
By the end of the course, students will be able to:

1. Understand the basic concepts and importance of entrepreneurship.
2. Identify the qualities of an entrepreneur and their role in society.
3. Learn simple ways to generate business ideas and understand government support.
4. Understand the basic steps to start a small business.
5. Recognize common challenges faced by entrepreneurs and ways to overcome them.

Course Outcomes: -
After completing this course, students will be able to:

1. Define entrepreneurship and describe the role of an entrepreneur.
2. Differentiate between an entrepreneur and a manager.
3. List sources of business ideas and apply simple creative thinking.
4. Explain government support available for small businesses.
5. Describe the basic steps involved in starting a small business.
6. Identifytypes of business ownership and basic funding sources.
7. Discussbasic problems faced by entrepreneurs and suggest simple solutions.

UNIT I: Fundamentals of Entrepreneurship
Meaning and Definition of Entrepreneurship, Characteristics and Qualities of an Entrepreneur, Types of Entrepreneurs (Innovative, Social, Entrepreneur, etc.), Importance of Entrepreneurship in Economic Development, Entrepreneur vs. Manager

UNIT II: Entrepreneurial Ecosystem and Idea Generation
Entrepreneurial Ecosystem: Institutions and Support Systems, Government Initiatives (Start-up India, MSME Schemes, etc.), Sources and Techniques of Idea Generation, Creativity, Innovation, and SWOT Analysis, Simple market research.

UN**IT III: Setting Up and Developing a Business**
Steps to Start a New Venture, Types of Business Ownership (Sole Proprietorship, Partnership, Company, etc.), Introduction to Business Plan and its Components, Sources of Finance (Self-funding, Loans, Investors), Challenges in Entrepreneurship (Risk, Failure, Motivation)

Text Books:
1. “Entrepreneurship Development" – S.S. Khanka
2, “Innovation and Entrepreneurship" – Peter F. Drucker.

Reference Books:
1. “Entrepreneurship" – Rajeev Roy
2. Government reports: MSME, NITI Aayog start-up policies.

3IMC205 MATHEMATIC FOUNDATION TO COMPUTER SCIENCE Lab –I

Course Outcomes:
1. Perform set, relation, and function operations using a programming language.
2. Simulate binomial coefficients and solve recurrence relations in code.
3. Build and analyze truth tables and logical equivalences programmatically.
4. Represent graphs and perform graph traversal and path detection.
5. Apply algorithms like Kruskal’s for minimum spanning trees.
6. Develop tree-based structures like binary and rooted trees through code.

Unit I: Sets, Relations, and Their Applications (Practicals 1–3)
1. **Set Operations and Venn Diagram Simulation**
   Program to perform union, intersection, difference, and complement operations on sets.
2. **Cartesian Product and Relation Representation**
   Create relations from sets and represent them using matrices and digraphs.
3. **Equivalence Relations and Closures**
   Check reflexive, symmetric, and transitive properties; compute closures and partitions.

Unit II: Fundamentals of Functions (Practicals 4–6)
1. **Function Mapping Simulator**
   Implement one-to-one, onto, and bijective functions; visualize mappings.
2. **Polynomial, Exponential, and Logarithmic Functions**
   Plot and evaluate values for various function types over a domain.

   Composite and Inverse Functions
   Implement composition of two functions and compute inverse where applicable.

Unit III: Binomial Expansion and Recurrences (Practicals 7–9)
1. **Binomial Coefficient Calculator**
   Compute coefficients using formula and Pascal’s Triangle.
2. **Binomial Expansion Generator**
   Expand expressions like (a+b)n(a + b)^n(a+b)n and extract general and middle terms.
3. **Fibonacci and Recurrence Solvers**
   Implement Fibonacci sequence, Tower of Hanoi solver, and linear recurrence relations using the characteristic equation method.

Unit IV: Logic and Proofs (Practicals 10–11)
1. **Truth Table Generator**
   Take compound logical expressions and generate corresponding truth tables.
2. **Logical Equivalence Checker**
   Use De Morgan’s laws and equivalences to verify if two propositions are logically equivalent.

Unit V: Introduction to Graph Theory (Practical’s 12–13)
1. **Graph Representation and Traversal**
   Represent a graph using adjacency and incidence matrices; perform BFS/DFS traversal.
2. **Euler and Hamiltonian Path Finder**
   Implement programs to check if Eulerian/Hamiltonian paths or circuits exist in a graph.

Unit VI: Trees and Special Graph Structures (Practical's 14–15)

1. **Binary Tree Implementation**
   Create and traverse a binary tree; display leaf nodes, height, etc.
2. **Spanning Tree using Kruskal's Algorithm**

   Represent a weighted graph and compute the minimum spanning tree using Kruskal's algorithm.

3IMC206 JAVA Programming Lab.

Course Outcomes (CO):
CO1- At the end of this course, students will be able to:
CO2- To develop problem-solving skills using Java.
CO3- To implement object-oriented principles practically.
CO4- To build interactive applications using Java features like GUI, multithreading, and collections.

1. Basic Java Programs
- Write a program to check if a number is even or odd.
- Calculate the factorial of a number using a loop.

2. Classes and Objects
- Create a class Student with attributes and methods to display student details.
- Write a program to calculate the area of a rectangle using a class.

3. Constructors and Method Overloading
- Create a class Employee with parameterized constructors.
- Demonstrate method overloading by creating methods that calculate the area of different shapes.

4. Inheritance and Method Overriding
- Create a class hierarchy with Person -> Employee -> Manager and override a method to display details.

5. Abstract Classes and Interfaces
- Create an abstract class Shape and implement Rectangle and Circle.
- Create an interface Printable and implement it in a class.

6. Exception Handling
- Write a program to handle Arithmetic Exception and Array Index Out Of Bounds Exception.
- Create a user-defined exception for invalid input.

7. File Handling
- Write a Java program to read data from a text file and display it.
- Write a Java program to append text to an existing file.

8. GUI Programming using Swing
- Design a login form using JLabel, JTextField, and JButton.
- Implement event handling for a simple calculator GUI.

9. Multithreading
- Create a Java program to demonstrate thread creation using Thread and Runnable.
- Synchronize two threads accessing a shared resource.

10. Collections Framework
- Write a program using ArrayList to store student names and display them.
- Use HashMap to store and retrieve employee ID and name.

3IMC207 DATA STRUCTURE AND ALGORITHMS- Lab

Course Outcomes:
1. Implement fundamental data structures such as arrays, strings, stacks, queues, and linked lists using C programming.
2. Apply algorithmic techniques like searching, sorting, and expression evaluation to solve basic computational problems.
3. Construct and traverse non-linear data structures like trees and graphs using appropriate algorithms (BFS, DFS).

Tentative list of experiments (ideas) based on the syllabus:
1, Write a program to find time and space complexity of a simple algorithm. (e.g., sum of N numbers)

1. Implement linear search and binary search on an array.
2. Perform basic array operations:
   Insert, delete, traverse, and search elements.
3. Implement matrix operations using 2D arrays.
4. Implement basic string operations:
   Length, copy, compare, reverse, and concatenate.
5. Implement PUSH, POP, and PEEK operations on a stack using arrays.
6. Convert infix expression to postfix and evaluate the postfix expression using stack.
7. Implement a linear queue using arrays.
8. Implement a singly linked list with insert, delete, and display functions.
9. Implement and compare sorting algorithms:
   Bubble, Selection, and Insertion Sort.

3IMC208 CRYPTOGRAPHY - Lab.

Course Objectives:
1. To introduce students to the basic concepts of cryptography through hands-on implementation.
2. To develop an understanding of classical and modern encryption techniques.
3. To provide practical experience with key cryptographic algorithms including symmetric and asymmetric encryption.
4. To familiarize students with hashing, digital signatures, and secure communication practices.
5. To encourage use of Python for simulating cryptographic methods and secure data handling.

Course Outcomes:
1. Implement classical encryption techniques such as Caesar and Vigenère ciphers.
2. Apply modern symmetric algorithms like AES for secure data encryption.
3. Demonstrate key generation and encryption/decryption using RSA.
4. Use cryptographic hash functions for data integrity verification.
5. Create simple secure message exchange systems and understand digital signatures.

List of Practicals:
1. Implement the Caesar Cipher and Vigenère Cipher for encrypting and decrypting text.
2. Simulate the Playfair Cipher and Hill Cipher using matrix operations in Python.
3. Implement monoalphabetic and polyalphabetic substitution techniques.
4. Perform frequency analysis attack on Caesar Cipher to decrypt without key.
5. Write a Python program to encrypt and decrypt data using DES (or simulate simplified DES).
6. Encrypt and decrypt messages using AES with a key and IV in Python (using PyCryptodome).
7. Implement RSA algorithm for key generation, encryption, and decryption of messages.
8. Create a simple digital signature system using hash + RSA and verify its authenticity.

Text Books:
1. Cryptography and Network Security by William Stallings
2. Cryptography and Network Security by Atul Kahate

Reference Book:
1. Applied Cryptography by Bruce Schneier.

3IMC209 SOFT COMPUTING – Lab.

Course Outcomes:
- Implement fuzzy inference systems using Python.
- Build and train neural networks for classification tasks.
- Develop optimization solutions using genetic algorithms.
- Apply hybrid approaches combining fuzzy, neural, and genetic models.
- Utilize open-source soft computing tools in practical use-cases.

List of Practicals:
1. Implement fuzzy set operations and membership functions using scikit-fuzzy.
2. Design a fuzzy inference system (FIS) using Mamdani model for temperature control.
3. Implement a Perceptron learning model for binary classification using NumPy.
4. Train a multi-layer feedforward neural network using Keras on the Iris dataset.
5. Visualize the effect of activation functions on neural network learning.
6. Implement the backpropagation algorithm manually for a small neural network.
7. Use Genetic Algorithm to solve a function optimization problem using DEAP.
8. Simulate crossover and mutation operators with a population of candidate solutions.

****

SEMESTER IV

4IMC201 MATHEMATICAL FOUNDATION TO COMPUTER SCIENCE- II

Course Objectives:

1. To understand basic mathematical functions and their applications in economics.
2. To learn the rules and techniques of differentiation and integration.
3. To study numerical methods for solving algebraic and transcendental equations.
4. To apply numerical techniques for differentiation and integration.
5. To understand the fundamentals of operations research and linear programming.
6. To solve linear programming problems using advanced methods like simplex and duality.

Course Outcomes

Upon completing this course, students will be able to:

1. Apply differentiation to functions like demand, cost, and profit.
2. Solve integrals and basic differential equations.
3. Use numerical methods to find roots of equations.
4. Apply numerical techniques for differentiation and integration.
5. Solve linear programming problems using the graphical method.
6. Apply simplex and advanced methods for optimization problems.

SYLLABUS:

UNIT I – Functions and Differentiation (7 Hours)

Concept of real functions: constant function, linear function, x^n, e^x, a^x,log x. Demand, Supply, Total Revenue, Average Revenue, Total cost, Average cost and Profit function. Equilibrium Point, Break-even point. **Derivative**, Derivatives of Sum, Differences, Product & Quotients, Chain Rule, Derivatives of Composite Functions, Logarithmic Differentiation, First Principle, Derivatives of functions: Constant function, x^n, e^x, a^x,, log x,simple Problem on all rule,

Unit II: Integration (7 Hours)

Definition and Properties, Standard integration, Definite integral,, Substitution and partial fraction, Integration by parts,, First order differential equation, INTEGRATION: Integral as Limit of Sum, Fundamental Theorem of Calculus(without proof.),on simple Problems.

UNIT III – Introduction to Numerical Analysis & Root Finding Techniques (8 Hours)

Introduction to Numerical Analysis, Definition: Study of algorithms that use numerical approximation for solving mathematical problems., Scope: Solving equations, interpolation, differentiation, integration, optimization, etc: Importance in Computer Science and Engineering, Causes of errors and their propagation in iterative methods, Solution of Nonlinear Equations (Root-Finding) ,Transcendental and Polynomial Equations, Problem formulation and mathematical background, Methods for Solving Equations: Bisection Method, False Position Method (Regula Falsi),Secant Method, Newton-Raphson Method.

UNIT IV – Numerical Differentiation and Integration (7 Hours)

Numerical Differentiation, Gregory-Newton's Forward Difference Formula, Gregory-Newton's Backward Difference Formula Use cases in: Velocity and acceleration computations, Gradient estimation in machine learning, Numerical Integration Basic Integration Rules: Midpoint Rule,Trapezoidal Rule,Simpson's 1/3 Rule,Simpson's 3/8 Rule, Composite Rules:Composite Trapezoidal Rule, Composite Simpson's Rule.

Unit V: Introduction to Operations Research & Linear Programming :(8 Hours)

Introduction to Operations Research ,History of Operations Research, Stages of Development of OR,,OR Tools and Techniques, Applications of OR, Limitations of OR Linear Programming – Graphical Method ,Introduction to Linear Programming, Linear Programming Problem Formulation, Formulation with Different Types of Constraints ($\leq$, $\geq$, =),Graphical Analysis of Linear Programming, Graphical Linear Programming Solution Design, Concepts: Design Variables, Constraints, Objective Function, Problem Formulation, Brief Introduction to Calculus Method (for optimization)

UNIT VI: Linear Programming Problem (8 Hours)

Linear Programming Problem Linear Programming Problem, Simplex method, Two-phase method, Big-M method, duality, Integer linear Programming, Dynamic Programming, Sensitivity analysis.

TEXTBOOKS AND REFERENCES:

Text Books:

1. Garg, Reena, Engineering Mathematics, Khanna Book Publishing Company, 2024.
2. Kolman B., Busby R. and Ross S., Discrete Mathematical Structures, 6th Edition, Pearson Education, 2015.
3. Vasishtha A. R. and Vasishtha A. K., Matrices, Krishna Prakashan, 2022.

Reference Books:

1. West Douglas B., Introduction to Graph Theory, Second Edition, Pearson Education, 2015
2. P.K. Gupta & D.S. Hira, "Operations Research", S. Chand & Co. 3. J.K. Sharma, "Operations Research: Theory and Applications", Mac Millan
3. B.S. Grewal, "Elementary Engineering Mathematics", 34th Ed., 1998. 2.
4. Shanti Narayan, "Integral Calculus", S. Chand & Company, 1999.

4IMC202 OPERATING SYSTEM

Course Objectives:

1. Understand the structure and basic functions of operating systems.
2. Learn how operating systems manage processes, memory, and files.
3. Gain knowledge about process synchronization, deadlocks, and CPU scheduling.
4. Analyze various memory management and file system strategies.
5. Understand protection, security mechanisms, and Linux fundamentals.
6. Apply basic Linux commands and shell scripting for system operations.

Course Outcomes (COs):

At the end of this course, students will be able to:

1. Explain the role and architecture of an operating system.
2. Apply CPU scheduling and process management techniques.
3. Demonstrate synchronization techniques and deadlock handling.
4. Evaluate memory allocation techniques such as paging and segmentation.
5. Describe file system structure and I/O management strategies.
6. Use Linux commands and write basic shell scripts.

Unit 1: Introduction to Operating Systems (7 Hours)

Definition and functions of OS, Types: Batch, Time-sharing, Real-time, Distributed, System calls, OS structure: Monolithic, Microkernel, Layered

Unit 2: Process Management (7 Hours)

Process concept and Process Control Block (PCB), Process states, Threads and multithreading, CPU scheduling algorithms: FCFS, SJF, RR, Priority

Unit 3: Process Synchronization & Deadlocks (6 Hours)

Critical section problem, Synchronization tools: Semaphores, Mutex, Deadlock: Causes, Prevention, Avoidance, Detection, Recovery

Unit 4: Memory Management (7 Hours)

Logical vs Physical address space, Contiguous allocation, Paging, Segmentation, Virtual memory: Demand paging, Page replacement (FIFO, LRU)

Unit 5: File and I/O Management (6 Hours)

File system concepts, operations, Directory structures, File allocation methods, Disk scheduling algorithms: FCFS, SSTF, SCAN

Unit 6: Security and Protection (6 Hours)

Security goals and threats, Authentication methods, Access control, Encryption, Firewalls,

Text Books:

1. Silberschatz, Galvin, Gagne, *Operating System Concepts*, Wiley, 10th Edition.
2. Andrew S. Tanenbaum, Herbert Bos, *Modern Operating Systems*, Pearson, 4th Edition.
3. William Stallings, *Operating Systems: Internals and Design Principles*, Pearson, 9th Edition.

Reference Books:

1. D.M. Dhamdhere, *Operating Systems: A Concept-Based Approach*, McGraw Hill.
2. Neil Matthew & Richard Stones, *Beginning Linux Programming*, Wrox Publication.
3. Achyut S. Godbole, *Operating Systems*, McGraw Hill Education.

4IMC203 DATABASE MANAGEMENT SYSTEM

Course Objectives:

1. Understand the fundamental concepts and architecture of database systems.
2. Learn the use of SQL for database creation and manipulation.
3. Explore the principles of relational database design and normalization.
4. Gain insight into transaction management and concurrency control.
5. Understand the internals of query processing and optimization.

Course Outcomes:

At the end of this course, students will:

- CO1: Describe database concepts and architecture.
- CO2: Apply SQL commands to define and manipulate database objects.
- CO3: Design normalized relational schemas for real-world applications.
- CO4: Explain transaction processing and recovery techniques.
- CO5: Analyze query execution and optimization strategies.

Unit I: Introduction to DBMS (7 Hours)
DBMS – Need and Advantages, Database users, Applications, Data Models: Hierarchical, Network, Relational, Object-oriented

Unit II: Architecture and Languages in DBMS (7 Hours)
Three-level architecture and Data Independence, Database languages: DDL, DML, DCL, Data Dictionary and Metadata

Unit III: Relational Model and SQL (6 Hours)
Relational Model – Concepts, Schema, Instances, Keys: Primary, Foreign, Candidate, Super Key, Relational Algebra – Basic Operations, SQL Queries: Selection, Projection, Joins, Sub-queries, Views, Indexes

Unit IV: Database Design (7 Hours)
Entity-Relationship (ER) Model – Entities, Attributes, Relationships, ER to Relational Mapping, Functional Dependencies.

Unit V: Normalization (6 Hours)
Purpose of normalization, Data redundancy and anomalies, Importance in database design, Armstrong's axioms (inference rules), Normalization: 1NF, 2NF, 3NF, BCNF.

Unit VI: Transaction Management and Concurrency Control (6 Hours)
Transactions: Properties (ACID), States, Serializability and Schedules, Concurrency Control: Locks, Time Stamp, Deadlock handling.

Unit 6: Query Processing and Optimization (6 Hours)
Steps in Query Processing, Query Optimization, Query Cost Estimation, Evaluation of Expressions

Text Books:

1. **"Database System Concepts"** – Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw Hill, 7th Edition
2. **"Fundamentals of Database Systems"** Ramez Elmasri, Shamkant B. Navathe, Pearson Education, 7th Edition.

Reference Books:

1. "Database Management Systems" – Raghu Ramakrishnan and Johannes Gehrke, McGraw-Hill.
2. "An Introduction to Database Systems" – C.J. Date, Addison-Wesley.
3. "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence" – Pramod J. Sadalage, Martin Fowler.

4IMC204 ARTIFICIAL INTELLIGENCE

Course Objectives:

1. Introduce foundational concepts and techniques in Artificial Intelligence.
2. Understand problem-solving methodologies using intelligent agents.
3. Explore knowledge representation, reasoning, and learning strategies.
4. Introduce applications in Natural Language Processing (NLP) and ethical considerations in AI.
5. Discuss emerging trends like Generative AI and Large Language Models (LLMs).

Course Outcomes:

1. Define and explain fundamental AI concepts and goals.
2. Model problem-solving tasks using search strategies and agents.
3. Apply basic knowledge representation techniques for reasoning.
4. Understand and apply basic concepts in machine learning and NLP.
5. Analyze real-world applications and implications of AI technologies.
6. Reflect on ethical issues and emerging trends like Generative AI.

Unit I: Foundations of AI and Intelligent Agents (6 Hours)
Definition and History of AI and Applications of AI, Foundations and Branches of AI, Comparison: AI vs. Human Intelligence, Intelligent Agents: Types and Architectures, Agent Environments.

Unit II: Solving Problems with Search (6 Hours)
Problem Formulation, Uninformed Search: Breadth-First Search (BFS), Depth-First Search (DFS), Informed Search: Greedy Search, A* Algorithm, Heuristics and Evaluation Functions.

Unit III: Representing and Reasoning with Knowledge (7 Hours)
Introduction to Knowledge Representation, Propositional Logic: Basics, Syntax, Semantics, Predicate Logic: Forward and Backward Chaining, Semantic Networks, Frames, and Scripts, Simple Rule-Based Expert Systems

Unit IV: Learning from Data: Introduction to Machine Learning (7 Hours)
Overview of Machine Learning and its Types, Supervised and Unsupervised Learning, Introduction to Decision Trees, Model Evaluation, Basics of Neural Networks.

Unit V: Understanding Language in AI Systems (7 Hours)
Introduction to Natural Language Processing (NLP), Text Preprocessing: Tokenization, Stop Words, Stemming, Part-of-Speech (POS) Tagging, Parsing, Text Classification and Sentiment Analysis.

Unit VI: AI in Practice: Ethics, Applications, and the Future (7 Hours)
Real-World Applications of AI: Smart Assistants, Autonomous Systems, AI in Education, Introduction to Generative AI: ChatGPT, DALL·E (Conceptual Overview), Ethical Issues: Bias, Fairness, Transparency, Explainable AI (XAI) and Responsible AI.

Text Book: Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Pearson.

Reference Books:
1. Deepak Khemani, A First Course in Artificial Intelligence, McGraw Hill.
2. Elaine Rich, Kevin Knight, and Shivashankar B. Nair, Artificial Intelligence, McGraw Hill.
3. Shai Shalev-Shwartz and Shai Ben-David, Understanding Machine Learning, Cambridge University Press.

4IMC205 MATHEMATIC FOUNDATION TO COMPUTER SCIENCE Lab –II

Course Outcomes:
1. Apply differentiation to functions like cost, revenue, and profit.
2. Solve integrals and first-order differential equations.
3. Use numerical methods to find roots of equations.
4. Apply numerical differentiation and integration techniques.
5. Solve linear programming problems graphically.
6. Solve complex LP problems using simplex and advanced methods in Python.

List of Practicals:

1. Plot constant, linear, polynomial, exponential, and logarithmic functions using Matplotlib.
2. Implement differentiation rules such as sum, product, and chain rule on business-related functions.
3. Calculate derivatives of xnx^nxn, exe^xex, axa^xax, and log x\log xlogx using SymPy.
4. Determine break-even and equilibrium points using cost and revenue functions.
5. Compute definite and indefinite integrals using SymPy.
6. Solve basic first-order differential equations using the dsolve() function from SymPy.
7. Apply the trapezoidal rule for numerical integration.
8. Use Simpson's 1/3 and 3/8 rules to perform numerical integration.
9. Implement Gregory-Newton Forward and Backward Difference formulas.
10. Estimate velocity and acceleration from data using numerical differentiation techniques.
11. Implement the Bisection Method to find roots of nonlinear equations.
12. Apply Newton-Raphson and Secant methods for root-finding.
13. Graphically solve linear programming problems using Matplotlib.
14. Solve linear programming problems using the Simplex method with scipy.optimize.linprog().
15. Implement integer and dynamic programming problem solutions using Python libraries.

4IMC206 OPERATING SYSTEM –Lab.

Course Outcomes:

1. To understand and simulate various CPU scheduling algorithms to evaluate their performance.
2. To explore and implement memory management techniques, particularly page replacement algorithms.
3. To study and simulate disk scheduling algorithms for efficient disk utilization.
4. To develop and execute basic Linux shell commands and scripting for file and directory operations.
5. To apply shell scripting for file manipulation, permission control, and user/group management.
6. To write and execute shell scripts using programming constructs like variables, loops, and conditionals.

List of Practicals:

1. Study and Simulation of Process Scheduling Algorithms
2. Study of Page Replacement Algorithms
3. Study of Disk Scheduling Algorithm Simulation
4. Write shell scripts for basic commands: ls, pwd, cd, cp, mv, rm, cat, mkdir, rmdir, clear
5. Write shell scripts for file manipulation, permissions, and navigation using commands like touch, chmod, chown, find
6. Write shell scripts for Create, delete, and manage users and groups using useradd, passwd, usermod, groupadd, groups
7. Write shell scripts for Modify and analyze file permissions using symbolic and numeric modes with chmod
8. Write simple shell scripts using variables, conditional statements, and loops

4IMC207 DATABASE MANAGEMENT SYSTEM - Lab.

Course outcomes:
CO1: Apply DDL commands to create and alter relational database structures.
CO2: Execute basic DML operations such as insert, update, delete, and retrieve data using SQL.
CO3: Construct SQL queries using WHERE, LIKE, IN, BETWEEN, ORDER BY, GROUP BY, and aggregate functions.
CO4: Perform data retrieval using various types of joins and subqueries.
CO5: Design an ER model for a real-world system and implement it using SQL.

List of Practical Experiments:
1. Create a database, use DDL commands to create and alter tables
2. Basic DML Operations: Insert, update, delete and retrieve records using SELECT.
3. SQL Queries: Use WHERE, LIKE, IN, BETWEEN, ORDER BY, GROUP BY, and aggregate functions.
4. Joins and Subqueries: Perform inner, outer, self joins, and nested queries.
5. Draw ER model for library management system and implement it in SQL.

4IMC208 ARTIFICIAL INTELLIGENCE - Lab.

Course Outcomes:
1. Simulate intelligent agents and solve problems using classical AI techniques.
2. Implement search strategies and game playing algorithms using Python.
3. Build and apply reasoning systems based on rule-based logic and knowledge bases.
4. Develop and evaluate supervised and unsupervised machine learning models.
5. Apply NLP techniques for language understanding and text classification.

List of Practicals:
1. Simulate a simple intelligent agent (e.g., vacuum cleaner or delivery bot) using condition-based logic.
2. Implement and visualize BFS and DFS on a graph or maze using Python, comparing their efficiency and traversal order.
3. Build a Tic-Tac-Toe game with an AI opponent using the Minimax algorithm.
4. Design a unified rule-based reasoning system that demonstrates both forward and backward chaining for simple inference tasks (e.g., medical or animal classification).
5. Develop a decision tree classifier using scikit-learn, apply it to the Iris or Titanic dataset.
6. Implement K-Means clustering on an unlabeled dataset using scikit-learn, and visualize the clusters to interpret data groupings.
7. Perform text preprocessing (tokenization, stopword removal, stemming) using NLTK or spaCy.
8. Use NLTK to perform POS tagging and syntactic parsing, and visualize the parse tree of complex English sentences.

****