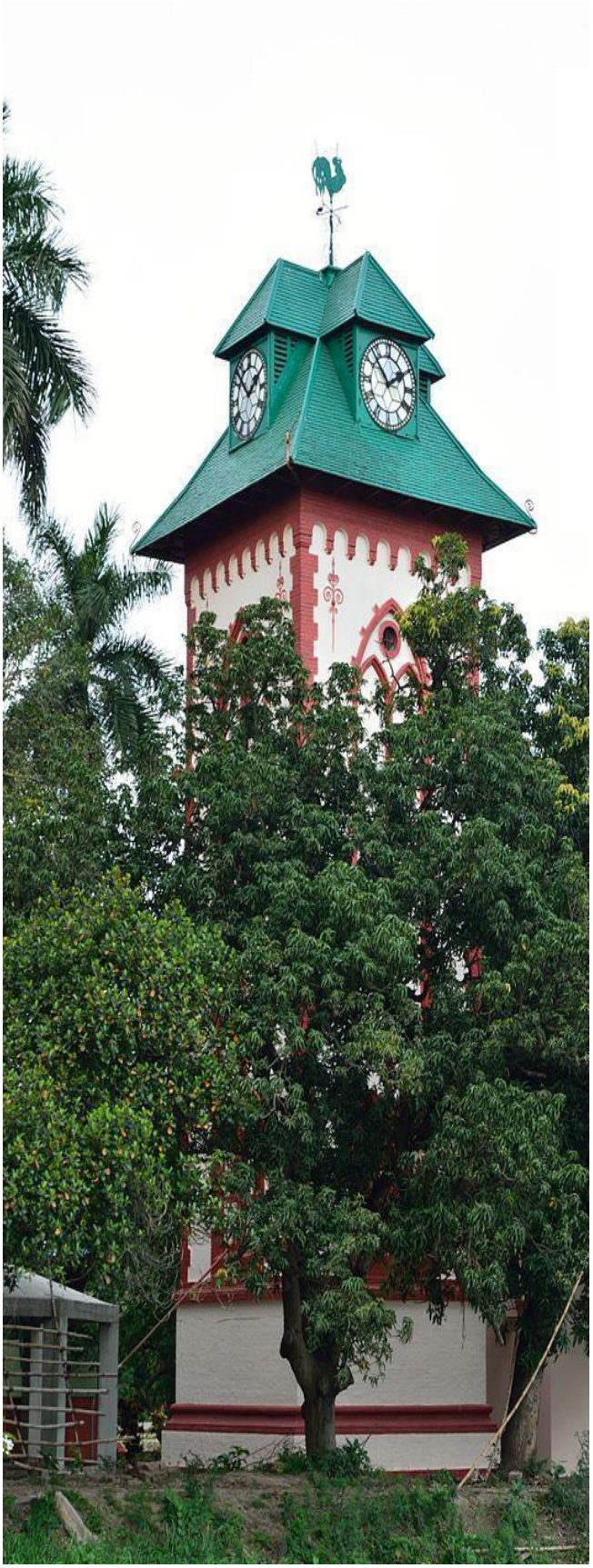

Department of Computer Science and Technology

Postgraduate Programmes

Course Structure and Syllabus

(Effective from 2025-26 admitting batch onwards)

Indian Institute of Engineering Science and Technology (IIEST), Shibpur

Botanic Garden, Howrah

Contents

Course Structure

First Semester

Sl. No.	Type	Course Name	Course code	Class Load/Week			Credit	Class load/ week	Marks
				L	T	P			
1	BSC	Engineering Mathematics – I		3	1	0	4	4	100
2	BSC	Chemistry		3	0	0	3	3	100
3	ESC	Introduction to Computing	CS 1101N	3	0	0	3	3	100
4	ESC	Basic Electronics		3	0	0	3	3	100
5	VAC	Well-being and Happiness		2	0	0	2	2	50
6	HSC	Professional communication in English		2	1	0	3	3	100
		Theory Sub-total		**16**	**2**	**0**	**18**	**18**	**550**
7	ESC	Workshop		0	0	3	2	3	50
8	ESC	Computer Programming Practice Lab	CS 1171N	0	0	3	2	3	50
9	BSC	Chemistry Lab		0	0	3	2	3	50
10		NSS/NCC/PT/Yoga					R*		
		Practical Sub-total		**0**	**0**	**9**	**6**	**9**	**150**
		First Semester Total		**16**	**2**	**9**	**24**	**27**	**700**
		*R: Required (Non-credit but with grade)							

Second Semester

Sl. No	Type	Course Name	Course code	Class Load/Week			Credit	Class load/ Week	Marks
				L	T	P			
1	BSC	Engineering Mathematics – II		3	1	0	4	4	100
2	BSC	Engineering Physics		3	0	0	3	3	100
3	ESC	Basic Electrical Engineering		3	0	0	3	3	100
4	ESC	Introduction to AI and ML	CS 1202N	3	0	0	3	3	100
5	VAC	Energy, Environment and Climate Change		2	0	0	2	2	50
6	PC	Data Structures	CS 1203N	3	0	0	3	3	100
		Theory Sub-total		**17**	**1**	**0**	**18**	**18**	**550**
7	ESC	Engineering Graphics		0	0	3	2	3	50
8	BSC	Physics Lab		0	0	3	2	3	50
9	ESC	Basic Electrical Engineering Lab		0	0	3	2	3	50
10	PC	Data Structure Lab	CS 1273N	0	0	3	2	3	50
11		NSS/NCC/PT/Yoga					R*		
		Practical Sub-total		**0**	**0**	**12**	**8**	**12**	**200**
		Second Semester Total		**17**	**1**	**12**	**26**	**30**	**750**
		*R: Required (Non-credit but with grade)							

Third Semester

Sl. No	Type	Course Name	Course	Class Load			Credit	Class load	Marks
				L	T	P			
1	BSC	Mathematics III		3	0	0	3	3	100
2	ESC	Physics of Materials		3	0	0	3	3	100
3	PC	Object Oriented Programming and Design	CS 2101N	3	0	0	3	3	100
4	PC	Digital Logic	CS 2102N	3	0	0	3	3	100
5	PC	Discrete Structures	CS 2103N	3	0	0	3	3	100
		Theory Sub-total		15	0	0	**15**	15	500
6	P	Mini Project-I	CS 2191N	0	0	3	2	3	50
7	ESC	Physics of Materials Lab		0	0	3	2	3	50
8	PC	Object Oriented Programming and Design Lab	CS 2171N	0	0	3	2	3	50
9	PC	Digital Logic Lab	CS 2172N	0	0	3	2	3	50
		Practical Sub-total		0	0	12	**8**	12	200
		Third Semester Total		**15**	**0**	**12**	**23**	**27**	**700**

Fourth Semester

Sl. No	Type	Course Name	Course Code	Class Load/Week			Credit	Class load/ week	Marks
				L	T	P			
1	PC	Theory of Computation	CS 2201N	3	0	0	3	3	100
2	PC	Computer Architecture & Organization	CS 2202N	3	0	0	3	3	100
3	PC	Database Management System	CS 2203N	3	0	0	3	3	100
4	PC	Design & Analysis of Algorithm	CS 2204N	3	0	0	3	3	100
5	OE	OE1 (From Pool-1)		3	0	0	3	3	100
		Theory Sub-total		15	0	0	**15**	15	500
6	P	Mini Project-II	CS 2291N	0	0	3	2	3	50
7	PC	Computer Architecture & Organization Lab	CS 2272N	0	0	3	2	3	50
8	PC	Database Management System Lab	CS 2273N	0	0	3	2	3	50
9	PC	Design & Analysis of Algorithm Lab	CS 2274N	0	0	3	2	3	50
		Practical Sub-total		0	0	12	**8**	12	200
		Fourth Semester Total		**15**	**0**	**12**	**23**	**27**	**700**

Catalogue of Courses

Catalogue of Engineering Science Courses (ESC)

Sl. No.	Course Code	Course Name
1	CS 1101N/CS 1201N	Introduction to Computing
2	CS 1171N/CS 1271N	Computer Programming Practice Lab
3	CS 1102N/CS 1202N	Introduction to AI and ML

Catalogue of Program Core (PC) Courses

Sl. No.	Course Code	Course Name
1	CS 1203N	Data Structures and Algorithms
2	CS 1273N	Data Structures and Algorithms Lab
3	CS 2101N	Object Oriented Programming and Design
4	CS 2102N	Digital Logic
5	CS 2103N	Discrete Structures
6	CS 2171N	Object Oriented Programming and Design Lab
7	CS 2172N	Digital Logic Lab
8	CS 2201N	Theory of Computation
9	CS 2202N	Computer Architecture & Organization
10	CS 2203N	Database Management Systems
11	CS 2204N	Design & Analysis of Algorithm
12	CS 2272N	Computer Architecture and Organization Lab
13	CS 2273N	Database Management Systems Lab
14	CS 2274N	Design & Analysis of Algorithm Lab

Catalogue of Open Elective(OE) Courses

Sl. No.	Course Code	Course Name	
1	CS 2261N	Data Structures and Algorithms	
2	CS 2262N	Object Oriented Technology	

Syllabi of Courses

FIRST SEMESTER

Course Code	CS1101N/CS1201N	Course Name	Introduction to computing	Course Category	ESC	L	T	P
						3	0	0

Pre-requisite Courses	Nil	Co-requisite Courses	Nil	Progressive Courses	Nil
Course Offering Department		Name of the Department		Data Book / Codes/Standards	Nil

Course Objectives	Upon completing this course, students will be able to: 1. Understand the basics of computer systems, 2. Use a computer for basic tasks, 3. Understand the fundamentals of programming, 4. Develop problem-solving skills.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Number system and Codes:** Positional & non positional number systems, Binary, Octal, Hexadecimal number system and Conversion, Representation of negative numbers & real numbers, Fixed and floating point numbers. Characteristics codes (ASCII, EBCDIC etc.) and others like Grey, Excess-3 etc.	6	Understand the basics of computer systems.
II	**Arithmetic and Logic:** Logic operations & gates, Half adder. & full adder subtraction using add. Repetitive addition and subtraction to accomplish multiplication & division etc.	5	Understand the fundamental logic operations and basic logic gates used in digital circuits.
III	**Computer Organisation:** CPU, Memory and I/O devices – Commonly used peripherals. Role of the CPU, Memory and I/O devices in the context of solving a problem.	6	Understand the structure and functioning of the central processing unit (CPU), memory hierarchy, and commonly used input/output (I/O) devices.
IV	**Problem Solving Steps and Program Development Cycle:** Systematic decomposition, Flowchart, Algorithm, the three constructs (sequential, conditional and iterative). Edit, compilation, Debugging & execution.	3	Learn how to solve problems.
V	**Introduction to Programming in C:** Idea of High level, Assembly level & M/c level language. Interpretation and compilation. Variables and data types (basic), simple programs, assignment, decision, loops, scope: Global & local, control structure (if, if-else, switch, for, while, do while, break and continue) Structural data type (Array, record, file, set etc.), Function, Recursion, Pointers, Introduction to dynamic data structure.	18	Understand how to write and analyze simple C programs using variables, basic data types, control structures (if, if-else, switch, loops), and functions, including recursion.

Course Outcome	CO1: Knowledge of the various parts in a modern computer system. CO2: Knowledge of the representation of numbers and basic operations on numbers in computer CO3: Knowledge of basic logic gates and logic operations. CO4: Ability to write programs in C programming language. CO5: Ability to use basic data structures such as arrays and structures in programs.

Learning Resources	**References and Books:** 1. Brian Kernighan and Dennis Ritchie, "The C Programming Language", 2nd Edition, Prentice Hall India 2. Gottfried Byron S, "Programming with C", Schaum's Outlines Series, Tata Outlines Series, Tata McGraw-Hill

Course Code	CS1171N/CS1271N	Course Name	Computer Programming practice Lab	Course Category	ESC	L	T	P
						0	0	3

Pre-requisite Courses	Nil	Co-requisite Courses	Nil	Progressive Courses	Nil
Course Offering Department		Name of the Department		Data Book / Codes/Standards	Nil

Course Objectives	Upon completing this Lab, students will be able to: 1. Develop practical programming skills, 2. Apply programming concepts, 3. Use programming tools and environments, 4. Analyze and solve problems, 5. Improve coding skills

Module	Syllabus	Duration (class-hour)	Module Outcome
I	Introduction to Linux commands, vi editor and program writing and executing	6	Able to understand the fundamental concepts of Linux platform
II	Assignments on conditional statements	3	Able to understand the conditional statements and their usefulness
III	Assignments on control structure - I	3	Able to understand the conditional statements and their usefulness
IV	Assignments on control structure - II	3	Able to understand the conditional statements and their usefulness
V	Assignments on structural data type (Array, record)	6	Able to understand the concept of Array and its applications
VI	Assignments on function and recursive function	6	Able to understand the concept of function and use of function
VII	Assignments on dynamic data structure	3	Able to understand the concept of dynamic data structure
VIII	Assignments on file handling and file operations	3	Able to learn the file handling concepts

Course Outcome	CO1: Ability to use Linux-based systems, basic Linux shell commands, editors, Linux reference manual, and so on CO2: Ability to select appropriate data structure and algorithm for solving simple problems. CO3: Ability to use the data structures and implement the algorithms, compile and execute programs CO4: Knowledge of techniques to debug various types of errors in programs

Learning Resources	**References and Books:** 1. Brian Kernighan and Dennis Ritchie, "The C Programming Language", 2nd Edition, Prentice Hall India 2. Gottfried Byron S, "Programming with C", Schaum's Outlines Series, Tata Outlines Series, Tata McGraw-Hill 3. Python Crash Course: A Hands-On, Project-Based Introduction to Programming (2nd Edition) 4. Python Programming: An Introduction to Computer Science (4th Edition)

SECOND SEMESTER

Course Code	CS1102N/CS1202N	Course Name	Introduction to AI and ML	Course Category	ESC	L	T	P
						3	0	0

Pre-requisite Courses	Mathematics, programming, and statistics	Co-requisite Courses		Progressive Courses	
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	(i) Students should able to know about the history of AI and basic principles of AI (ii) Student also able to know some basic algorithms of ML (iii) Students should understand the different applications of AI/ML.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Introduction:** Definitions of AI and ML, Brief history and evolution, Difference between AI, ML, and Deep Learning, Types of AI - Narrow vs. General, Types of ML - Supervised, Unsupervised, Reinforcement Learning	5	a) **Define and distinguish** between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning, and explain their interrelationships. b) **Describe the historical evolution** of AI and ML, highlighting key milestones and paradigm shifts in the field. c) **Classify the types of AI** (Narrow AI vs. General AI) and **types of ML** (Supervised, Unsupervised, Reinforcement Learning) with relevant examples.
II	**Understanding Data:** Importance of data in AI/ML, Types of data (structured vs unstructured), Data collection, labelling, and ethics, Basic data visualization	5	a) **Explain the role and significance of data** in the development and performance of AI/ML models. b) **Differentiate between structured and unstructured data** with examples relevant to real-world AI applications. c) **Demonstrate awareness of ethical considerations** in data collection and labeling, and perform basic data visualization using appropriate tools or techniques.

III	**Core Concepts in Machine Learning:** Features and labels, Model training and testing, Overfitting and under fitting (basic intuition), Concept of accuracy, precision, recall (basic)	5	a) **Identify and explain the roles of features and labels** in machine learning datasets and models. b) **Describe the basic process of model training and testing**, and recognize the concepts of overfitting and under fitting using simple examples. c) **Interpret basic evaluation metrics** such as accuracy, precision, and recall to assess model performance.
IV	**Popular Algorithms:** Linear Regression (basic prediction), Decision Trees, K-Means Clustering, Introduction to Neural Networks, Case studies (domain-specific)	7	a) **Explain the basic working principles** of key machine learning algorithms such as Linear Regression, Decision Trees, and K-Means Clustering. b) **Apply selected algorithms to simple prediction and classification tasks**, using domain-specific case studies. c) **Develop a basic understanding of neural networks** and their role in modern AI applications (introductory level).
V	**Ethics, Bias & Impact of AI:** AI fairness and bias, Data privacy concerns, Automation and jobs, Social, ethical, and legal implications	3	a) **Identify and discuss ethical concerns** related to AI, including bias, fairness, and data privacy. b) **Analyze the societal and legal implications** of AI adoption, particularly in the context of automation, employment, and decision-making systems.

VI	**Future of AI & Career Opportunities:** Interdisciplinary research and roles, AI trends- generative AI, conversational AI, robotics, Skills and career paths	3	a) **Identify emerging trends and technologies in AI**, such as generative AI, conversational AI, and robotics, and understand their potential impact across industries. b) **Explore interdisciplinary roles and career paths in AI**, and recognize the key skills required for various AI-related professions.
VII	**Basic Python Programming:** Introduction to Python, Python Syntax & Basic Constructs, Operators, Control Flow, Loops, Data Structures - Lists, Tuples, Dictionaries, Sets, Functions - Definition, Parameters and return values, Built-in functions, Lambda functions, Scope of variables, Strings - String creation and manipulation, String methods, String formatting, File Handling - Opening, reading, writing files, Modes, File pointers and closing files, Exception Handling. Case study of AI/ML problems using Python	14	a) **Write and execute Python programs using fundamental constructs**, including variables, operators, control flow statements, loops, and functions with parameters and return values. b) **Utilize core data structures such as lists, tuples, dictionaries, and sets** to organize and manipulate data effectively in Python programs. c) **Perform file handling operations and implement exception handling** to read from/write to files safely and manage runtime errors gracefully.

Course Outcome	**CO1:** To understand the fundamental concepts of Artificial Intelligence (AI) and Machine Learning (ML), including their capabilities and limitations. **CO2:** To identify real-world problems that can be addressed using AI and ML techniques and formulate problem-solving approaches. **CO3:** To explore different methods for representing knowledge and information for automated reasoning in AI systems. **CO4:** To learn how to design and apply heuristic functions for solving AI problems efficiently. **CO5:** To gain a foundational understanding of data, data types, and basic data analysis techniques for building predictive AI/ML models. **CO6:** Learn how to write and execute Python programs and develop efficient solutions of AI/ML problems using Python.

Learning Resources	**References and Books:** ● Machine Learning - Theory and practice M N Murty and Anantharayana V S ● Alpaydin, Ethem. Introduction to machine learning. MIT press, 2020. ● Artificial Intelligence Rich and Knight ● Explorations in Artificial Intelligence and Machine Learning https://www.routledge.com/rsc/downloads/AI_FreeBook.pdf ● Artificial Intelligence With an Introduction to Machine Learning By Richard E. Neapolitan, Xia Jiang ● Learning Python by Mark Lutz — A comprehensive and detailed book covering Python fundamentals deeply. ● Python Programming: An Introduction to Computer Science" by John Zelle — Introduces computer science concepts through Python, often used in academia.

Course Code	CS1203N	Course Name	Data Structures	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Mathematics, Programming in C	Co-requisite Courses	Nil	Progressive Courses	*Algorithm, OOPD*
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	i) Understand and explain the fundamental concepts of data structures and their role in algorithm development and software design. ii) Analyze time and space complexity of algorithms using Big-O notation to evaluate performance. iii) Implement and apply linear data structures such as arrays, linked lists, stacks, and queues in problem-solving. iv) Understand and use non-linear data structures like trees and graphs, including traversal and searching techniques. v) Select and design appropriate data structures for various computational problems to improve efficiency and scalability.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Abstract Data Type (ADT) and Algorithm:** ADT - concepts of data types, data structure and ADT, properties applicable for ADT. Algorithm – Properties, Concepts of Time and Space complexity	4	Understand the concepts of Abstract Data Types (ADT), their properties, and analyze algorithm efficiency using time and space complexity.
II	**Linked Lists:** Linear Linked List, Circular Linked List, Doubly Linked List, Multi-List, Applications	5	Understand the types, operations, and applications of linear, circular, doubly, and multi-linked lists.
III	**Stacks and Queues:** Concepts and Applications	5	Understand the concepts, operations, and applications of stacks and queues.
IV	**Recursion:** Difference between Recursion and Iteration, Design of Recursive Algorithms	2	Understand the difference between recursion and iteration, and design recursive algorithms.
V	**Trees:** Binary Trees. Properties, Binary Tree Traversals, Expression Trees, Conversion from General Tree to Binary Tree. Binary Search Trees and Operations on BST, Balanced Tree – AVL trees, Red-Black trees, B-Trees.	10	Understand the structure, properties, and traversals of binary trees, expression trees, and conversions from general trees to binary trees. Analyze and implement operations on binary search trees, balanced trees.
VI	**Heap:** Heap data structure and priority Queues	3	Understand the heap data structure and its application in implementing priority queues.
VII	**Graph:** Representations of Graph, Graph Traversal and its Applications	3	Understand graph representations, perform graph traversals, and apply them to solve real-world problems.
VIII	**Sorting:** Insertion Sorts, Exchange Sorts, Selection Sort, Merge Sort, Distribution Sort. Comparisons of Different Sorting Algorithms.	4	Understand and implement various sorting algorithms and compare their performance based on time and space complexity.
IX	**Searching:** Sequential Search, Sequential Search in Ordered List, Binary Search.	4	Understand and implement various searching techniques and compare their efficiency in different scenarios.
X	**Hashing:** Hashing functions, collision resolution techniques.	2	Explain hashing concepts, design hash functions, and implement collision resolution techniques for efficient data retrieval.

Course Outcome	CO1: Ability to select and design data structures and algorithms that are appropriate for a given problem. CO2: Ability to gauge how the choice of data structures and algorithm design methods impacts the performance of programs. CO3: Ability to analyze algorithms and to determine algorithm correctness and complexities. CO4: Skill to identify the scope for improving the performance (in terms of algorithmic betterment) of a given application. CO5: Ability to choose an appropriate algorithm out of different alternative algorithms for a given problem.

Learning Resources	**References and Books:** ● Data Structures using C and C++ by Y. Langsam, M. Augenstein And A. M. Tenenbaum ● Data Structures and Program Design in C by R. Kruse and B. Leung ● Fundamentals of Data Structures in C by E. Horowitz, S. Sahni, S. Anderson-Freed ● Data Structures – A Pseudocode Approach with C++ by R. F. Gillbert and B. A. Forouzan

Course Code	CS1273N	Course Name	Data Structures Lab	Course Category	PC	L	T	P
						0	0	3

Pre-requisite Courses	Programming in C	Co-requisite Courses	Data Structures	Progressive Courses	Nil
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	i) Provide hands-on experience in implementing basic data structures such as arrays, stacks, queues, linked lists, trees, and graphs using C programming language. ii) Strengthen the ability to solve computational problems efficiently by choosing and applying the most appropriate data structures. iii) Encourage analytical thinking by evaluating the time and space complexity of algorithms and understanding trade-offs in different data structure implementations. iv) Apply concepts of Abstract Data Types (ADT) to design modular, reusable, and maintainable code for solving real-world problems.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	Review of Computing Practice: Assignments using recursive and non-recursive functions on Array, etc.	3	Develop and debug recursive and non-recursive functions to solve problems using arrays and fundamental programming constructs.
II	Assignments based on Stack and its Applications	6	Implement stack operations and apply them to solve problems like parenthesis matching, expression conversion, and postfix evaluation.
III	Assignments on linked lists (linear, circular, doubly linked list, etc): Implementation and applications.	6	Implement and manipulate various types of linked lists and apply them to solve real-world data organization problems.
IV	Assignments on queues (circular queue, priority queue): Implementation and applications.	6	Implement circular and priority queues and apply them to model and solve scheduling and resource management problems.
V	Assignments on trees (binary tree, binary search tree, balanced trees): Implementation, creation, operations, applications, etc.	6	Implement and perform operations on various tree structures, applying them to expression evaluation and efficient data storage.
VI	Assignments on search algorithms (sequential, binary and interpolation) on ordered and/or unordered data. Hashing	6	Implement and analyze sequential, binary, and interpolation search techniques on different data sets, and design hashing methods for efficient data retrieval.
VII	Assignments on sorting algorithms (recursive and non-recursive algorithms): bubble sort, insertion sort, selection sort, mergesort, quicksort, etc.	6	Implement and compare various recursive and non-recursive sorting algorithms to understand their efficiency and applications.
VIII	Assignments on graph: Representations, Implementations and Applications	3	Implement graph representations and algorithms, and apply them to solve problems

Course Outcome	CO1: Ability to choose appropriate algorithm for solving a problem by comparing different implementations for the same purpose CO2: Realization of the suitability and importance of appropriate data structures for solving a problem. CO3: Ability to implement algorithms for some well-known computer science problems CO4: Given any unknown problem, ability to design algorithms to solve the problem, and implement the algorithms using appropriate data structures. CO5: Ability to develop small functions / modules, and subsequently achieve bigger tasks using smaller functions that have been developed.

Learning Resources	**References and Books:** ● Data Structures using C and C++ by Y. Langsam, M. Augenstein And A. M. Tenenbaum ● Data Structures and Program Design in C by R. Kruse and B. Leung ● Fundamentals of Data Structures in C by E. Horowitz, S. Sahni, S. Anderson-Freed ● Data Structures – A Pseudocode Approach with C++ by R. F. Gillbert and B. A. Forouzan

THIRD SEMESTER

Course Code	CS2101N	Course Name	Object oriented Programming and Design	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Data Structures	Co-requisite Courses	Nil	Progressive Courses	All subjects related to programming
Course Offering Department		Name of the Department		Data Book / Codes/Standards	Nil

Course Objectives	1. To introduce various programming paradigms and highlight the evolution and rationale for object-oriented programming. 2. To develop a strong understanding of core object-oriented concepts and apply them effectively using C++. 3. To instill good software design principles, UML modeling skills, and promote modular, reusable, and maintainable code development. 4. To familiarize students with common design patterns and test-driven development practices for building robust and scalable software.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Programming Paradigms and Evolution** Introduction to Programming Paradigms: Imperative, Functional, Logic, and Object-Oriented. Strengths and trade-offs of each paradigm. Evolution of programming languages and rationale for OOP.	2	Understand various programming paradigms, their strengths and trade-offs, and the evolution leading to the adoption of object-oriented programming.
II	**Foundations of Imperative and OOP** Concepts in Imperative Programming: Variables, assignment, control structures, state. Problems with procedural code: code reuse, maintainability, and software crisis. Principles of Object-Oriented Programming: Abstraction, Encapsulation, Inheritance, Polymorphism.	4	Grasp the core concepts of imperative programming, recognize its limitations, and understand how object-oriented principles address these challenges.
III	**C++ Basics and OOP Constructs** Overview of C++ syntax relevant to OOP. Classes, objects, constructors, destructors. Encapsulation and data hiding. Member functions and access specifiers. Operator Overloading and Function Overloading. Static members, friend functions, inline functions.	10	Learn the foundational C++ syntax and object-oriented constructs to effectively design and implement encapsulated, modular, and reusable code.
IV	**Inheritance and Polymorphism** Single and multiple inheritance. Constructor and destructor call order. Virtual functions and dynamic (runtime) polymorphism. Abstract classes and interfaces. Virtual base classes and diamond problem.	10	Understand inheritance and polymorphism in C++, including single and multiple inheritance, virtual functions, and mechanisms to resolve complexities like the diamond problem.
V	**Object-Oriented Design and UML** Object-oriented design principles. UML Basics: Class Diagrams, Sequence Diagrams Design Concepts: Coupling and Cohesion, Modularity and Reusability	6	Apply object-oriented design principles using UML to create modular, cohesive, and reusable software systems
VI	**Software Design Principles** SOLID, DRY, KISS	5	Understand and apply key software design principles like SOLID, DRY, and KISS to create efficient, maintainable, and scalable code.

VII	**Design Patterns and TDD** Introduction to Design Patterns: Singleton, Factory, Strategy Introduction to Test-Driven Development (TDD)	5	Learn fundamental design patterns and the principles of Test-Driven Development to build robust, flexible, and well-tested software solutions.

Course Outcome	CO1: Understand and compare various programming paradigms and the evolution that led to the adoption of object-oriented programming. CO2: Apply object-oriented principles such as encapsulation, inheritance, polymorphism, and abstraction using C++ to design and develop modular and maintainable code. CO3: Analyze and implement object-oriented design concepts using UML diagrams and software design principles like SOLID, DRY, and KISS. CO4: Demonstrate the use of design patterns and Test-Driven Development to create robust, reusable, and well-tested software systems.

Learning Resources	**References and Books:** ● The C++ Programming Language by Bjarne Stroustrup ● Object-Oriented Analysis and Design with Applications by Grady Booch ● Thinking in C++ (Volume 1) by Bruce Eckel

Course Code	CS2102N	**Course Name**	Digital Logic	**Course Category**	PC	L	T	P
						3	0	0

Pre-requisite Courses	HS standard Mathematics and Physics	**Co-requisite Courses**	Nil	**Progressive Courses**	Nil
Course Offering Department		CST		**Data Book / Codes/Standards**	Nil

Course Objectives	Students learn the followings: 1. The difference between Analog and Digital systems 2. The number systems, logic gates and Boolean algebra 3. The representation, manipulation and minimization of Boolean functions 4. How to design combinational and sequential circuits 5. The concepts of finite state machines, state minimization, and state machines 6. The analysis and synthesis of asynchronous circuits.

Module	Syllabus	Duration (class-hour)	Module Outcome After completion of each module, the students should have a clear understanding the following
I	**Number Systems and Binary representations:** Number System and Conversion, Signed and Unsigned Binary Number Representation, Binary Addition and Subtraction, BCD and Gray Code Representation, Floating-point Number Representation	3	How numbers are represented in computer and their manipulation using computer, Representation of information using binary code
II	**Boolean Algebra and Logic gates:** Basic theorems and properties of Boolean algebra, Boolean functions, Canonical and Standard forms, Other logic operations (AND, OR, NOR, NAND, NOR, EX-OR and EX-NOR)	3	The properties of switching algebra, Manipulation of Boolean expressions using Boolean algebra, about the different forms of switching functions
III	**Simplification of Boolean Functions:** Map Method, Product of sum simplification, NAND and NOR Implementation, Don't- care Conditions, Tabulation method	3	How to systematically simply the switching functions using Karnaugh maps and Quine McCluskey method
IV	**Combinational Logic:** Introduction, Design procedure, Adders, Subtractors Code conversion, Analysis procedure, Multilevel NAND and NOR circuits, Exclusive-OR and Equivalence Functions Binary Parallel adder, Decimal adder, Magnitude comparator, Decoders and Multiplexers.	9	How to design different combinational logic circuits and how to implement the circuits using Universal logic gates and how to implement the combinational circuits using basic building block such as mux and demux
VI	**Sequential Logic:** Flip-flops, Triggering of flip-flops, Analysis of Clocked sequential circuits, State reduction and assignment, Flip-flop excitation tables, Design procedure, Design with state equations, Registers, Shift-registers, Ripple counters Synchronous counters.	12	How to design sequential logic circuits and also implement them using Flip-flops and logic gates
VII	**Digital Integrated Circuits:** Diode as switch. Use of diodes in AND, OR circuits, Transistor as a switch. RTL, DTL, TTL logic gate circuits. MOS as a switch. Basic MOS inverter. MOS and CMOS logic gates. Fan-in and Fan-out of logic gates, propagation delay, Tristate logic.	8	About different Logic families, How to implement logic gates using transistors, diodes and resistors
VIII	**Testing:** Testing of Digital Circuits, Fault Modeling, Test Generation Pattern, Design for Testability and Built-in Self-Test	4	Familiarization of fault model, Test pattern generation and build-in-self-test

Course Outcome	After completion of this course, the students should do the following: 1. Design and implementation of combinational circuits such as adder, subtractor, code converter, comparator etc. 2. Design and implementation of sequential circuits such as Flip-flop, register, counter etc. 3. Implementation of universal logic gates using transistors, diodes and resistors

Learning Resources	1. Digital Logic and Computer Design - M Morris Mano, Pearson India Education Pvt. Ltd(Old Edition) 2. Fundamentals of Logic Design - Charles H. Roth, Fourth Edition, Jaico Publishing House 3. Switching and Finite Automata Theory- Kohavi and Jha, Third Edition, Cambridge University Press

Course Code	CS2103N	Course Name	Discrete Structures	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Mathematics - I & II	Co-requisite Courses	Nil	Progressive Courses	Design & Analysis of Algorithms, Theory of Computation, Graph Algorithms, Database Management Systems.
Course Offering Department		Name of the Department		Data Book / Codes/Standards	Nil

Course Objectives	1. To get familiar and understand the fundamental notions in discrete mathematics. 2. To describe binary relations between two sets; determine if a binary relation is reflexive, symmetric, or transitive or is an equivalence relation; combine relations using set operations and composition. 3. To understand and demonstrate the basic concept of an algorithm and its application in combinatorial mathematics. 4. To identify the base step and the recursive or inductive step in applied problems and give a recursive and a non-recursive definition for an iterative algorithm 5. To identify the basic properties of graphs and trees and model simple applications.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Logic:** Propositional Logic: Syntax, Semantics, Validity and Satisfiability, Basic Connectives and Truth Tables, Logical Equivalence: The Laws of Logic, Logical Implication, Rules of Inference, Use of Quantifiers.	8	-Use rules of logic to understand the meaning of mathematical statements and reason with them. -Construct valid mathematical arguments from propositions
II	**Proof Methods:** Proof Methods and Strategies, Forward Proof, Proof by Contradiction, Proof by Contraposition, Proof of Necessity and Sufficiency, The Well-Ordering Principle, Proof by Induction, Strong Induction.	8	-Undestand the use case for different proof techniques -Apply these techniques to solve both mathematical and computer science problems
III	**Sets, Relations, and Functions:** Operations and Laws of Sets, Cartesian Products, Binary Relation, Closures of Relations, Partial Ordering Relation, Equivalence Relation, Image of a Set, Sum and Product of Functions, Bijective Functions, Inverse and Composite Function, Size of a Set, Finite and Infinite Sets, Countable and Uncountable Sets, Cantor's Diagonal Argument, Power Set theorem..	10	-Familiarization with different discrete structures and understanding their uses in various fields of computer science - like data structures, algorithms, DBMS, etc.
IV	**Combinatorics:** The basics of counting, Pigeonhole Principle, Permutations & Combinations, Advanced Counting Techniques - solving linear recurrence relations and their application in analyzing algorithms, generating functions and their use to solve combinatorial problems.	8	-Apply basic counting techniques such as the sum and product rules, and use the Pigeonhole Principle to solve fundamental combinatorial problems. -Analyze and solve linear recurrence relations, with applications in recursive and divide-and-conquer algorithms -Construct and manipulate generating functions to represent sequences, and use them to solve combinatorial problems.
V	**Graphs and Trees:** Graphs and their properties, Degree, Connectivity, Path, Cycle, Graph Representation, Subgraph, Isomorphism, Graph coloring, Matching, Eulerian and Hamilton circuits; Trees: Introduction to Trees, Application of Trees, Spanning Tree.	8	-Understand modeling and solving of real-life problems in form of graphs and trees

Course Outcome	CO1: Distinguish between the notion of discrete and continuous mathematical structures CO2: Use proper proof technique to prove mathematical statements/theorems CO3: Apply induction and other proof techniques towards problem solving CO4: Solve problems in Computer Science using graphs and trees

Learning Resources	1. Kenneth Rosen, Discrete Mathematics and Its Applications, 7th Edition, McGraw Hill Publishing Co., 2017. 2. Susanna S. Epp., Discrete Mathematics with Applications, 4th edition, Brooks/Cole Pub Co, 2010. 3. C. L. Liu and D. P. Mohapatra, Elements of Discrete Mathematics: A Computer Oriented Approach, 3rd Edition, Tata McGraw-Hill, 2017.

Course Code	CS2171N	Course Name	Object Oriented Programming and Design Lab	Course Category	PC	L	T	P
						0	0	3

Pre-requisite Courses	Data Structures	Co-requisite Courses	Object oriented Programming and Design	Progressive Courses	All subjects related to programming
Course Offering Department		Name of the Department		Data Book / Codes/Standards	Nil

Course Objectives	1. To develop proficiency in C++ programming through practical implementation of object-oriented concepts such as classes, objects, inheritance, and polymorphism. 2. To apply software design principles like encapsulation, modularity, and reusability in solving real-world problems C++ constructs. 3. To enhance problem-solving and debugging skills by integrating design patterns, templates, exception handling, and basic test-driven development practices.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	Basics of C++ and Programming Paradigms: Introduction to C++ syntax. Input/output operations. Control structures (if-else, loops)	6	Write basic C++ programs using control structures and recognize the foundational differences between imperative and object-oriented approaches.
II	Functions, Classes, and Objects: Functions: definition. call by value/reference. Defining classes and objects. Constructors and destructors	6	Create classes with constructors/destructors and modularize code using functions and objects.
III	Encapsulation and Overloading: Access specifiers: public, private, protected. Getter/setter methods. Operator overloading (arithmetic, stream). Function overloading	6	Implement encapsulation and enhance class usability through operator and function overloading.
IV	Inheritance: Single and multilevel inheritance. Constructor and destructor order. Accessing base class members	6	Demonstrate code reuse and relationship modeling using inheritance and understand object construction flow.
V	Polymorphism and Virtual Functions: Function overriding. Base class pointers and dynamic binding. Virtual functions and abstract classes	6	Apply runtime polymorphism using virtual functions and design abstract interfaces.
VI	Templates and Exception Handling: Function and class templates. Try, catch, throw for exception handling. User-defined exceptions	6	Write generic programs using templates and implement robust error handling using exceptions.
VII	Design Patterns / TDD: Use basic design patterns (e.g., Singleton or Factory). Introduce test-driven development (write test cases before implementation)	3	Integrate object-oriented programming, design patterns, and basic testing to build a functional and well-structured application.
VII	Development of a use-case as a project following the principles of Object Oriented Design	3	Designed and developed a complete use-case-driven project applying Object-Oriented Design principles.

Course Outcome	CO1: Implement fundamental object-oriented programming concepts such as classes, objects, constructors, and encapsulation using C++. CO2: Apply inheritance, polymorphism, and operator overloading to design reusable and extendable C++ programs. CO3: Demonstrate the use of templates, exception handling, and UML diagrams to model and develop robust and generic software solutions. CO4: Design and develop a project using object-oriented principles, basic design patterns, and test-driven development techniques.

Learning Resources	**References and Books:** ● The C++ Programming Language by Bjarne Stroustrup ● Object-Oriented Analysis and Design with Applications by Grady Booch ● Thinking in C++ (Volume 1) by Bruce Eckel

Course Code	CS2172N	**Course Name**	Digital Logic Lab	**Course Category**	PC	L	T	P
						0	0	3

Pre-requisite Courses	Nil	**Co-requisite Courses**	Nil	**Progressive Courses**	Nil
Course Offering Department		CST		**Data Book / Codes/Standards**	Nil

Course Objectives	Students learn the followings: 1. Design and implementation of Logic gates (different logic families) using diodes, transistors, and resistors 2. Design and implementation of combinational circuits (Adder, subtractor, parity generator/checker) using universal logic gates. 3. How to implement Boolean function using multiplexer and decoder/demultiplexer. 4. Design and implementation of sequential circuits (latches, flip-flops, counters) using universal logic gates.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	Logic family: Implementation of OR and AND gates using diodes. Study on characteristics of DTL and TTL inverters using discrete components, Study on characteristics of TTL and CMOS gates	14	Students should able to implement different Logic gates using diodes, transistors, and resistors
II	Combinational logic circuits: Design and implementation of combinational circuits such as, Adders, comparators, parity generator and checker.Implementation of Boolean functions using multiplexer and decoder/demultiplexer.	14	Students should be able to design and implement combinational circuits such as adder, subtractor, parity generator/checker using universal logic gates. They should also implement Switching Function using multiplexer and decoder/demultiplexer.
III	Sequential circuits: Study of latch and flip-flop, design of counters.	14	Students should be able to implement latches and flip-flops using logic gates.

Course Outcome	After completion of this course, the students should do the following: Design and implementation of combinational circuits such as adder, subtractor, comparator etc. using universal logic gates. They should also implement Switching Function using multiplexer and decoder/demultiplexer. Design and implementation of sequential circuits such as Latches, Flip-flop using logic gates. Implementation of universal logic gates using transistors, diodes and resistors

Learning Resources	Laboratory Manual

FOURTH SEMESTER

Course Code	CS2201N	Course Name	Theory of computation	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Discrete Structures	Co-requisite Courses	Nil	Progressive Courses	Compiler Design, Natural Language Processing
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	1. To give an overview of the theoretical foundations of computer science and limitations of different computational models 2. To illustrate finite state machines to solve problems in computing 3. To explain the hierarchy of problems arising in the computer sciences. 4. Ultimately enabling students to analyze algorithms

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Introduction:** Computations, Different models of computation, Language recognizer and generator	2	To learn about different model of computation
II	**Regular Languages:** Finite Automata – Deterministic and non-deterministic, Regular expression, regular grammar, Equivalence of regular languages, Pumping lemma, Myhill-Nerode Theorem, Minimization of FSM, Properties of the class of Regular languages, Decision algorithm for regular sets.	12	To learn about finite automata (DFA, NFA), their properties, and their relationship to regular languages and also the limitations of this model
III	**Context Free Language:** Context free grammers (CFG) and languages (CFL), Parse trees, Ambiguous, unambiguous and inherently ambiguous grammars, Normal Forms (Chomsky and Greibach), simplification of CFG, Pushdown automata (deterministic and non-deterministic), Acceptance of language by empty stack, final state and their equivalence, Properties of the class of CFLs, Proving a language to be CFL or not, Pumping lemma for CFG, Decision algorithm for CFG	12	Comprehend the definition of CFG, its relationship to languages and automata, and the Chomsky hierarchy, constructing CFGs for given languages, Recognize that not all languages are context-free and understand the limitations of CFGs
IV	**Recursive and Recursively enumerable Language:** Unrestricted grammar, Computable function, Turing Machines (deterministic and non-deterministic), Equivalence of deterministic and non-deterministic TM, Extensions og TM and their simulations, Universal TM, Halting problem of TM, Decidability, Non-computability, Complexity classes, notion of reductions	10	Knowing about Turing machine model, Understanding limits of computation, including decidability and undesirability. Understand the concepts of complexity theory, including P, NP, and NP-complete problems

Course Outcome	Upon successful completion of this course, students will be able to 1. Give the mathematical definition of various computational models and state and prove their limitations. 2. explain the notion of nondeterminism, reductions and the idea of NP-Completeness.

Learning Resources	**Referred Books:** 1. J. E. Hopcroft, R. Motwani, & J. D. Ullman Introduction to Automata Theory, Languages, and Computation, Third Edition, Pearson, 2008. 2. Denial I. A. Cohen Introduction to Computer Theory, Second Edition, John Wiley & Sons, 1997. 3. "Introduction to Languages And The Theory Of Computation" John C Martin

Course Code	CS2202N	Course Name	Computer Architecture and Organization	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Digital Logic	Co-requisite Courses		Progressive Courses	Nil
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	To make students familiar with the ● Principles and implementation of computer arithmetic. ● Operations of CPU, ALU, instruction cycle and buses. ● Fundamentals of different instruction set architectures and implementation in CPU. ● Memory system/storage and I/O organization. ● Principles of high speed computation and multiprocessor systems.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Introduction:** History of computing, von Neumann machine, Instruction and data, fixed-point and floating-point numbers, errors, IEEE standards.	4	Witnessing the history of computing, generations of computer systems, and learning the basics of number systems and representations of numbers.
II	**Processor design:** Instruction Set Architecture - Instruction format, opcode optimization; operand addressing; Instruction implementation - data movement, branch control, logical, Input/output and debugging instructions; arithmetic instruction implementation – addition and subtraction, multiplication-division, 2's complement multiplication; Booth's algorithm – theory and examples; bit-pair algorithm; high performance arithmetic.	8	Learning of designing an instruction set for a CPU/processor and their (arithmetic/logical/data movement/program control/etc instructions) implementation in hardware.
III	**Control unit design:** Hardwired control, micro-programmed control design – micro-instruction formats, control optimization.	6	Learning the design of control unit to explore the detailed functioning of the CPU/processor.
IV	**Memory subsystem:** Memory technology, memory interfacing, memory interleaving, Memory hierarchy – introduction to virtual memory system; cache – performance, address mapping, coherence; content addressable memory (CAM).	10	Knowledge about memory cell and memory module design, characterization of different types of memories, the interfacing techniques of memory modules with CPU/processor, high speed memory system design.
V	**Peripherals:** Basic properties, bus architectures, interfacing of I/O devices, data transfer schemes – programmed I/O, DMA, mass storage, RAID.	6	Characterization of I/O devices, exploring I/O interface and data transfer mechanisms between CPU and I/O, knowledge about the mass storage attached to the computing system.
VI	**Pipelining:** Pipelining data path and instructions, speed up, CPI, latency; linear/non-linear pipeline – reservation table, MAL; hazards, super-pipelined and super- scalar processors; multiprocessor system architecture.	8	Knowledge about the parallelization schemes in processor execution to speed up, computing models and bottlenecks.

Course Outcome	Upon successful completion of this course, students will be able to ● Follow the architecture of modern computers, and how the m/c performs arithmetic/logical operations. ● Implement different arithmetic/logical/register transfer instructions. ● Understand the detailed functioning of the CPU. ● Exemplify the memory organization and its communication with CPU. ● Exemplify I/O devices and its communication with CPU. ● Understand the properties of mass storage attached to the computing unit. ● Learn principles of modern high speed computation, computing models and bottlenecks.

Learning Resources	1. Computer Architecture and Organization, John P. Hayes, McGraw Hill. 2. Computer System Architecture, M. Morris Mano, Pearson. 3. Advanced Computer Architecture: Parallelism, Scalability, Programmability, Kai Hwang, McGraw Hill.

Course Code	CS2203N	Course Name	Database Management System	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Data Structures	Co-requisite Courses	Nil	Progressive Courses	Software Engineering
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	• The objectives of the course are to • Equip students with strong foundation in database and database management systems concepts with an emphasis on how to organize, maintain and retrieve data efficiently from a database • Focus on database design theory using ER modelling, Normalization and lossless decomposition • Cover Transaction processing and Recovery techniques

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Database and Database Management System:** Traditional File systems versus Database systems, database users, Role of Database Administrators (DBA), Concepts of the 3-level architecture, Features of DBMS.	3	Ability to understand the overall domain of database management systems (DBMS) with the emphasis on the difference between database and DBMS.
II	**Entity Relationship Model:** High-level conceptual modelling, ER Modelling concepts, Cardinality constraints, Weak-entity types, Subclasses and inheritance, Specialization and Generalization, Conversion of ER diagram into relations and Database Design Guidelines	4	Ability to apply design guidelines for designing relational schemas towards improving data integrity, reducing redundancy and anomalies.
III	**Relational Model, Languages and Systems:** Relational algebra, Relational Calculus, Relational model concepts, ER to relational mapping, Data definition, manipulation and related queries in SQL, Views, Integrity constraints.	6	Prepare students with the knowledge to design, implement, and maintain relational databases efficiently.
IV	**Formal Database Design:** Concept of functional dependencies, Normal forms based on Functional Dependency (FD), Multi-valued Dependency and Join Dependency, Lossless decomposition, Dependency Preservation, Canonical cover of a set of FD.	8	Equip students to transform application requirements into a logical and physical database model.
V	**Indexing Structures:** Basic terminologies, Different types of indexes, B-trees, B+ trees.	3	Gather knowledge about how to store and retrieve data efficiently towards improving searching/update performance
VI	**Transaction Processing and Concurrency Control:** Concurrency issues, need for transactions, Transactions properties, Transaction states, Serializability, Locking, Deadlocks and starvation, Lock-based protocols, Timestamp-ordering based protocol,	5	Ability to handle concurrent transactions while ensuring data integrity.
VII	**Database Recovery Techniques:** Recovery concepts, Deferred updates technique, Immediate update technique.	3	To be familiar with the techniques of restoring a database from a failure to a consistent state while maintaining data integrity.
VIII	**Query Processing and Optimization:** Basic query operations, Heuristics in query optimization, cost estimates in query optimization.	3	Transform queries into an efficient form towards minimizing query execution time

IX	**Database Security and Authorization:** Discretionary access control, Mandatory access control and multi-level security, Statistical database security	3	Identify the need of database security and challenges involved and implement them through efficient management of user access and privileges.
X	**Advanced Topics in DBMS:** Introductory overviews on Distributed Databases, Object Oriented Databases, Need for non-relational databases, Types of NoSQL databases - Document, Key-Value, Column-Family, Graph. Key features and comparisons with relational databases, Basic operations - insert, retrieve, update, delete.	4	Ability to understand the need of database paradigms other than relational model, grasp the differences in basic concepts and applications over the conventional relational databases

Course Outcome	• CO1: Ability to design entity-relationship diagrams to represent simple database application scenarios • CO2: Skill to convert entity-relationship diagrams into relational tables (database design), populate a relational database and formulate SQL queries on the data • CO3: Crosscheck and improve the design by normalization • CO4: Master the basics of query evaluation techniques and query optimization • CO5: Understanding the role of transaction processing and concurrency control application development

Learning Resources	**Referred Books:** 1. Database System Concepts – Silberschatz, Korth, and Sudarshan (McGraw Hill). 2. Fundamentals of Database Systems – Elmasri, and Navathe (Benjamin Cummings Publishing Company Inc.). 3. Principles of Database Systems – J. D. Ullman (Galgotia Publications (P) Ltd.). 4. Database Systems – C. J. Date (Addison Wesley). 5. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence - Pramod J. Sadalage, Martin Fowler (Addison-Wesley).

Course Code	CS2204N	Course Name	Design & Analysis of Algorithm	Course Category	PC	L	T	P
						3	0	0

Pre-requisite Courses	Nil	Co-requisite Courses	Nil	Progressive Courses	Nil
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	• To introduce advanced ideas in design of algorithms; • To study the performance guarantees of algorithms; • to understand complexity classes, To introduce methods for coping with NP-hard problems.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Mathematical Foundations and Basic of Complexities:** Time and Space complexity, Asymptotic growth of functions, Recurrences and methods of solving recurrences (substitution, iteration, recursion tree, Master method). Worst, Average and Amortized complexities.	4	To equip the students with mathematical foundation that are needed to analyze algorithm.
II	**Design and Analysis techniques:** Divide and Conquer, Dynamic programming, Greedy Algorithms	4	To state different algorithm design techniques
III	**Example Algorithms for divide and conquer approach:** finding minimum and second minimum, Quick sort, Merge sort	4	To explain divide and conquer technique in detail
III	**Sorting and Order Statistics:** Quicksort and Merge Sort Complexity analysis as divide and conquer strategy, Lower bound for comparison based sorting, sorting in linear time (Counting, Radix and Bucket sort), Selection of Medians and ranked elements and their complexity	4	To explain algorithm to find element of any rank and thereby using it in quick sort
IV	**Example Algorithms for dynamic programming:** Matrix chain multiplication, Longest common subsequence, Polygon triangulation.	6	To explain dynamic programming technique in detail
V	**Example Algorithms for greedy strategy (selective list, not exhaustive):** Data compression, Matroid based formulation, Scheduling algorithm	4	To explain greedy strategy in detail
VI	**Advanced Data Structures and applications:** Data structures for dynamic sets, Hashing and associated search complexity, Data structures for disjoint sets, Complexity of union and find operations.	4	Understanding the advantage of modifying a data structure to improve the performance of certain operation
VII	**Graph algorithms:** Connected components of graph, Minimum Spanning Trees of graph, Single source and all-pair shortest paths	4	To be familiar with graph algorithm with few examples
VIII	**Number theoretic algorithm:** Fast exponentiation, GCD algorithm, Primality testing algorithm, Handling large size integers, Algorithms for public key cryptography	4	To introduce number theoretic algorithm and there use in public key cryptography
IX	**Concept of NP-Completeness:** Polynomial-time verification, Concept of NP-hard and NP-completeness, Notion of approximation Algorithms for NP-complete problems	4	To build understanding of NP-completeness.

Course Outcome	CO1: To comprehend and select algorithm design approaches in a problem specific manner. CO2: To develop sound theoretical understanding of advanced algorithms and practical problem solving skills using them CO3: To understand the necessary mathematical abstraction to solve problems. CO4: To come up with analysis of efficiency and proofs of correctness

Learning Resources	**References:** 1. Introduction to Algorithms, 3rd Edition,Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, MIT Press 2009, ISBN 978-0-262-03384-8. 2. Algorithm Design. Kleinberg, Jon, and Éva Tardos, Addison-Wesley, 2006.

Course Code	CS2261N	Course Name	Data Structures and Algorithms	Course Category	OE	L	T	P
						3	0	0

Pre-requisite Courses	Mathematics, Programming in C	Co-requisite Courses	*None*	Progressive Courses	*Algorithm, OOPD*
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	i) Understand and explain the fundamental concepts of data structures and their role in algorithm development and software design. ii) Analyze time and space complexity of algorithms using Big-O notation to evaluate performance. iii) Implement and apply linear data structures such as arrays, linked lists, stacks, and queues in problem-solving. iv) Understand and use non-linear data structures like trees and graphs, including traversal and searching techniques. v) Select and design appropriate data structures for various computational problems to improve efficiency and scalability.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Abstract Data Type (ADT) and Algorithm:** ADT - concepts of data types, data structure and ADT, properties applicable for ADT. Algorithm – Properties, Concepts of Time and Space complexity	4	Understand the concepts of Abstract Data Types (ADT), their properties, and analyze algorithm efficiency using time and space complexity.
II	**Linked Lists:** Linear Linked List, Circular Linked List, Doubly Linked List and their applications	5	Understand the types, operations, and applications of linear, circular, doubly, and multi-linked lists.
III	**Stacks and Queues:** Concepts and Applications	5	Understand the concepts, operations, and applications of stacks and queues.
IV	**Recursion:** Difference between Recursion and Iteration	2	Understand the difference between recursion and iteration.
V	**Trees:** Binary Trees. Properties, Binary Tree Traversals, Expression Trees, Conversion from General Tree to Binary Tree. Binary Search Trees and Operations on BST, Balanced Tree – AVL trees	10	Understand the structure, properties, and traversals of binary trees. Analyze and implement operations on binary search trees, balanced trees.
VI	**Heap:** Heap data structure and priority Queues	3	Understand the heap data structure and its application in implementing priority queues.
VII	**Graph:** Representations of Graph, Graph Traversal and its Applications	3	Understand graph representations, perform graph traversals, and apply them to solve real-world problems.
VIII	**Sorting:** Insertion Sorts, Exchange Sorts, Selection Sort, Merge Sort. Comparisons of Different Sorting Algorithms.	4	Understand and implement various sorting algorithms and compare their performance based on time and space complexity.
IX	**Searching:** Sequential Search, Sequential Search in Ordered List, Binary Search. Hashing and Hashing functions, collision resolution techniques.	6	Understand and implement various searching techniques and compare their efficiency in different scenarios. Understanding hash based search.

Course Outcome	CO1: Ability to select and design data structures and algorithms that are appropriate for a given problem. CO2: Ability to gauge how the choice of data structures and algorithm design methods impacts the performance of programs. CO3: Ability to analyze algorithms and to determine algorithm correctness and complexities. CO4: Skill to identify the scope for improving the performance (in terms of algorithmic betterment) of a given application. CO5: Ability to choose an appropriate algorithm out of different alternative algorithms for a given problem..

Learning Resources	**Referred Books:** 1. Data Structures using C and C++ by Y. Langsam, M. Augenstein and A. M. Tenenbaum 2. Data Structures and Program Design in C by R. Kruse and B. Leung 3. Fundamentals of Data Structures in C by E. Horowitz, S. Sahni, S. Anderson-Freed 4. Data Structures – A Pseudocode Approach with C++ by R. F. Gillbert and B. A. Forouzan

Course Code	CS2262N	Course Name	Object Oriented Technology	Course Category	OE	L	T	P
						3	0	0

Pre-requisite Courses	Data Structures	Co-requisite Courses	Nil	Progressive Courses	All subjects related to programming
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	1. To introduce various programming paradigms and highlight the evolution and rationale for object-oriented programming. 2. To develop a strong understanding of core object-oriented concepts and apply them effectively using C++. 3. To instill good software design principles, UML modeling skills, and promote modular, reusable, and maintainable code development. 4. To familiarize students with common design patterns and test-driven development practices for building robust and scalable software.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**Programming Paradigms and Evolution** Introduction to Programming Paradigms: Imperative, Functional, Logic, and Object-Oriented. Strengths and trade-offs of each paradigm. Evolution of programming languages and rationale for OOP.	4	Understand various programming paradigms, their strengths and trade-offs, and the evolution leading to the adoption of object-oriented programming.
II	**Foundations of Imperative and OOP** Concepts in Imperative Programming: Variables, assignment, control structures, state. Problems with procedural code: code reuse, maintainability, and software crisis. Principles of Object-Oriented Programming: Abstraction, Encapsulation, Inheritance, Polymorphism.	6	Grasp the core concepts of imperative programming, recognize its limitations, and understand how object-oriented principles address these challenges.
III	**C++ Basics and OOP Constructs** Overview of C++ syntax relevant to OOP. Classes, objects, constructors, destructors. Encapsulation and data hiding. Member functions and access specifiers. Operator Overloading and Function Overloading. Static members, friend functions, inline functions.	12	Learn the foundational C++ syntax and object-oriented constructs to effectively design and implement encapsulated, modular, and reusable code.
IV	**Inheritance and Polymorphism** Single and multiple inheritance. Constructor and destructor call order. Virtual functions and dynamic (runtime) polymorphism. Abstract classes and interfaces. Virtual base classes and diamond problem.	14	Understand inheritance and polymorphism in C++, including single and multiple inheritance, virtual functions, and mechanisms to resolve complexities like the diamond problem.
V	**Object-Oriented Design and UML** Object-oriented design principles. UML Basics: Class Diagrams, Sequence Diagrams Design Concepts: Coupling and Cohesion, Modularity and Reusability	6	Apply object-oriented design principles using UML to create modular, cohesive, and reusable software systems

Course Outcome	CO1: Understand and compare various programming paradigms and the evolution that led to the adoption of object-oriented programming. CO2: Apply object-oriented principles such as encapsulation, inheritance, polymorphism, and abstraction using C++ to design and develop modular and maintainable code. CO3: Analyze and implement object-oriented design concepts using UML diagrams

Learning Resources	**References and Books:** ● The C++ Programming Language by Bjarne Stroustrup ● C++ Primer by Stanley B. Lippman, Josée Lajoie, and Barbara E. Moo ● Object-Oriented Analysis and Design with Applications by Grady Booch ● Thinking in C++ (Volume 1) by Bruce Eckel

Course Code	CS2272N	Course Name	Computer Architecture and Organization Lab	Course Category	PC	L	T	P
						0	0	3

Pre-requisite Courses	Digital logic laboratory	Co-requisite Courses	Nil	Progressive Courses	Nil
Course Offering Department		CST		Data Book / Codes/Standards	Nil

Course Objectives	To provide students ● Hands-on experience in understanding the fundamental principles of computer systems. ● Developing practical skills in designing, implementing, and evaluating different units of computer systems. ● Expertise in digital design.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	Memory design and test (Two experiments)	12	Become familiar with design of memory module and read/write operation of the memory and verification of the data written.
II	Implementation of simple CPU instructions (One experiment)	6	Hands-on experience to realize simple instructions of instruction set architecture of a CPU.
III	Realization of data transfer among CPU registers and main memory (One experiment)	6	Becoming familiar with and realization of data transfer between buffer (DR) of CPU and main memory (RAM) and external world.
IV	Design of simple ALU (One experiment)	6	Expertise to design high speed adder/subtractor (and cascading) for ALU of the CPU.
V	Microprogrammed control design (One experiment)	6	Familiarization with a) Register level data transfer through common bus, and b) Microprogrammed realization of the register level data transfer.
VI	Hardwired control design (One experiment)	6	Developing expertise on relatively complex hardware design to explore functioning of the CPU.

Course Outcome	Upon successful completion, the students will be able to ● Design and verify different combinational/sequential circuits. ● Learn the testing of the memory subsystems. ● Design simple ALU of a CPU. ● Demonstrate the design of a simple instruction set computer.

Learning Resources	1. Computer System Architecture, M. Morris Mano, Pearson. 2. Laboratory manuals.

Course Code	CS2273N	**Course Name**	DBMS Lab	**Course Category**	PC	L	T	P
						0	0	3

Pre-requisite Courses	Nil	**Co-requisite Courses**	Nil	**Progressive Courses**	Nil
Course Offering Department		CST		**Data Book / Codes/Standards**	Nil

Course Objectives	1. Understand and apply SQL for creating and managing relational databases. 2. Perform DDL, DML, and complex queries using SQL clauses and joins. 3. Implement PL/SQL blocks, cursors, triggers, and stored procedures. 4. Manage transactions and user access using SQL commands. 5. Use basic NoSQL database operations.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	**SQL using Relational Databases** 1.Create databases and tables using DDL commands, Implement constraints. 2. Insert, update, and delete records using DML commands. 3. Retrieve data using SELECT queries with WHERE, ORDER BY clauses. 4. Use aggregate functions and GROUP BY, HAVING clauses. 5. Perform different types of joins between tables. 6. Perform nested queries and subqueries. 8. Create and use views, indexes, and sequences. 9. Implement transactions using COMMIT, ROLLBACK, and SAVEPOINT. 10. Creating Database Users, Use of DCL commands GRANT and REVOKE.	21	Students will be able to design and manage relational databases using SQL by creating tables, applying constraints, performing data manipulation and retrieval, using joins and subqueries, managing views and indexes, handling transactions, and controlling access with DCL commands.
II	**Procedural flavour with PL/SQL** Writing PL / SQL code blocks, Cursors and Triggers, Writing PL / SQL Stored Procedures.	12	Students will be able to develop procedural programs using PL/SQL by writing code blocks, implementing cursors and triggers, and creating stored procedures.
III	**NoSQL (eg. MongoDB) using Non-Relational Databases** 1. Create a database and collections using MongoDB. 2. Perform basic CRUD operations on documents. 3. Query documents using filters, projections, and conditions. 4. Work with embedded documents and arrays. 5. Create indexes and perform basic aggregation queries.	9	Students will be able to use NoSQL to create databases and collections, perform CRUD operations, query documents, handle embedded data structures, and apply indexing and basic aggregation techniques.

Course Outcome	CO1: Create and manage relational databases using SQL. CO2: Retrieve and manipulate data using queries and joins. CO3: Implement PL/SQL code with cursors, triggers, and procedures. CO4: Handle transactions and manage database users. CO5: Use of NoSQL operations to handle non-relational databases..

Learning Resources	**Referred Books:** 1. SQL, PL/SQL: The Programming Language of Oracle - Ivan Bayross (BPB Publications). 2. Oracle PL/SQL Programming - Steven Feuerstein (O'Reilly Media, 6th Edition). 3. MongoDB: The Definitive Guide - Kristina Chodorow, Michael Dirolf (O'Reilly Media, 3rd Edition).

Course Code	CS2274N	Course Name	Design and Analysis of Algorithm Lab	Course Category	PC	L	T	P
						0	0	3

Pre-requisite Courses	Nil	Co-requisite Courses	Nil	Progressive Courses	Nil
Course Offering Department		Name of the Department		Data Book / Codes/Standards	Nil

Course Objectives	The objective of this laboratory course is to provide hands-on experience in implementing and analyzing fundamental algorithms using appropriate programming languages. Through practical exercises, students will develop the skills to evaluate the time and space complexity of algorithms, compare different approaches to problem-solving, and enhance their ability to write efficient and optimized code.

Module	Syllabus	Duration (class-hour)	Module Outcome
I	Experimentation of Various comparison sort algorithms and comparing their efficiencies	6	To implement various sorting algorithm in a programming language
II	Applications of Divide-and-Conquer	9	To gain hands on experience of the algorithm
III	Applications of Dynamic Programming	6	To gain hands on experience of the algorithm
IV	Applications of Greedy algorithms	6	To gain hands on experience of the algorithm
V	Implementation of graph algorithms	9	To gain hands on experience of different algorithm such minimum spanning trees
VI	Implementation of few approximation algorithm	6	To gain hands on experience of few approximation algorithm

Course Outcome	By the end of this course, students will be able to: CO1. Implement fundamental algorithms using appropriate programming languages and development tools. CO2. Apply algorithmic techniques such as divide-and-conquer, greedy strategies, dynamic programming, and backtracking to solve real-world problems. CO3. Analyze the time and space complexity of algorithms through empirical testing and theoretical reasoning. CO4. Compare the performance of different algorithms for the same problem to determine the most efficient approach. CO5. Demonstrate the ability to debug, test, and optimize code for better performance and scalability.

Learning Resources	References: 1. Introduction to Algorithms, 3rd Edition,Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, MIT Press 2009, ISBN 978-0-262-03384-8. 2. Algorithm Design. Kleinberg, Jon, and Éva Tardos, Addison-Wesley, 2006.