Punyashlok Ahilyadevi Holkar Solapur University, Solapur

Name of the Faculty: Science and Technology

CHOICE BASED CREDIT SYSTEM

Syllabus

Name of the Course : **MCA – I (Sem. I and II)**
(Two Year)

(Syllabus to be implemented from June. 2026)

MASTER OF COMPUTER APPLICATIONS (SCIENCE & TECHNOLOGY FACULTY)

DETAIL SYLLABUS OF SEMESTER I AND II

1. **Introduction:** The Master of Computer Applications (M. C. A.) Programme has been designed with a semester approach in mind. It is a two years course and, in each year, there are two semesters. Courses in semester-I to semester-IV are aimed at skills development in computers using various technologies.

2. **Program Outcomes:**
   - Students are able to take up positions as systems analysts, systems designers, programmers and managers in any field related to information technology.
   - Students are able to apply knowledge of Mathematical Foundations in computing problems.
   - Students pass on their knowledge for planning, designing and building complex Application Software Systems as well as provide support to automated systems or application.
   - Produce entrepreneurs who can develop customized software solutions for small to large Enterprises.
   - Students are able to function as an effective communicator and team member through essential skills in multidisciplinary projects.

3. **Intake Capacity: 60**

4. **Ordinances and regulations**

5. **ELIGIBILIY:** The eligibility criteria for admission to the MCA course will be as decided by the All India Council of Technical Education (AICTE), New Delhi and Directorate of Technical Education (DTE), Government of Maharashtra. It will be published on their respective websites time to time.
   - Passed B.C.A. / Bachelor Degree in Computer Science (B.C.S) / B.Sc. (Entire Computer Science / Computer Science) / Bachelor Degree in Computer Science Engineering or equivalent Degree
     OR passed B.Sc. / B.Com. / B.A. with Mathematics at 10+2 Level or at Graduation Level (with additional bridge Courses as per the norms of the concerned University).
   - Obtained at least 50% marks (45% marks in case of candidates belonging to reserved category) in the qualifying Examination.

6. **FEES STRUCTURE:** The tuition fees or laboratory fees and other fees have to be paid at the beginning of every semester. The Fees for this programme shall be as approved by P. A. H. Solapur University or as per directives of Shikshan Shulka Samiti, Govt. of Maharashtra and competent authority.

7. COURSE STRUCTURE: The MCA course is a FOUR semester course. The teaching for the semesters I and III will be during the first half of the academic year and for the semesters II and IV will be during the second half the academic year.

a) A candidate will be awarded a class or distinction as per the rules of other science subjects.

b) The Regulations / Ordinance not covered in this shall be followed from the Regulations / Ordinance laid down for the science faculty.

A Four Semester M.C.A. Course

Semester	No. of Papers / Practical / Project	Marks	Credits
Semester - I			
• Theory Papers	06	600	24
• Practical Papers	02	100	04
• Project	01	50	02
Semester - II			
• Theory Papers	06	600	24
• Practical Papers	02	100	04
• Project	01	50	02
Semester - III			
• Theory Papers	06	600	24
• Practical Papers	02	100	04
• Project	01	50	02
Semester - IV			
• Project	01	250	10
Total marks and credits		**2500**	**100**

Bridge Course for B.Sc. / B.Com. / B.A. candidates / candidates without Mathematics / Computer background

Note : Students eligible for the Bridge Course must pass the bridge course. However, the credits earned in the Bridge Course shall not be included in the calculation of total credits required for the MCA programme.

Semester - I	**No. of Papers / Practical/Project**	**Marks** (internal assessment only)	**Credits**
• Theory : Programming using C	01	50	02
• Practical : Programming using C	01	50	02

MCA – I Semester I and II : Structure of the Syllabus

M. C. A. Part – I Semester – I						
Paper Code	**Title of the Paper**	**Contact hrs./week**	**Distribution of Marks for Exam.**			**Credits**
			Internal	**University**	**Total**	
	Hard Core – Theory					
HCT 1.1	Object Oriented Programming using C++	04	20	80	100	04
HCT 1.2	Data Structures	04	20	80	100	04
HCT 1.3	Advanced DBMS	04	20	80	100	04
HCT 1.4	Software Engineering	04	20	80	100	04
HCT 1.5	Operating Systems	04	20	80	100	04
Soft Core - Theory (Any One Group)						
SCT 1.1	Discrete Mathematical Structures	04	20	80	100	04
SCT 1.2	Operation Research					
Hard Core – Practical						
HCP 1.1	Practical-I based on HCT 1.1	04	10	40	50	02
HCP 1.2	Practical-II based on HCT1.2 and HCT1.3	04	10	40	50	02
HCP 1.3	Project –I	02	10	40	50	02
	Total	**-**	**150**	**600**	**750**	**30**
M. C. A. Part – I Semester – II						
Paper Code	Title of the Paper	**Contact hrs./week**	**Distribution of Marks for Exam.**			**Credits**
			Internal	**University**	**Total**	
Hard Core – Theory						
HCT 2.1	Java Programming	04	20	80	100	04
HCT 2.2	Python Programming	04	20	80	100	04
HCT 2.3	Web Development (using HTML, CSS & JavaScript)	04	20	80	100	04
HCT 2.4	Computer Communication Network	04	20	80	100	04
Soft Core - Theory (Any One)						
SCT 2.1	Object Oriented Design	04	20	80	100	04
SCT 2.2	Graph Theory					
Open Elective (Any One)						
OET 2.1	Power BI for Data Analytics	04	20	80	100	04
OET 2.2	SWAYAM course*	--	--	--	--	
Hard Core – Practical						
HCP 2.1	Practical-III based on HCT 2.1, HCT 2.2 and HCT 2.3	04	10	40	50	02
HCP 2.2	Project - II	04	10	40	50	02
Open Elective - Practical (Any One)						
OEP 2.1	Practical Based on OET 2.1	02	10	40	50	02
OEP 2.2	Practical / Seminar / Viva based on SWAYAM course OET2.2					
	Total	**-**	**150**	**600**	**750**	**30**

* : The credits will be transferred as per university policy and UGC guidelines after submitting the completion certificate / mark list from the SWAYAM.

Bridge Course for B.Sc. / B.Com. / B.A. students / candidates without Mathematics/Computer background M. C. A. Part – I Semester – I						
Paper Code	**Title of the Paper**	**Contact hrs./week**	**Distribution of Marks for Exam.**			**Credits**
			Internal	**University**	**Total**	
	Hard Core – Theory					
HCT-B1	Theory: Programming using C	02	50	--	50	02
HCP-B1	Practical: Programming using C	02	50	--	50	02

8. Passing Standard: Passing standard is same as that of other M.Sc. courses in the Punyashlok Ahilyadevi Holkar Solapur University. The candidate has to appear for internalevaluation of 20 marks and external evaluation (university exam) for 80 marks for each theory paper. The nature of internal evaluation of practical and project will be decided by the respective schools / departments. The internal evaluation is a process of continuous assessment.

A student who failed in university examination & passed in internal assessment of a paper (subject) shall be given FC (Failed in Term End Exam) Grade. Such student will have to appear for university examination only. A student who fails in Internal assessment and passed in university examination shall be given FR (Failed in Internal Assessment) Grade. Such student will have to appear for university examination as well as internal assessment.

9. Nature of theory question paper

M. C. A._____Sem.____

Paper Name _____

Time : 3 hrs Marks : 80

Instructions :
1. Question No. 1 and 2 are compulsory
2. Attempt any 3 questions from Q. No. 3 to Q. No. 7
3. Figures to the right indicate full marks

Q. 1.	A) Choose correct alternatives (10 questions)	10
	B) Fill in the blanks or true / false (06 questions)	06
Q.2.	Answer the following A) B) C) D)	16
Q.3.	Answer the following A) B)	(10 + 6 **OR** 8 + 8)
Q.4.	Answer the following A) B)	(10 + 6 **OR** 8 + 8)
Q.5.	Answer the following A) B)	(10 + 6 **OR** 8 + 8)
Q.6.	Answer the following A) B)	(10 + 6 **OR** 8 + 8)
Q.7.	Answer the following A) B)	(10 + 6 **OR** 8 + 8)

MASTER OF COMPUTER APPLICATIONS
SEMESTER I
HCT 1.1: Object Oriented Programming using C++

Course Objectives

1. To understand the basic concepts of object-oriented programming (OOP) and how they differ from procedural programming.
2. To learn and apply core OOP principles such as encapsulation, inheritance, polymorphism, and abstraction.
3. To design and implement real-world problems using classes and objects.
4. To understand advanced C++ features like constructors, destructors, operator overloading, and templates.
5. To gain knowledge of dynamic memory management and file handling in C++.
6. To improve logical thinking and problem-solving abilities through practical programming exercises.

Unit – I **[15]**

Introduction to programming: Algorithms and Flowcharts, Steps in problem solving., Variables and data types, Expressions, Constants, Operators, Type conversions

Branching and Looping constructs: If…else statements, Switch-Case construct while, do…while, for loops

Functions : Functions, Passing arguments, Function prototyping, Default argument initializers

Overview Of C++: Object Oriented Programming, Introducing C++ Classes, Concepts of Object Oriented Programming, C++ as a superset of C, New style comments, main function in C++, meaning of empty argument list, User defined data types: enumerated types, use of tag names, anonymous unions, scope of tag names. **Classes & Objects**: Classes, Structure & Classes, Union & Classes, Inline Function, Scope Resolution operator, Static Class Members: Static Data Member, Static Member Function, Passing Objects to Function, Returning Objects, Object Assignment. Friend Function, Friend Classes.

Unit – II **[15]**

Array, Pointers References & The Dynamic Allocation Operators: Arrays, Array initialization, Multi-dimensional arrays, Character arrays, Working with character strings Array of Objects, Pointer, Pointers to Object, Type Checking C++ Pointers, The This Pointer, Pointer to Derived Types, Pointer to Class Members, References: Reference Parameter, call by reference and return by reference Passing References to Objects, Returning Reference, Independent Reference, C++'S Dynamic Allocation Operators, Initializing Allocated Memory, Allocating Array, Allocating Objects.

Constructor & Destructor: Introduction, Constructor, access specifiers for constructors, and instantiation, Parameterized Constructor, Multiple Constructor in A Class, Constructor with Default Argument, Copy Constructor, Destructor.

Unit – III **[15]**

Overloading as polymorphism: Function & Operator Overloading: Function Overloading, Overloading Constructor Function Finding the Address of an Overloaded Function, Operator Overloading: Creating A Member Operator Function, CreatingPrefix & Postfix Forms of the Increment & Decrement Operation, Overloading The Shorthand Operation (i.e. +=,-= etc), Operator Overloading Restrictions, Operator Overloading Using Friend Function, Overloading New & Delete, Overloading Some Special Operators, Overloading [ ], ( ), -, Comma Operator, Overloading << and >>.

Inheritance: Base Class Access Control, Inheritance and Protected Members, Protected Base Class Inheritance, Inheriting Multiple Base Classes, Constructors, Destructors and Inheritance, When Constructor and Destructor Function are Executed, Passing Parameters to Base Class Constructors, Granting Access, Virtual Base Classes.

Unit – IV **[15]**

Virtual Functions & Polymorphism: Virtual Function, Pure Virtual Functions, Early Vs. Late Binding.

Exception handling in C++: try, throw, catch sequence, multiple catch blocks, uncaught exceptions, catch-all exception handler

Templates: Reason for templates compactness and flexibility, function template examples explicit specialization, class templates, out of class definition of member functions.

The C++ I/O System Basics: C++ Streams, The Basic Stream Classes C++

PredefinedStreams, Formatted I/O: Formatting Using The Ios Members, Setting The Formal Flags, Clearing Format Flags, An Overloaded Form of setf(), Using width() precision() and fill(), Using Manipulators to Format I/O, Creating Your own Manipulators.

Reference Books:

1. C++: The Complete Reference: Herbert Schildt, Tata McGraw Hill.
2. Object Oriented Programming with C++: E. Balguruswami, Tata McGraw Hill.
3. Programming with C++ made simple: M. Kumar, Tata McGraw Hill.

Course Outcome

1. Understand the fundamental concepts of Object-Oriented Programming (OOP) and apply them using the C++ programming language.
2. Develop programs using basic C++ features such as data types, operators, control structures, functions, and arrays.
3. Implement core OOP concepts including classes, objects, encapsulation, inheritance, and polymorphism in C++ programs.
4. Apply advanced C++ features such as function overloading, operator overloading, constructors, destructors, and templates.
5. Use file handling and exception handling mechanisms in C++ to develop reliable applications.

HCT 1.2: Data Structures

Course Objectives

1. To understand the fundamental concepts of data structures and their importance in efficient problem solving.
2. To learn different types of data structures such as arrays, linked lists, stacks, queues, trees, and graphs.
3. To analyze and evaluate the performance of algorithms using time and space complexity.
4. To develop the ability to select appropriate data structures for solving specific computational problems.
5. To implement various data structures and their operations using programming languages (such as C/C++).
6. To understand searching and sorting techniques and their applications.
7. To enhance problem-solving skills and logical thinking through practical applications of data structures.

Unit – I **[15]**

Fundamental notions: Primitives and composite data types, choice of data structure and complexity of algorithms, Abstract Data Type.

Arrays: Single and Multidimensional Arrays, sparse matrices.

Stacks: Processing the stacks, Linked list implementation, Application of Stacks for expression solving, Non recursive implementation of recursive algorithms.

Unit – II **[15]**

Queues: Processing the queues, Linked list implementation, Dequeues, Priority queues and their applications.

Linked List: Processing linked list, circularly linked list, doubly linked list, Multilinked lists, String and characters manipulation using arrays and linked list.

Unit – III **[15]**

Trees: Representation of hierarchical relationships, Tree processing, Binary trees, linked list implementation, traversal algorithms, Graph theoretic solutions and tree traversals, Binary trees, Threaded binary trees, Height balanced trees, General Trees.

Design and analysis of algorithm for the implementation: Greedy methods, Dynamic programming, Backtracking, Branch and bound.

Unit – IV **[15]**

Sorting and searching: Various sorts viz. Insertion, Bubble sort, Selection sort, Quick sort, Merge sort, Radix / Bucket sort, Counting sort, searching algorithms and their complexities, Binary tree indexing, B-tree indexing, Hash indexing.

Reference Books:

1. Data structures and algorithms: Alfred Aho, John Hopcraft and Jeffrey Ullman, Addision – Wesley.
2. Introduction to data structures: Bhagat Singh and Thomas Nap, West Publishing Company.
3. The C Programming Language: Brian W. Kernighan, Dennis M. Ritchie, Prentice Hall, 1988.
4. Introduction to Data Structures with applications: J. P. Tremble, Tata McGraw Hill, 1984.

Course Outcome

1. Understand the basic concepts of data structures, algorithms, and their importance in problem solving.
2. Apply appropriate data structures such as arrays, linked lists, stacks, and queues to organize and manage data efficiently.
3. Analyze and implement searching and sorting algorithms for efficient data processing.
4. Design and implement tree and graph data structures to represent hierarchical and network-based data.
5. Evaluate the time and space complexity of algorithms using asymptotic notations.
6. Develop efficient programs using suitable data structures and algorithms to solve real-world problems.

HCT 1.3 : Advanced DBMS

Course Objectives

1. To understand advanced concepts of database management systems beyond basic relational models.
2. To study database design techniques, normalization, and schema refinement for efficient data organization.
3. To learn advanced topics such as transaction management, concurrency control, and recovery techniques.
4. To understand query processing and optimization methods for improving database performance.
5. To explore distributed databases, parallel databases, and NoSQL databases.
6. To gain knowledge of data warehousing and data mining concepts.
7. To develop practical skills in writing advanced SQL queries, stored procedures, and triggers.
8. To understand issues related to database security, integrity, and backup management.

Unit – I **[15]**

Introduction to Database Systems: Database – Definition, Limitations of traditional fileprocessing systems, Advantages of DBMS, Users of DBMS.

Database Architecture and Environment: Components of DBMS, Architecture, Physical, logical and view, DDL, DML, DCL, schemas, life cycle of Database System Development, Functions of DBMS.

Conceptual Database Modelling: Data Model – Concept, types of data models, ER model, concepts of entity, entity set, attributes, domains, existence dependency, Keys: candidate, primary, composite, strong and weak entities, cardinality, specialization, generalization, aggregation, Relational Algebra, Relational Calculus.

Unit – II **[15]**

Relational Database Systems: Characteristics, relation, attribute, tuple, domain, null, Normalization, Functional Dependencies, Multivalued Dependencies, 1NF, 2NF, 3NF, 4NF,5NF Boyce codd's normal form.

SQL and PL/SQL: DDL, DML, DCL, Select: From, Where, Order by, Group by, Having, Intersect, Union, Distinct, Between, In, Between, Different types of functions, Delete, Update, Insert, Nested queries, joins, create, alter and drop, constrains, index, views, Triggers, Grant, Revoke, Commit, RollBack, Savepoint, PL/SQL: %Type, %Rowtype, Exception, Cursor etc.

Unit – III **[15]**

Transaction Management and Concurrency Control: Transaction – properties (ACID),states, Concurrency – control, locks, two phase locking serialization.

Distributed Databases: Standalone v/s Distributed databases, Replication, Fragmentation, Client/Server architecture, types of distributed databases.

Unit – IV **[15]**

Database Recovery: Need for recovery, techniques – log based recovery, check point, differed and immediate updates, shadowing, Catastrophic and non- catastrophic failures, Recovery in multi-database environments, Two phase commit protocol.

Query Processing: Steps in query processing, advantages of optimization. [

Object – Relational Databases: Abstract Datatypes, Nested Tables, Varying Arrays, LargeObjects, Naming Conventions for Objects.

Reference Books:

1. Database System Concepts by Korth: Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw-Hill Higher Education, 2006.
2. Ramez Elmasri, Shamkant Navathe, Pearson Education India, 2011.
3. An Intro. to Database Systems: C. J. Date, Pearson Education India.

Course Outcome

1. Understand advanced concepts of Database Management Systems, database architecture, and database design principles.
2. Apply advanced SQL queries, stored procedures, triggers, and views for efficient database management.
3. Analyze and implement transaction management, concurrency control, and recovery techniques in database systems.
4. Design and manage distributed databases and client–server database architectures.
5. Evaluate and apply database security, authorization, and integrity constraints to protect data.
6. Develop and manage advanced database applications using modern database technologies.

HCT 1.4: Software Engineering

Course Objectives

1. To understand the fundamental principles and concepts of software engineering and its importance in software development.
2. To learn various software development life cycle (SDLC) models such as Waterfall, Agile, Spiral, and Iterative models.
3. To develop the ability to analyze, design, and model software systems using appropriate techniques.
4. To understand requirements engineering, including gathering, analysis, specification, and validation of software requirements.
5. To learn software design principles, including modularity, cohesion, and coupling.
6. To gain knowledge of software testing techniques, quality assurance, and debugging methods.

Unit – I **[15]**

Introduction: Product and Process: Evolving role of software, software characteristic and components, crisis, myths, software engineering – a layered technology, software process, linear sequential model, prototyping model, RAD model, evolutionary software process model.

Software Process and Project Metrics: Measures, metric indicators, metric in process and the project domains, software measurement, metrics for software quality,software quality assurance.

Unit – II **[15]**

Analysis Concepts And Principles: Requirement analysis, communication techniques, analysis principles, software prototyping & Specification.

Analysis Modeling: Elements of the analysis model, data modeling, functional modeling, behavioral modeling, the mechanics of structured analysis, data dictionary, other classical analysis methods.

Unit – III **[15]**

Design Concepts & Principles: Software Design and software Engineering design process, Design principles, Design concepts, Design methods-Data design, Architectural design and process, Transform and Transaction mappings, Design post processing, Architectural design optimization, Interface design, Procedural design.

Unit – IV **[15]**

Software Testing Methods: Fundamentals, Test case design, White box testing, basispath testing, control structure testing, black box testing, Software testing strategies.

Object Oriented Software Engineering: Object oriented concepts, Identifying the elements of an object model, Management of object-oriented software projects, Object-oriented analysis, design and testing.

Reference Books:

1. Software Engineering: Roger S. Pressman, McGraw Hill, 1997.
2. Software Engineering: Shooman, McGraw Hill, 1987.
3. Software Engineering: Ian Sommerville, Addison Wesley, 1985.
4. Object Oriented Analysis and Design: Grady Booch, Pearson.
5. Object Oriented Modeling and Design: James Rambaugh, Michael Bluha, Prentice Hall India, 1991.

Course Outcome

1. Understand the fundamental concepts, principles, and importance of Software Engineering in software development.
2. Analyze different software development life cycle (SDLC) models such as Waterfall, Spiral, Agile, and Prototype models.
3. Apply techniques for software requirement analysis, specification, and system modeling.
4. Design software systems using appropriate software design methodologies and architectural patterns.
5. Implement effective software testing, debugging, and quality assurance techniques.
6. Apply software project management, risk management, and maintenance practices for successful software development.

HCT 1.5: Operating System

Course Objectives

1. To understand the basic concepts and functions of an operating system and its role as an interface between user and hardware.
2. To learn different types of operating systems and their structures.
3. To study process management, including process scheduling, synchronization, and inter-process communication.
4. To understand memory management techniques such as paging, segmentation, and virtual memory.
5. To learn file system management and disk scheduling methods.
6. To gain knowledge of deadlocks, their prevention, avoidance, and recovery techniques.
7. To understand device management and I/O system organization.

UNIT-I **[15]**

Introduction and structure of Operating System: Concept of multi – programming, Parallel, Distributed and real – time – sharing, Operating system structure - Operating system components and Services System Call System.

UNIT-II **[15]**

Process Management, Synchronization and Deadlocks: Process Concept, process scheduling, cooperating process, Threads, inter process communication, CPU scheduling Criteria, scheduling algorithms-FCFS, SJF, Priority, Round Robin, Multilevel feedback queue scheduling.

Process Synchronization and Deadlocks: Critical Section problem, Synchronization hardware, Semaphores, Critical region, Monitors, Deadlock system model, characterization of deadlocks and deadlock Prevention, Avoidance and detection, recovery from deadlock.

UNIT-III **[15]**

Memory Management: Memory Management, Logical and physical Address Space, Swapping, Contiguous Allocation, Paging, Segmentation.

Virtual, memory: demand paging and its performance, page replacement algorithm, allocation of frames, thrashing.

UNIT-IV **[15]**

File System: Secondary Storage Structure File Concept Access method, Directory Structure, Protection and consistency Semantics, File System Structure, Allocation Method, Free space Management, Directory implementation, Disk Structure, Disk Scheduling methods, Disk Management, Swap space Management.

Security and protection: goals of protection domain of protection, access matrix, security program threats, system threats.
CASE Study: Network Operating System, OS Environment, Comparison of Distributed Operating System, Multiprocessor Time Sharing Systems and Network Operating System.

Reference Books:

1. Operating System Principles: Abraham Silberschatz, Peter Galvin, Greg Gagne, 7th ed., Wiley Student Edition.
2. Operating System: Bawn.
3. Modern Operating Systems: Andrew S. Tanenbaum, Prentice Hall India.
4. Operating Systems: Stuart E. Madnick, John J. Donovan, McGraw-Hill, 1974.
5. Operating system: Peterson.

Course Outcome

1. Understand the basic concepts, structure, and functions of an operating system.
2. Explain and apply different process management and CPU scheduling algorithms used in operating systems.
3. Analyze issues related to process synchronization, deadlocks, and resource management.
4. Implement and evaluate various memory management techniques such as paging, segmentation, and virtual memory.
5. Understand and manage file systems and input/output operations in operating systems.

SCT 1.1: Discrete Mathematical Structures

Course Objectives

1. To understand the fundamental concepts of discrete mathematics and their applications in computer science.
2. To develop logical thinking using propositional and predicate logic.
3. To learn techniques of proofs and reasoning, such as induction and contradiction.
4. To study set theory, relations, and functions and their applications.
5. To understand combinatorics, including counting techniques and permutations & combinations.
6. To learn the basics of graph theory and its applications in computing.
7. To understand algebraic structures such as groups and their properties.

Unit – I **[15]**

Combinatorics: Permutations and combinations, Distinct and non-distinct objects, Generating functions for combinations, Enumerators for permutations, Distribution of distinct objects.

Matrices: Basic concepts, Types of matrices, Arithmetic operations on matrices, Scalar Multiplication, Transpose of matrix, Symmetric matrix, Inverse of matrix, Solving simultaneous equation using matrices, Boolean matrices, Eigen values, Eigen vectors, Determinant.

Unit – II **[15]**

Mathematical Logic: Notations, Connectives, Normal forms, Theory of inference for statement calculus, Propositional logic, Predicate calculus, Inference theory of the predicate calculus.

Unit – III **[15]**

Relations and Functions: Elementary set theory, product sets, Relations, Closure properties and related algorithm, Functions, Types of functions, Computer representation of sets, Relations, functions and their manipulations, ordering functions, Recursion.

Unit – IV **[15]**

Graph Theory: Definition, walks, paths, trails, connected graphs, Di-graph representation of relations, regular and bipartite graphs, cycles and circuits, eccentricity of a vertex, radius and diameter of a graph, Central graphs, Hamiltonian and Eulerian graphs, and planar graphs.

Algebraic structures: Groups, Lattices, Applications of the Residue Arithmetic's to computers, Group Codes, Definition & examples of algebraic structures their applications to computer science.

Reference Books:

1. Applied Discrete Structure for Computer Science: Kenneth Levasseur, Alan Doerr, Galgotia Publications, 1986.
2. Discrete Mathematical Structures for Computer Science: B. Kolman and R. C. Busby, Prentice Hall, 1987.
3. Foundations of Discrete Mathematics: K. D. Joshi, Wiley Eastern.
4. Elements of Discrete Mathematics: C. L. Liu, D. P. Mahapatra, Tata McGraw Hill, 1977.
5. Concepts in Discrete Mathematics: S. K. Sahni, Camelot Publishing Co., USA.

Course Outcome

1. Understand the fundamental concepts of discrete mathematics, including sets, relations, and functions.
2. Apply principles of propositional logic and predicate logic to analyze logical statements and solve problems.
3. Analyze mathematical structures such as relations, functions, and algebraic structures used in computer science.
4. Apply techniques of combinatorics and counting principles to solve computational problems.
5. Understand and use concepts of graph theory and trees for modeling and solving real-world problems in computing.
6. Develop mathematical reasoning and problem-solving skills required for computer science applications such as algorithms, data structures, and cryptography.

SCT 1.2: Operations Research

Course Objectives

1. To understand the fundamental concepts and importance of operations research in decision-making and problem-solving.
2. To develop the ability to formulate real-world problems into mathematical models.
3. To learn optimization techniques such as linear programming, integer programming, and nonlinear programming.
4. To understand methods for solving problems like transportation, assignment, and network models.
5. To study inventory control models and their applications in resource management.
6. To learn queuing theory and simulation techniques for analyzing system performance.

Unit – I **[15]**

Introduction of Linear Programming: Various definitions, statements of basic theorems and properties, Advantages, Limitations and Application areas of Linear Programming.

Linear Programming Problems: The Graphical method – Graphical Solution methods of Linear Programming problem, Phase II of the Simplex Method, Primal and Dual Simplex Method, Big –M method, Transportation Problem and its solution, Assignment Problem and its solutions by Hungarian Method.

Unit – II **[15]**

Non-Linear programming: Kuhn-Tucker conditions, Convex functions and convex regions, Convex programming problems, Algorithms for solving convex programming problems.

Unit – III **[15]**

PERT and CPM: Basic differences between PERT and CPM, Arrow Networks, time estimates, Earliest expected time, Latest – allowable occurrences time, Forward Pass Computation, Backward Pass Computation, Representation in Tabular Form, Critical Path, Probability of meeting scheduled date of completion, Calculation on CPM network. Various floats for activities, Critical path updating projects. Operation time cost trade off Curve project, Time cost – trade off Curve, Selection of schedule based on Cost.

Unit – IV **[15]**

Network Flow Problem: Formulation, Max-Flow Min-Cut theorem, Ford and Fulkerson's algorithm. Exponential behavior of Ford and Fulkerson's algorithm.

Matroids: Definition, Graphic and Cographic matroids, Matroid intersection problem.

Reference Books:

1. Linear Programming: G. Hadley, Addison Wesley, 1969.
2. Operations Research an Introduction: H. A. Taha, Macmillan N. Y., 1971.
3. Operations Research: Kanti Swaroop, Gupta and Manmohan, SultanChand and Co., 1985.
4. Operations Research Theory and Applications: J. K. Sharma, 2nd Ed. Macmillan India ltd, 2003.
5. Mathematical Models Operations Research: J. K. Sharma, McGraw Hill,1986.

Course Outcome

1. Understand the fundamental concepts and importance of Operations Research techniques in decision-making and optimization problems.
2. Formulate and solve Linear Programming Problems (LPP) using graphical and simplex methods.
3. Apply different optimization techniques such as transportation and assignment models to minimize cost and maximize efficiency.
4. Analyze network models including shortest path, minimum spanning tree, and project scheduling techniques such as PERT and CPM.
5. Apply queuing theory and inventory control models for effective resource and system management.
6. Use quantitative techniques to solve real-world managerial and computational problems.

Practical and Project		
HCP 1.1	Practical-I based on HCT 1.1	Minimum 15 Practical Assignments should be conducted based on HCT1.1
HCP 1.2	Practical-II based on HCT1.2 and HCT1.3	Minimum 15 Practical Assignments should be conducted based on HCT1.2 and HCT1.3
HCP 1.3	**Project –I**	

Students are required to:

- Form a group of two members only.
- Select/receive the project topic as instructed by the concerned faculty.
- Complete the analysis, design, development, and documentation of the project within the given time.
- Submit the project report and source code before the specified deadline.
- Give a project presentation and demonstration as scheduled.
- Participation of both group members is compulsory

SEMESTER II

HCT 2.1: Java Programming

Course Objectives

1. To understand the fundamentals of the Java programming language and its platform-independent features.
2. To learn and apply object-oriented programming (OOP) concepts such as classes, objects, inheritance, polymorphism, abstraction, and encapsulation.
3. To develop programming skills using Java constructs like data types, control statements, arrays, and methods.
4. To understand advanced concepts such as exception handling, multithreading, and file handling.
5. To gain knowledge of Java packages and standard libraries for efficient application development.

Unit – I **[15]**

Introduction to Java: Importance and features of java, keywords, constants, variablesand data types, Operators and expressions, Decision making, branching and looping: if..else, switch, ?: operator, while, do, for statements, labeled loops, jump statements: break, continue, return.

Classes and Objects: defining a class, adding variables and methods, creating objects,constructors, class inheritance.

Unit – II **[15]**

Arrays and strings: creating an array, one- and two-dimensional arrays, string array and methods, String and String Buffer classes, Wrapper classes.

Inheritance: Basics types, using super, Multilevel hierarchy abstract and final classes, Object class, Packages and interfaces, Access protection, Extending Interfaces, packages.

Exception Handling: Fundamentals, exception types, uncaught exceptions, throw, final, built in exception, creating your own exceptions.

Unit – III **[15]**

Multithreaded Programming: Fundamentals of Java thread model, priorities, synchronization, messaging, thread class, Runnable interface, interthread Communication, suspending, resuming and stopping threads.

Input/Output: Basics, Streams, Byte and Character stream, predefined streams, Reading and writing from console and files. Using Standard Java Packages (lang, util,io, net).

Unit – IV **[15]**

Event Handling: Event Model, Event Classes, Event Listener Interfaces, Adapter

andInner Classes, Working with windows, graphics and text, using AWT controls, Layoutmanagers and menus, handling Image, animation, sound and video, Java Applet.[08] **JDBC:** JDBC API, JDBC Drivers, Products, JDBC Design considerations, Basic stepsto JDBC, setting up a connection to database, Creating and executing SQL statements.

Reference Books:

1. Java-2 the complete Reference: Patrick Naughton and Herbertz Schidt, Tata McGraw-Hill Education, 2002.
2. Programming with Java: Balaguruswamy, Tata McGraw-Hill Edn., 4th edition, 2006.
3. Computing Concepts with Java 2 E Essentials: Horstmann, John Wiley, 2nd edition, 1999.
4. Programming Java: Decker and Hirshfield, Vikas Publication, 2000.

Course Outcome

1. Understand the basic concepts of Java programming, object-oriented principles, and the Java programming environment.
2. Develop Java programs using data types, control structures, arrays, and methods.
3. Implement object-oriented concepts such as classes, objects, inheritance, polymorphism, and encapsulation in Java applications.
4. Use advanced Java features such as exception handling, multithreading, and file handling in program development.
5. Design and develop GUI-based applications using Java frameworks such as AWT or Swing.
6. Build simple Java-based applications for solving real-world computing problems.

HCT-2.2 : Python Programming

Course Objectives

1. To understand the fundamentals of the Python programming language and its syntax.
2. To develop programming skills using data types, control structures, functions, and modules in Python.
3. To learn and apply object-oriented programming concepts in Python.
4. To understand file handling and exception handling techniques.
5. To gain knowledge of Python libraries and packages for various applications.
6. To introduce concepts of data analysis, visualization, and basic scripting using Python.

Unit -I **[15]**

String -Declaring string, String manipulation using string functions, formatting string literals List-Introduction to list, list functions

Tuple- Introduction to tuple, manipulating tuple.

Dictionary- Introduction, accessing values in dictionaries, create, delete and update dictionary items. Function- Types of function, defining function, calling function, advantages of function, function parameters, Anonymous function, Global and local variables, inbuilt functions- map, zip, reduce, filter, any, chr, ord etc. Modules- Importing module, creating and exploring modules, math module, time module, random module, OS, calendar, sys etc. Set-Introduction to set, manipulate set. Package- Introduction, importing from package

File- File opening, closing file, various types of file modes, reading and writing to file manipulating directories Exception handling - try, else, finally, raise keyword.

Regular Expression- various types of regular expression, using match and search function.

Unit –II **[15]**

GUI- Introduction to GUI library, Advantages, Layout management, Events and binding Drawing on canvas (line, oval, rectangle etc) widget such as Frame, Label, Button, Checkbutton, Entry, Listbox, Radiobutton, Text, Spinbox etc.

Database- Introduction, Connections, Executing queries, Transactions, Error Handling OOPs Concept: Introduction to OOP, Classes and objects, Inheritance Method overloading and method overriding, Abstract method and Abstract class, Interfaces in python, Abstract class vs Interfaces, constructor, instance methods, class methods, static methods.

Unit –III **[15]**

Generators- Introduction, communicating with generators with send( )

Decorators –Introduction, simple function decorator, classes as decorators, chained decorators, decorator arguments.

Threads – Introduction, Uses of Threads, creating Thread without using a class, creating a Thread by creating a Sub Class to Thread Class, creating a Thread without creating a Sub Class to Thread Class, Communication between Threads, Thread

communication using notify() and wait() methods

Unit – IV **[15]**

Data science using python

Data Frame- Creating Data Frame from an Excel Spreadsheet, Creating Data Frame from .csv file, Creating Data Frame from python Dictionary, Creating Data Frame from python List of Tuples, Operations on Data Frames.

Data visualization- Bar Graph, Histogram, creating a pie chart, creating line graph

NumPY - Introduction, creating NumPY arrays, indexing and slicing in NumPy.

Pandas - Introduction, installation of panda, data frame, series, range data, slice data, drop a column, concatenation.

References:

- Introduction To Computation and Programming Using Python: With Application to Understanding Data, John V. Guttag
- Think Python, By Allen B. Downey, O'reilly
- Introducing Python: Modern Computing In Simple Packages by Bill Lubanovic
- Python Programming: An Introduction To Computer Science by John Zelle
- Core Python Programming, Dr. R. Nageshwara Rao, Dreamtech
- Introduction to Computer Science using Python, Charles Dierbach, Wiley

E-Resources: -

- Python Book (http://upload.wikimedia.org/wikipedia/commons/9/91/Python_Programmig.pdf)
- http://pythonbooks.revolunet.com/
- Python Threading: http://www.tutorialspoint.com/python/python_multithreading.htm GUI:
- https://wiki.python.org/moin/TkInter
- https://wiki.python.org/jython/LearningJython http://www.tutorialspoint.com/python/python_gui_programming.htm
- Database:
- Python MySQL API https://wiki.python.org/moin/DatabaseInterfaces http://www.tutorialspoint.com/python/python_database_access.htm
- Web Framework: http://webpy.org/docs/0.3/tutorial

Course Outcome

1. Understand the basic concepts of the Python programming language, its syntax, and programming environment.
2. Develop Python programs using data types, control structures, functions, and modules.
3. Apply Python data structures such as lists, tuples, sets, and dictionaries to solve programming problems.
4. Implement file handling, exception handling, and object-oriented programming concepts in Python.
5. Use Python libraries for data processing and problem-solving applications.

HCT 2.3 : Computer Communication Network

Course Objectives

1. To understand the fundamental concepts of computer networks and data communication.
2. To learn different types of network models such as the OSI model and TCP/IP model.
3. To study various network topologies, transmission media, and switching techniques.
4. To understand the functions of different layers in a network and related protocols.
5. To learn error detection and correction techniques and data transmission methods.

Unit – I **[15]**

Introduction:

Uses of Computer networks: Business Applications, Home Applications, Mobile Users, Social Issues;

Network Hardware: Local Area Networks, Metropolitan Networks, Wide Area Networks, Wireless Networks, Home Networks, Internetworks;

Network Software: Protocol Hierarchies, Design Issues for the Layers, Connection-Oriented and Connectionless Service Primitives, Relationship of Services to Protocols;

Example of Networks: The Internet, The ARPANET, NSFNET, Internet usage, Architecture of the internet.

Data Link Layer:

Data Link Layer Design Issues: Services Provided to the Network Layer, Framing, Error Control, Flow Control;

Error Detection and Correction: Error-Correcting Codes, Error-Detecting Codes;

Elementary Data Link Protocols: An Unrestricted Simplex Protocol, A Simplex Stop-and-Wait Protocol, A Simplex Protocol for a Noisy Channel;

Sliding Window Protocols: A One-Bit Sliding Window Protocol, A Protocol Using Go Back N, A Protocol Using Selective Repeat;

Example Data Link Protocols: HDLC—High-Level Data Link Control, The Data Link Layer in the Internet.

Unit – II **[15]**

Network Layer: Network Layer Design issues: Store and Forward packet Switching, Services Provided to the Transport Layer, implementation of Connectionless Service, Implementation of Connection-oriented Services, Comparison of Virtual Circuit and Datagram subnets;

Routing algorithms: The Optimality Principle, Shortest Path Routing, Flooding, Distance Vector Routing, Link state Routing, Hierarchical Routing, Broadcast Routing, Routing for Mobile Hosts;

Congestion Control Algorithms: General Principles of Congestion Control, Congestion Prevention Policies, Congestion Control in Virtual-Circuit Subnets, Congestion Control in Datagram Subnet, Load Shedding, Jitter Control;

Quality of Service: Requirements, Techniques for Achieving Good Quality of Service;

Internetworking: Differences in Networks, Network Connection, Concatenated Virtual Circuits, Connectionless Internetworking; Tunneling; Internetwork Routing, Fragmentation;
The Network Layer in the Internet: The IP Protocol, IP Addresses, Internet Control Protocols,Mobile IP; IPV6.

Unit – III **[15]**

The Transport Layer: The Transport Service: Services Provided to the Upper Layers, Transport Service Primitives, Berkeley Sockets;
Elements of Transport Protocols: Addressing,Connection Establishment, Connection Release Flow Control and Buffering, Multiplexing, Crash Recovery;
The Internet Transport Protocol – UDP: Introduction to UDP, Remote Procedure Call, The Real-Time Transport Protocol;
The Internet Transport Protocols – TCP:Introduction to TCP, The TCP Service Model, The TCP Protocol, The TCP Segment Header,TCP Connection Establishment, TCP Connection Release, Modeling TCP Connection Management TCP Transmission Policy, TCP Congestion Control, Wireless TCP and UDP.

Unit – IV **[15]**

The Application Layer:
DNS – The Domain Name System: The DNS Name Space, ResourceRecords, Name Servers; Electronic Mail: Architecture and Services, The User Agent, MessageFormats, Message Transfer, Final Delivery; The World Wide Web: Architectural Overview, Static Web Documents, Dynamic Web Documents, HTTP, Performance Enhancements, TheWireless Web.

Reference Books:

1. Computer Networks: Andrew S. Tanenbaum, 4th Edition, Pearson Education, Asia,2002.
2. Communication Networks: Fundamental Concepts and Key Architectures, Alberto Leon-Garcia, Indra Widjaja, Tata McGraw Hill, 2006.
3. Data Communications and Networking: Behrouz A. Forouzan, Tata McGraw Hill,Second Edition, 2001.

Course Outcome

1. Understand the fundamental concepts of data communication and computer networks, including network components and transmission media.
2. Explain the functions and working principles of different network models such as OSI and TCP/IP.
3. Analyze various network protocols and data transmission techniques used in computer networks.
4. Understand and apply concepts of switching, routing, and network addressing (IP addressing and subnetting).
5. Evaluate different network technologies such as LAN, WAN, wireless networks, and internet services.

HCT 2.4 : Web Development (Using HTML, CSS & JavaScript)

Course Objectives

1. To understand the fundamentals of web technologies.
2. To design and develop static and interactive websites.
3. To apply HTML, CSS, and JavaScript for modern web development.
4. To build responsive and user-friendly web pages.

Unit I: (15 Hours)

Evolution of Web and Internet, WWW, Internet, Intranet, Extranet, Client–Server Architecture, Web Browsers and Web Servers, HTTP and HTTPS Protocol, URL, URI and DNS, Web Application Architecture

Web Architecture **:** Client–Server architecture, Role of client, server, and database in web systems

Static and Dynamic Websites**:** Static websites, Dynamic websites, Difference between static and dynamic web pages

Unit II (15 Hours)

Introduction to HTML : Introduction to Web Page Designing, Overview of HTML (Hyper Text Markup Language), Features and advantages of HTML, Structure of an HTML document, HTML tags (Singular & Paired tags)

Basic HTML Elements : Headings and paragraphs, Text formatting tags (bold, italic, underline, superscript, subscript), Line breaks and horizontal rules, Comments in HTML

Lists in HTML : Ordered lists, Unordered lists, Definition lists, Nested lists

Hyperlinks **:** Anchor tag, Internal links, External links, Email links, Image links

Tables : Creating tables in HTML, Table tags (table, tr, th, td), Table attributes (border, cellpadding, cellspacing), Rowspan and colspan, Table formatting

Image Tag in HTML : Inserting images in web pages, Image attributes (src, alt, width, height) , Image alignment

HTML Forms : Introduction to forms, Form elements and attributes, Input controls (text, password, radio button, checkbox, submit, reset), Dropdown list and textarea, Labels and fieldsets

HTML5 **:** HTML5 semantic elements (header, footer, section, article, nav), Audio and video elements, Canvas basics

Unit III: (15 Hours)

Introduction to CSS and its advantages, Types of CSS (Inline, Internal, External), CSS syntax and selectors, CSS colors, backgrounds and borders, CSS fonts and text properties, CSS box model, Margins, padding and spacing, CSS positioning (static, relative, absolute, fixed), Display and visibility properties, CSS layout techniques, Flexbox and basic Grid layout, Responsive web design concepts, Media queries

Unit IV: (15 Hours)

Introduction to JavaScript, JavaScript syntax and structure, Variables and data types, Operators and expressions, Conditional statements (if, switch), Looping statements (for, while, do-while), Functions in JavaScript, Arrays and objects, String and number functions, Date and Math objects, Error handling in JavaScript

Reference Books :

1. HTML and CSS: Design and Build Websites – by Jon Duckett
2. Learning Web Design – by Jennifer Niederst Robbins
4. JavaScript and JQuery: Interactive Front-End Web Development – by Jon Duckett
5. Eloquent JavaScript – by Marijn Haverbeke
6. JavaScript: The Definitive Guide – by David Flanagan
7. Web Programming with HTML5, CSS, and JavaScript – by John Dean
8. Beginning HTML, CSS, and JavaScript – by Jon Duckett

Course Outcome

1. Understand the fundamental concepts of web technologies, web browsers, and website architecture.
2. Design static web pages using HTML elements, tags, forms, tables, and multimedia components.
3. Apply CSS styling techniques to create visually appealing and responsive web page layouts.
4. Develop interactive web pages using JavaScript for client-side scripting and event handling.
5. Integrate HTML, CSS, and JavaScript to build dynamic and user-friendly websites.
6. Design and implement simple web-based applications following modern web development practices.

SCT 2.1: Object Oriented Design

Course Objectives

1. To understand the fundamentals of object-oriented analysis and design (OOAD).
2. To learn the importance and role of UML (Unified Modeling Language) in software development.
3. To develop the ability to model software systems using various UML diagrams such as use case, class, sequence, activity, and state diagrams.
4. To understand the process of requirements analysis and system design using object-oriented techniques.
5. To learn how to represent real-world problems through object-oriented models.
6. To gain knowledge of design patterns and their role in software design.
7. To enhance skills in designing efficient, maintainable, and scalable software systems using UML tools.

Unit – I **[15]**

Object Oriented Design and Modeling: Object Oriented Fundamentals, Objects and object classes, object oriented design process, importance of modeling, principlesof modeling, object oriented modeling, Rational Unified Process.

Introduction to UML: Conceptual model of UML, building blocks of UML, Mechanisms in UML, architecture, software development life cycle.

Unit – II **[15]**

Basic Structural Modeling: Classes, relationships, common mechanisms, class and object diagrams.

Advanced structural Modeling: Advanced classes, advanced relationships, Interfaces types and roles, packages, instances and object diagrams.

Unit – III **[15]**

Collaboration Diagrams and Sequence Diagrams: Terms, concepts and depicting a message in collaboration diagrams, Terms and concepts in sequence diagrams, Difference between collaboration and sequence diagram, Depicting synchronous messages with/without priority call back mechanism.

Basic behavioral modeling: Interactions, use cases, Use Case Diagrams, Interaction Diagrams and activity diagrams.

Unit – IV **[15]**

Advanced behavioral modeling: Events and signals, state machines, process and threads, time and space, state chart diagrams.

Architectural Modeling: Terms, Concepts, examples, Modeling techniques for

component diagrams and deployment diagrams.

Reference Books:

1. The Unified Modeling Language User Guide: Grandy Booch, James Rumbough, Ivar Jacobson, Pearson Education 2002.
2. Software Engineering: Ian Sommerville, Sixth Edition, 2003.
3. Fundamentals of Object Oriented Design in UML: Meilir Page Jones, Addison Wesley, 2000.

Course Outcome

1. Understand the principles of Object-Oriented Analysis and Design (OOAD) and the role of Unified Modeling Language (UML) in software development.
2. Analyze system requirements and represent them using UML diagrams such as use case diagrams and activity diagrams.
3. Design system structure using class diagrams, object diagrams, and package diagrams.
4. Model system behavior using sequence diagrams, collaboration diagrams, and state diagrams.
5. Apply object-oriented design principles and UML modeling techniques to develop efficient software system designs.
6. Develop complete software design documentation using UML for real-world applications.

SCT 2.2: Graph Theory

Course Objectives

1. To understand the fundamental concepts of graph theory and its applications in computer science.
2. To learn different types of graphs, their representations, and properties.
3. To study graph traversal algorithms such as Breadth-First Search (BFS) and Depth-First Search (DFS).
4. To understand concepts like connectivity, shortest paths, and spanning trees.
5. To understand applications of graph theory in areas like network design, scheduling, and optimization.
6. To enhance problem-solving and analytical skills using graph-based models.

Unit – I **[15]**

Introducing graphs and algorithmic complexity: Introducing graphs, Introducing algorithmic complexity, Introducing data structures and Depth-first searching, Two linear –time algorithms.

Spanning – trees, branchings and connectivity:

Spanning-trees and branchings, Optimum weight spanning-trees, Optimum branchings, enumeration of spanning-trees, Circuits, cut-sets and connectivity, Fundamental circuits of a graph, Fundamental cut-sets of a graph, Connectivity.

Unit – II **[15]**

Planner graphs: Basic properties of planner graphs, Genus, crossing-number and thickness, Characterisations of planarity, Dual graphs, A planarity testing algorithm.

Networks and flows: Networks and flows, Maximising the flow in a network, Menger's theorem and connectivity, A minimum-cost flow algorithm, Summary and references, Exercises.

Unit – III **[15]**

Matchings: Definitions, Maximum-cardinality matchings, Perfect matchings, Maximum-weight matchings, Summary and references, Exercises.

Eulerain and Hamiltonian tours: Eulerain paths and circuits, eulerian graphs, Finding Eulerian circuits, Postman problems, counting Eulerain circuits, The Chinese postman problem for undirected graphs, The Chinese postman problem for digraphs, Hamiltonian tours, Some elementary existence theorems, Finding all Hamiltonian tours by metrical products, The traveling salesman problem, 2-factors of a graph. [18]

Unit – IV **[15]**

Colouring graphs: Dominating sets, independence and cliques, Colouring graphs, Edge-colouring, Vertex-colouring, Chromatic polynomials, Face- colouring of embedded graphs, The five-colour theorem, The four-colour theorem, Summary and references, Exercises.

Reference Books:

1. Algorithmic graph theory: Alan Gibbons, Cambridge University Press.
2. Graph theory: Harary, Addison Wesley, 1972.

Course Outcome

1. Understand the basic concepts and terminology of graph theory, including graphs, vertices, edges, and types of graphs.
2. Analyze different graph representations and properties used in computer science applications.
3. Apply graph traversal techniques such as Breadth First Search (BFS) and Depth First Search (DFS).
4. Solve problems related to trees, spanning trees, and shortest path algorithms.
5. Apply graph theory concepts in network design, scheduling, and optimization problems.
6. Develop mathematical models using graphs to represent and solve real-world computational problems.

OET 2.1: Power BI for Data Analytics

Course Objectives

1.To introduce students to Business Intelligence and Power BI concepts.
2.To understand how to connect and transform data using Power Query.
3.To learn data modeling concepts and DAX functions in Power BI.
4.To develop interactive reports and dashboards using data visualization tools.
5.To understand advanced Power BI features for data security and performance optimization.

Unit I: Introduction to Power BI [15]

Introduction to Business Intelligence and Data Analytics. Introduction to Power BI. Overview of Power BI components including Desktop, Service, Mobile and Gateway. Power BI architecture overview and installation and setup of Power BI Desktop. Comparison of Power BI with Excel and Tableau. Understanding the Power BI Desktop interface. Connecting data sources such as Excel, CSV files, SQL Server, databases, web, PDF and SharePoint. Understanding fields, tables and different data types in Power BI.

Unit II: Data Transformation using Power Query [15]

Introduction to Power Query Editor and its role in data preparation. Data cleaning techniques including removing rows and columns and filtering data. Changing and managing data types. Handling missing values and errors. Splitting and merging columns for better data organization. Grouping and summarizing data. Merging queries using joins and appending queries using union operations. Creating custom columns using basic M language.

Unit III: Data Modeling and DAX [15]

Introduction to data modeling concepts in Power BI. Creating relationships between tables and understanding relationship types such as one-to-many and many-to-one. Understanding star schema and snowflake schema. Managing relationships using the relationship view. Creating calculated columns and hierarchies such as Year, Quarter and Month. Handling circular references and inactive relationships. Introduction to DAX (Data Analysis Expressions) and understanding calculated columns and measures. Using aggregation functions such as SUM, AVERAGE and COUNT. Applying logical functions such as IF and SWITCH and text functions such as LEFT, RIGHT and CONCATENATE.

Unit IV: Data Visualization and Advanced Features [15]

Introduction to data visualization in Power BI. Creating different types of visualizations such as bar charts, column charts, line charts, area charts, pie charts, donut charts, tables, matrix visuals, cards, gauges, KPIs and map visualizations. Using slicers and filters to create interactive reports. Understanding visual interactions and applying drill down and drill through features. Formatting visuals by modifying colors, labels and legends and customizing tooltips. Using bookmarks and buttons for navigation. Understanding advanced features such as Row Level Security (RLS), using parameters in Power BI, advanced dynamic tooltips, performance optimization techniques and using custom visuals from the Power BI marketplace. Introduction to paginated reports.

Reference Books

1. Microsoft Power BI Cookbook – Brett Powell. Beginning Microsoft Power BI – Dan Clark. Power BI for Dummies – Jack A. Hyman.
2. Pro Power BI Desktop-Free interactive data analysis with Microsoft Power BI by AdamAspin, Apress
3. Introducing Microsoft Power BI by Alberto Ferrari and Marco Russo, Microsoft Press
4. Mastering Microsoft Power BI by Brett Powell, Packt BIRMINGHAM– MUMBAI
5. Microsoft Power BI Complete Reference by Devin Knight, Brian Knight, Mitchell Pearson, Manuel Quintana, Brett Powell, Packt
6. Learn Power BI by Greg Deckler,Packt

Course Outcomes

Students will able to:

1. Understand the basic concepts of Power BI and Business Intelligence.
2. Connect and transform data using Power Query tools.
3. Create relationships and perform data modeling in Power BI.
4. Use DAX functions for data calculations and analysis.
5. Design interactive dashboards and apply advanced Power BI features

OET 2.2: SWAYAM Course

1. Student has to register one course of minimum 4 credits from SWAYAM
2. The selected course should not be from the syllabus.
3. After registration student has to report to the SWAYAM mentor of the dept.
4. Student should register for the online exam of the same course, pass the exam. and submit the marklist / certificate from SWAYAM to the mentor.

<table>
<tr><th colspan="3">Practical and Project</th></tr>
<tr><td>HCP 2.1</td><td>Practical-III based on HCT 2.1and HCT 2.2</td><td>Minimum 15 Practical Assignments should be conducted based on HCT 2.1 and HCT 2.2</td></tr>
<tr><td>HCP 2.2</td><td colspan="2">Project –II</td></tr>
<tr><td colspan="3">Students are required to:
Form a group of two members only.
Select/receive the project topic as instructed by the concerned faculty.
Complete the analysis, design, development, and documentation of the project within the given time.
Submit the project report and source code before the specified deadline.
Give a project presentation and demonstration as scheduled.
Participation of both group members is compulsory</td></tr>
<tr><td>OEP 2.1</td><td>Practical Based on OET2.1</td><td>Minimum 15 Practical Assignments should be conducted based on OET2.1</td></tr>
</table>

Bridge Course for B.Sc. / B.Com. / B.A. students / candidates without Mathematics/Computer background

M. C. A. Part – I Semester – I

HCT-B1	**Theory : Programming using C**

Unit – I [15]

Introduction to problem solving: Algorithms and Flowcharts, pseudo code

Language Fundamentals: History, Character set, C Tokens, Keywords, Identifiers, Variables, Constant, Data Types Operators: Types of operators, Precedence and Associativity, Expression, Statement and types of statements, Structure of 'C' program. Console based I/O and related built-inI/O function: printf( ), scanf( ), getch( ), getchar( ), putchar( )

Unit – II [15]

Control structures: Decision making structures (if, if-else, Nested if –else, Switch), Loop Control structures (while, do-while, for, Nested for loop), break, continue, goto, exit.

Functions: Basic types of function, Declaration and definition, Function call, Parameter passing, Call by value, Call by reference, Scope of variables, Storage classes, Recursion.

Arrays: Definition, declaration and initialization of one-dimensional array, accessing array elements, Displaying array elements, Sorting arrays, Arrays and function, Two-Dimensional array,**Unit – III** [15]

Pointers: Definition and declaration, Initialization, indirection operator, address of operator, pointerarithmetic, dynamic memory allocation, arrays and pointers, function and pointers.

Unit – IV [15]

Strings: Definition, declaration and initialization of strings, standard library functions: strlen( ),strcpy( ), strcat( ), strcmp( ), implementation without using standard library functions.

Structures: Definition and declaration, structure variables initialization, Accessing fields and structure operations, Nested structures,

Reference Books:

1. Let us C: Yashwant Kanetkar, 12th Edn., B P B Publications.

2. Programming in ANSI C: E. Balguruswamy, Tata McGraw Hill

HCP-B1	**Practical: Programming using C**

Minimum 15 practicals based on the above HCT-B1 syllabus.

Punyashlok Ahilyadevi Holkar Solapur University, Solapur

Faculty of Science and Technology

Equivalent Subject for Old Syllabus MCA - I (Semester–I and II)

Paper No.	Name of the Old Paper (w.e.f. 2023-24)	Paper No.	Name of the New Paper (w.e.f. 2026-2027)
Semester - I			
HCT 1.1	Object Oriented Programming using C++	HCT 1.1	Object Oriented Programming using C++
HCT 1.2	Data Structures	HCT 1.2	Data Structures
HCT 1.3	Advanced DBMS	HCT 1.3	Advanced DBMS
HCT 1.4	Software Engineering	HCT 1.4	Software Engineering
HCT 1.5	Operating Systems	HCT 1.5	Operating Systems
SCT 1.1	Discrete Mathematical Structures	SCT 1.1	Discrete Mathematical Structures
SCT 1.2	Operation Research	SCT 1.2	Operation Research
Semester - II			
Paper No.	**Name of the Old Paper (w.e.f. 2023-2024)**		**Name of the New Paper (w.e.f.2022-2023)**
HCT 2.1	Java Programming	HCT 2.1	Java Programming
HCT 2.2	Python Programming	HCT 2.2	Python Programming
HCT 2.3	Computer Communication Network	HCT 2.3	Computer Communication Network
HCT 2.4	System Software	----	**No equivalence**
SCT 2.1	UML	SCT 2.1	UML
SCT 2.2	Graph Theory	SCT 2.2	Graph Theory
OET 2.1	Office Automation	-----	**No equivalence**