

INFORMATION SCIENCE AND ENGINEERING

SCHEME and SYLLABUS OF III and IV SEMESTER B.E.

FOR THE A.Y.: 2026-27

Vision of the College:

To develop thoughtful and creative young minds in a learning environment of high academic ambience by synergising spiritual values and technological competence.

Mission of the College:

1. To continuously strive for the total development of students by educating them in state-of-the-art-technologies and managerial competencies providing best in class learning experience with emphasis on skills, values and learning outcomes and helping them imbibe professional ethics and societal commitment.
2. To create research ambience that promotes interdisciplinary research catering to the needs of industry and society.
3. To collaborate with premier academic and research institutions and industries to strengthen multidisciplinary education, applied research, innovation, entrepreneurship and consulting ecosystems.

Vision of the Department:

To be a centre for quality education and research in Information Science and Engineering to create high quality professionals for catering to the need of the society.

Mission of the Department:

- 1) To enable students to acquire strong fundamental concepts related to the Information Science and Engineering through experiential learning.
- 2) To educate students towards state-of-the-art-technologies and multidisciplinary practices for a successful career by creating learning-teaching-learning ambience.
- 3) To inculcate life-long learning through innovation and research attitudes among students related to Information Science and Engineering.

Program Educational Objectives (PEOs):

The objectives of Information Science and Engineering degree program are to prepare students to meet the academic excellence, professionalism, and ability to solve a broad range of problems in rapidly changing technological, economic and social environment.

Graduates of the program will:

1. Pursue career as software engineer, project manager, data scientist, entrepreneur and pursue higher studies and research in Information Science and Engineering domains.
2. Apply mathematical, scientific and Information Science and Engineering knowledge with multidisciplinary approaches to solve real world problems.
3. Possess professionalism, ethical and societal responsibilities and engage in life-long learning through pursuit of skill development and certification courses in Information Science and Engineering.

Programme Outcomes (POs):

To achieve the above objectives, Information Science and Engineering degree programme strives to obtain the following outcomes which should be achieved by all graduates at the time of their graduation.

Engineering Graduates will be able to:

- PO1: Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
- PO2: Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
- PO3: Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
- PO4: Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
- PO5: Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)
- PO6: The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
- PO7: Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
- PO8: Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
- PO9: Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences
- PO10: Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
- PO11: Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological

change. (WK8)

Programme Specific Outcomes (PSOs):

- 1) **Computing System:** Demonstrate the knowledge of evolving hardware and/or software to develop solutions to real life computational problems with a focus on performance optimization.
- 2) **Communication and Security:** Design and develop solutions for providing efficient transmission, storage, security and privacy of data in diverse computing environment.
- 3) **Information management:** Apply tools and techniques for management of information system, data analysis and knowledge discovery in the process of decision making.

**SCHEME and SYLLABUS
OF
III SEMESTER B.E.**

**INFORMATION SCIENCE AND ENGINEERING
FOR THE A.Y: 2026-27**

SCHEME OF TEACHING AND EXAMINATION : III Semester

Sl. No.	Course and Course Code		Course Title	Teaching / Paper setting Dept.	Teaching and Learning scheme				Total Hours / Sem.	Examination				Credits
					Lecture	Tutorial	Practical/ Drawing	TW&SL		Duration in hrs.	CIE Marks	SEE Marks	Total Marks	
					L	T	P	SL						
1.	ASC /PCC	P3MAT1	Statistics and Probability	Maths department	42	28	0	50	120	3	50	50	100	4
2.	IPCC	P3ISI1	Object Oriented Programming with JAVA	ISE	42	0	28	50	120	3	50	50	100	4
3.	PCC	P3IS1	Digital Circuits and Computer Organization	ISE	42	28	0	50	120	3	50	50	100	4
4.	PCC	P3IS2	Operating Systems	ISE	42	0	0	48	90	3	50	50	100	3
5.	PCC	P3IS3	Data Structures	ISE	42	0	0	48	90	3	50	50	100	3
6.	PCCL	P3ISL1	Data Structures Laboratory	ISE	0	0	28	2	30	2	50	50	100	1
7.	AEC	P3ISAX	Ability Enhancement Course Laboratory	ISE	0	0	28	2	30	2	50	50	100	1
8.	SDC	SDCIS3	Community Project (Project-Based Learning)	ISE	0	0	0	30	30	2	50	50	100	1
9.	NCMC	PMC1	National Service Scheme (NSS)	NSS coordinator	0	0	0	28	28	--	100	--	100	PP
		PMC2	Physical Education (PE) (Sports and Athletics)	Ph.Edn. Director										
		PMC3	Yoga (Yoga Teacher)											
		PMC4	Music (Music Teacher)											
			Total								500	400	900	21
		AAP	AICTE Activity Points (Regular students have to earn 100 activity points from I to VI semesters and Lateral Entry students have to earn 75 activity points from III to VI semesters)	Compulsory requirement for the award of a degree										
10.	NCMC	P3MATDIP	Mathematics course for Lateral Entry Students	Maths Dept.	14	0	0	14	28	3	100	---	100	PP
Note: ASC: Applied Science Course PCC: Professional Core Course, IPCC: Integrated Professional Core Course, PCCL: Professional Core Course laboratory, AEC: Ability Enhancement Course, SDC: Skill Development Course NCMC: Non Credit Mandatory Course L: Lecture, T: Tutorial, P: Practical SL: Self Learning TW&SL: Team works & Self Learning CIE: Continuous Internal Evaluation, SEE: Semester End Evaluation.														
Ability Enhancement Course (Laboratory)														
P3ISA1		Data Visualization		P3ISA2		TypeScripting								
P3ISA3		UNIX and Shell Programming		P3ISA4		Prompt Engineering								
Ability Enhancement Course Laboratory can be offered either as a single compulsory course or as multiple elective options.														

Community Project: A community is a social unit or group of people sharing socially-significant characteristics, such as place, set of norms, culture, religion, values, customs or identity. A community project involves addressing issues or needs within such a community or a network of entities working toward a common purpose. These projects may cover a wide range of areas, including welfare, sustainability, technology integration, and social development. Examples include establishing and maintaining an orphanage, implementing solar power generation and its maintenance, or developing environmental improvement solutions, etc. A community project is an experiential learning activity that encourages students to identify, analyse, and address real-life problems of the community using engineering knowledge. It aims to promote social responsibility and civic engagement, interdisciplinary thinking and collaboration and practical application of theoretical concepts, thereby enabling students to contribute meaningfully to community welfare and sustainable development. Students can take up project individually or in a group not exceeding 4 students.

The evaluation shall be done as per the following;

CIE: The CIE marks shall be awarded by a committee consisting of the Head of the concerned Department and two senior faculty members of the Department, one of whom shall be the Guide. The CIE marks awarded for the project work shall be based on the rubrics.

SEE: SEE will be conducted by the two examiners appointed by the Institute. The SEE marks awarded for the project work shall be based on the rubrics.

National Service Scheme / Physical Education / Yoga / Music (NSS / PE / YOG / Music): All students are required to register for any one of the following courses; National Service Scheme (NSS), Physical Education (PE) (Sports and Athletics), Yoga (YOG) or Music (MUSIC) during the first week of the third semester through online.

- Students may opt for different activities/options across semesters. For instance, a student participating in PE during 3rd semester may choose NSS in the 4th semester or Yoga.
- Activities shall be conducted over two semesters (III & IV), and successful completion of the registered course / or courses along with the required CIE score is mandatory for the award of the degree.
- The events are appropriately scheduled and reflected in the semester-wise calendar for NSS, PE, Music, and Yoga activities.
- These courses shall not be considered for the calculation of SGPA or CGPA and for vertical progression. However, completion of course(s) is compulsory for degree eligibility.

Statistics and Probability

Contact Hours/ Week:	3L+2T	Credits:	4
Total Lecture Hours:	42	CIE Marks:	50
Total Tutorial Hours:	28	SEE Marks:	50
Sub. Code:	P3MAT1		

Course objectives:

This course will enable students to:

1.	Develop and conduct appropriate experimentation, analyze and interpret data and use engineering judgment to draw conclusion.
2.	Introduce the basic concepts and applications of probability in engineering.
3.	Introduce the random variable, random process and modeling of random processes in engineering.
4.	Deal with multiple random variables and stochastic process.
5.	Investigate the variability in sample statistics from sample, to measure of central tendency; dispersion of sample statistics and pattern of variability of sample.

Module I: Statistics

Introduction, Curve Fitting: Straight line, reducible to Linear and Quadratic form-parabola. Definition of Correlation and regression lines, formula for correlation coefficient, regression lines with proof and angle between the regression lines, Rank correlation.

(8 Hours)

Module II: Probability

Basic terminology, Definition of probability, Probability and set notations, Types of events, Addition law of probability, conditional probability, multiplication law of probability, Baye's theorem.

(9 Hours)

Module III: Random Variable

Definition of Random Variable, Discrete Probability distribution, expectation, Variance, Binomial distribution, Poisson distribution. Continuous Probability distribution- expectation, Variance, Normal distribution and Exponential distributions.

(9 Hours)

Module IV: Joint Probability

Joint probability distribution, Discrete and independent random variables, Expectation, Covariance, Correlation coefficient. Probability vectors, stochastic matrices, fixed point matrices, Regular stochastic matrices, Markov chains, Higher transition-probabilities, stationary distribution of regular Markov chains and absorbing states.

(8 Hours)

Module V: Sampling Distribution

Introduction, Objectives, sampling distribution, testing of hypothesis, level of significance, confidence limits, simple sampling of attributes, test of significance of large samples, comparison of large samples, sampling of variables, central limit theorem, confidence limits for unknown mean, test of significance for means of two large samples, Sampling of variables – small samples, Student's t-distribution.

(8 Hours)

TEXT BOOKS

1	B.S.Grewal	“Higher Engineering Mathematics”, 43 rd edition, Khanna Publications, 2015. ISBN:9788174091955.
2	Ramana .B.V	“Higher Engineering Mathematics”, latest edition, Tata-McGraw Hill, 2016. ISBN;0070535167.

REFERENCE BOOKS

1	Erwin Kreyszig	“Advanced Engineering Mathematics”, 10 th edition, Wiley Publications, 2015. ISBN;9788174091955.
2	C. Ray Wylie and Louis C. Barrett	“Advanced Engineering Mathematics”, 6 th Edition, Tata-McGraw Hill 2005, ISBN: 978-0071135436.
3	Louis A. Pipes and Lawrence R. Harvill	“Applied Mathematics for Engineers and Physicists”, 3 rd Edition, McGraw Hill, 2014, ISBN: 978-0486779515.

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Apply least square method to fit a curve for the given data and evaluate the correlation coefficient and regression lines for the data.
CO2	Analyze the nature of the events and hence determine the appropriate probabilities of the events.
CO3	Classify the random variables to determine the appropriate probability distributions and hence compute the associated probability.
CO4	Compute the joint probability and its parameters. Predicts the long run behavior of a Markov chain using transition matrix.
CO5	Estimate the parameters of a population and sample in testing of hypothesis.

Mapping of Course Outcomes (COs) to Program Outcomes (POs) & Program Specific Outcomes (PSOs)

		POs										
		1	2	3	4	5	6	7	8	9	10	11
COs	CO1	3	1									
	CO2	3	1									
	CO3	3	1									
	CO4	3	1									
	CO5	3	1									

1: Low association 2: Moderate association 3: High association

Object Oriented Programming with Java (IPCC)

Contact Hours/ Week:	3L+2P	Credits:	4
Total Lecture Hours:	42	CIE Marks:	50
Sub. Code:	P3ISI1	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Understand the basics of Java programming and object-oriented concepts.
2.	Learn inheritance, packages, and interfaces to develop reusable Java programs.
3.	Apply exception handling and multithreading concepts to develop reliable Java applications.
4.	Understand generics and Java I/O streams for type-safe programming and file handling.
5.	Use the Collections Framework for storing and managing data efficiently.

UNIT I

An Overview of Java: Object-Oriented Programming, A First Simple Program, A Second Short Program, Two Control Statements, Using Blocks of Code, Lexical Issues.

Introducing Classes: Class Fundamentals, Declaring Objects, Assigning Object Reference Variables, Introducing Methods, Constructors, The this Keyword, Garbage Collection, The finalize() Method, A Stack Class, A Closer Look at Methods and Classes: Overloading Methods, Using Objects as Parameters, A Closer Look at Argument Passing, Returning Objects, Recursion, Introducing Access Control, Understanding static, Introducing final, Arrays Revisited.

8 Hours

UNIT II

Inheritance: Inheritance, Using super, Creating a Multilevel Hierarchy, When Constructors are Called, Method Overriding, Dynamic Method Dispatch, Using Abstract Classes, Using final with Inheritance, The Object Class.

Packages and Interfaces: Packages, Access Protection, Importing Packages, Interfaces.

8 Hours

UNIT III

Exception Handling: Exception-Handling Fundamentals, Exception Types, Uncaught Exceptions, Using try and catch, Multiple catch Clauses, Nested try Statements, throw, throws, finally, Java's Built-in Exceptions, Creating Your Own Exception Subclasses, Chained Exceptions, Using Exceptions.

Multithreading: Thread life cycle, The Main Thread, Creating a Thread, Creating Multiple Threads, Using isAlive() and join(), Thread Priorities, Synchronization, Interthread Communication, Suspending, Resuming, and Stopping Threads.

8 Hours

UNIT IV

Generics: What Are Generics?, A Simple Generics Example, A Generic Class with Two Type Parameters, The General Form of a Generic Class, Bounded Types, Using Wildcard Arguments, Creating a Generic Method, Generic Interfaces.

Input/Output: Exploring java.io: The Java I/O Classes and Interfaces. File, The Closeable and Flushable Interfaces, The Stream Classes, The Byte Streams - InputStream, OutputStream, FileInputStream, FileOutputStream.

9 Hours

UNIT V		
The Collections Framework: Collections Overview, Recent Changes to Collections, The Collection Interfaces, The Collection Classes, Accessing a Collection via an Iterator. Storing User-Defined Classes in Collections. The RandomAccess Interface. Working with Maps.		
9 Hours		

TEXT BOOKS		
1	Herbert Schildt	Java The Complete Reference, 13 th Edition, Tata McGraw Hill, 2024.

REFERENCE BOOKS		
1	Daniel Liang	Introduction to Java Programming (Comprehensive Version), 13 th Edition, Pearson, 2024.
2	Cay S Horstmann	Core Java: Volume 1 Fundamentals, Pearson Education, 13 th Edition, 2024.

Course Outcomes:	
Upon completion of this course the student will be able to:	
CO1.	Apply object-oriented programming concepts to construct basic Java programs using classes, methods, and constructors.
CO2.	Develop Java applications using inheritance, packages, and interfaces.
CO3.	Implement modular Java applications using error-handling mechanisms and multithreading concepts.
CO4.	Apply generics and Java I/O stream classes to build reusable Java applications.
CO5.	Design efficient, reusable Java solutions to real-world problems using the collections framework.

Mapping of Course Outcomes (COs) to Program Outcomes (POs) & Program Specific Outcomes (PSOs)

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3	2			2			2	2			3		
	CO2	3	2	2		2			2	2			3		
	CO3	3	2	2		3			2	3			3		
	CO4	3	3	2		3			2	2			3		
	CO5	3	3	2		3			2	3			3		
AVG		3	3	2		3			2	3			3		

1: Low association 2: Moderate association 3: High association

Experiment No.	Lab Experiments
1	A college wants to automate the management of student records. Each student has a USN, name, and marks in three subjects. The system should calculate the average and display student details. Task: - Create a Student class with appropriate data members. - Use constructors to initialize values.

	• Implement methods to calculate average marks and display details. • Create objects for at least 5 students and display their results.
2	A bank requires a simple software module to manage customer accounts and transactions. Task: • Create a BankAccount class with account number, customer name, and balance. • Implement deposit and withdrawal methods. • Use this keyword in constructors. • Display updated account details after transactions.
3	A company employs both permanent and contract employees. The salary calculation differs for each category. Task: • Create a base class Employee. • Derive PermanentEmployee and ContractEmployee classes. • Override the salary calculation method. • Demonstrate dynamic method dispatch using object references.
4	An e-commerce company supports multiple payment methods such as UPI, Credit Card, and Net Banking. Task: • Create an interface Payment. • Implement the interface in different payment classes. • Use packages to organize the classes. • Display payment confirmation for different payment modes.
5	A bank ATM must prevent invalid transactions and maintain transaction safety. Task: • Accept account balance and withdrawal amount. • Throw a custom exception if the withdrawal amount exceeds the balance. • Handle exceptions using try-catch-finally. • Display appropriate transaction messages.
6	A railway reservation counter handles multiple booking requests simultaneously. Task: • Create multiple threads representing passengers. • Simulate ticket booking using synchronization. • Display thread status using isAlive() and join(). • Ensure no overbooking occurs.
7	A university maintains records of students and faculty members using a common storage mechanism. Task: Create a generic class to store and display records. • Use two type parameters for ID and Name. • Demonstrate generic methods for retrieving records. • Display stored information for students and faculty.
8	A digital library stores book information in text files and retrieves them when required. Task: • Create a file to store book details. • Use FileOutputStream to write data into the file. • Use FileInputStream to read and display file contents. • Handle file-related exceptions appropriately.

9	A college maintains a list of students registered for different courses. Task: • Create a user-defined class Student. • Store student objects in an ArrayList. • Use an Iterator to display student details. • Search and display students registered for a specific course.
10	A retail store maintains product details and stock information for efficient inventory management. Task: • Create a Product class. • Store product ID and stock details using a HashMap. • Perform insertion, update, and retrieval operations. • Display all product records and available stock information.

Digital Circuits and Computer Organization

Contact Hours/ Week:	3L+2T	Credits:	4
Total Lecture Hours:	42	CIE Marks:	50
Total Tutorial Hours:	28	SEE Marks:	50
Sub. Code:	P3IS1		

Course objectives:

This course will enable students to:

1	Learn different techniques of simplification of Boolean expressions and design of combinational circuits.
2	Describe different sequential logic circuits.
3	Recognize the significance of number representation, arithmetic operations, memory locations, and machine instructions.
4	Learn various addressing modes including implementation of variables and constants, indirection and pointers, indexing, arrays, and relative addressing.
5	Explore different techniques of performing arithmetic operations.

UNIT I

Introduction to Digital System: Boolean Algebra and Boolean Functions, min-term and max-term notations. Simplification of Boolean expressions using Boolean laws and Rules, Simplification of Boolean expressions using Karnaugh map techniques. Design of combinational Logic Circuits: Design of Decoders, Multiplexers, Demultiplexers.

8 Hours

UNIT II

Sequential Logic Circuits

Flip Flops: Introduction to Flip-Flops, Types of Flip-Flops: RS FF, SR FF, JK FF and M/S JK FF. Counters: Definition of Counter, Types of Counters, Design of Asynchronous Counters.

Registers: Basic Register, Shift-Register, Types of Shift Registers, Unidirectional and Bidirectional Shift Register, Johnson counter and Ring Counters.

8 Hours

UNIT III

Basic Structure of Computer: Functional Units, Basic Operational Concepts, Bus Structures, Performance - Processor Clock, Basic Performance Equation, Clock Rate, Performance Measurement. Machine Instructions and Programs: Numbers, Arithmetic Operations and Characters - Number Representation, Addition of Positive Numbers, Addition and Subtraction of Signed Numbers, Overflow in Integer Arithmetic, Characters, Memory Location and Addresses - Byte Addressability. **The CISC and RISC machines.**

9 Hours

UNIT IV

Addressing Modes: Implementation of Variables and Constants, Indirection and Pointers, Indexing and Arrays, Relative Addressing, Additional Modes.

Basic Processing Unit: Execution of a Complete Instruction - Branch Instructions, Multiple Bus Organization, Hard wired Control - A Complete Processor, Micro programmed Control - Microinstructions.

8 Hours

UNIT V

Arithmetic: Addition and Subtraction of Signed Numbers - Addition/Subtraction Logic Unit, Multiplication of Positive Numbers, Signed Operand Multiplication - Booth Algorithm, Fast Multiplication - Bit-Pair Recoding of Multipliers, Carry-Save Addition of Summands, Integer Division, Floating-point Numbers and Operations - IEEE Standard for Floating-Point Numbers,

Arithmetic Operations on Floating-Point Numbers - Addition and Subtraction Operations.

9 Hours**TEXT BOOKS**

1	Thomas L Floyd	Digital Fundamentals, Pearson Publications. 11 th Edition, 2020.
2	Carl Hamacher, Zvonko Vranesic, SafwatZaky	Computer Organization, Tata McGraw Hill, 5 th Edition, 2017.

REFERENCE BOOKS

1	Donald P Leach, Albert Paul Malvino and Goutam Saha	Digital Principles and Applications, Tata McGraw – Hill, Ed.8, 2014.
2	Ronald J Tocci, Neil S Widmer, Gegory L. Moss,	Digital Systems-Principles and Applications, Pearson Education, 12 th Edition, 2016.

Tutorials (Design of Digital System for Real World Problems)

- Design and implementation of Half-adder and Full adder using minimum number of NAND gates only.
- Design and implement the Full adder and Full subtractor using only multiplexer and other gates.
- Design and implement the Full adder and Full subtractor using decoder and other logic gates.
- Implement the following using 4-bit shift register IC 74LS95.
a) Right Shift b) SISO c) PIPO d) PISO e) SIPO
f) Left Shift g) Ring Counter h) Johnson Counter
- Design a sequence generator to generate the given sequence using a) shift Register IC and other gates and b) Multiplexer and counter.
- Design a 3-stage Asynchronous (mod-8) counter using MS J-K flip flops IC7476 and 7490IC.
- Implement UP-Down pre-settable counter using IC 74LS190 for the given mod N.

Course Outcomes:

Upon completion of this course the student will be able to:

CO1.	Design combinational logic circuits for the required applications.
CO2.	Design sequential logic circuits such as counters and registers for the specific needs.
CO3.	Analyze the performance of a basic computer system.
CO4.	Apply the knowledge of addressing modes to develop assembly language code and Design control sequence for the given instruction on different CPU bus structures.
CO5.	Apply appropriate technique to solve arithmetic related problems in computer.

Mapping of Course Outcomes (COs) to Program Outcomes (POs) & Program Specific Outcomes (PSOs)

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3	2										3		
	CO2	3	2	2									3		
	CO3	3	2	1									3		
	CO4	3	2	1									3		
	CO5	2	1	1									3		
AVG		3	2	1									3		
1: Low association 2: Moderate association 3: High association															

Operating Systems

Contact Hours/Week	:	3	Credits	:	3.0
Total Lecture Hours	:	42	CIE Marks	:	50
Course Code	:	P3IS2	SEE Marks	:	50

Course objectives:

This course will enable students to:

1. Define fundamental OS abstractions such as processes, threads, files etc.
2. Visualize the intricate relationship between an operating system and its underlying hardware.
3. Explain scheduling algorithms, deadlock detection algorithms and memory management strategies.
4. Apply the principles of concurrency and synchronization, to write concurrent programs/software.

UNIT I

INTRODUCTION: What operating systems do - User view, System view, Defining operating systems, Operating System Structure, Operating System Operations – Dual mode and multi-mode operation, Timer, Process Management; Memory Management; Storage Management; Protection and Security. [Textbook 1: chapters 1.1, 1.4 to 1.9]

SYSTEM STRUCTURES: Operating System Services; System calls; Types of system calls; System programs; Operating System Structure –Simple structure, Layered approach, Micro kernels, Modules [Textbook 1: Chapters 2.1, 2.3 to 2.5, 2.7.1 to 2.7.4]

Self study : Hybrid Systems – Mac OS X, iOS, Android.[Textbook 1: Chapter 2.7.5]

8 Hours

UNIT II

PROCESS: Process concept, Process state, Process control block, Process scheduling, Scheduling queues, Schedulers, Context switch, Operations on processes – Process creation and termination, Inter-process communication, Shared memory and message passing systems. [Textbook 1: Chapters 3.1 to 3.4]

PROCESS MANAGEMENT: Basic concepts, CPU scheduler, Preemptive and non-preemptive scheduling, Scheduling criteria, Scheduling algorithms – FCFS, SJF, Priority and Round robin scheduling. [Textbook 1: Chapters 5.1 to 5.3.4]

Self Study : Multi-level and multilevel feedback queue scheduling.[Textbook 1: Chapter 5.3.5 and 5.3.6]

8 Hours

UNIT III

THREADS: Overview, Benefits, Multi core Programming, Types of parallelism, Multi threading models. [Textbook 1: Chapters 4.1-4.3]

PROCESS SYNCHRONIZATION: Background, The Critical section problem, Peterson's solution, Synchronization hardware, Mutex locks, Semaphores, Classic problems of synchronization, Bounded buffer problem. [Textbook 1: Chapters 6.1 to 6.7.1]

Self-Study: Readers writer's problem, Dining philosopher's problem. [Textbook 1: Chapters 6.7.2 , 6.7.3]

8 Hours

UNIT IV

DEADLOCKS: System model, Deadlock characterization, Methods for handling deadlocks, Deadlock prevention, Deadlock avoidance, Deadlock detection and recovery from deadlock. [Textbook 1:7.1-7.7]

MEMORY MANAGEMENT: Background, Basic hardware, Address binding, Logical and physical address, swapping, Dynamic loading and linking. [Textbook 1: Chapters 8.1, 8.2]

8 Hours

UNIT V

MEMORY MANAGEMENT: Contiguous memory allocation, Segmentation, Paging. [Textbook 1: 8.3, 8.4, 8.5]

VIRTUAL MEMORY MANAGEMENT: Basic concepts, Demand paging, Copy-on-write, Page replacement – FIFO, LRU, Optimal. [Textbook 1: 9.1-9.4]

Self-Study: Structure of page table, Hierarchical paging, Hashed paging, Inverted paging. [Textbook 1: Chapter 8.6]

8 Hours

NOTE: Self-study topics are assessed only in CIE-Quizzes/ Assignment

TEXT BOOKS

1	Abraham Silberschatz, Peter Baer Galvin, Greg Gagne	Operating System Concepts Wiley-India, 9th Edition, 2013.
---	---	---

REFERENCE BOOKS

1	D.M Dhamdhare	Operating System - A Concept Based Approach, Tata McGraw-Hill, 2nd Edition, 2002.
2.	P.C.P. Bhatt	Operating Systems, PHI 4th Edition, 2013.

Course Outcomes:

Upon completion of this course the student will be able to:

CO1:	Illustrate the structure, functions and services of different operating systems and their major components.
CO2:	Select appropriate scheduling algorithm for process selection and execution.
CO3:	Analyze the techniques used for addressing process synchronization issues and inter-process communication mechanisms.
CO4:	Analyze various deadlock prevention, avoidance, detection and recovery mechanisms to ensure safe process execution.
CO5:	Evaluate the performance of various memory management techniques and page replacement algorithms to improve system efficiency.

Mapping of Course Outcomes (COs) to Program Outcomes (POs) & Program Specific Outcomes (PSOs)

		POs												PSOs		
		1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
COs	CO1	3												3		
	CO2	3	3						1	1				3		
	CO3	3	2						1	1				2		
	CO4	3	3						1	1				3		
	CO5	3	2						1	1				2		
		3	3						1	1				3		

1: Low association 2: Moderate association 3: High association

Data Structures

Contact Hours/ Week:	3	Credits:	3
Total Lecture Hours:	42	CIE Marks:	50
Course Code:	P3IS3	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Describe the efficient data storage mechanisms for easy access.
2.	Describe the properties of various data structures such as stacks, queues, lists, and trees.
3.	Implement Stack and Queue data structures and their applications.
4.	Design and implement various types of linked lists, trees and their applications.
5.	Apply the knowledge to select an appropriate data structure for a problem to be solved.

UNIT I

Structures and Pointers: Defining a Structure, declaring Structure variables, accessing Structure members, Structure initialization, Structure within Structure, Structures and Functions, Pointers: Defining a pointer, operations, accessing structures members using pointers.

The Stack: Definition and Examples, representing Stacks in C, Applications of stacks: Postfix, Infix and Prefix expressions, conversion and evaluation.

Text book 1: Chapter 10, 12

8 Hours

UNIT II

Recursion: Recursive Definition and Processes, Recursion in C, Writing recursive programs: The Towers of Hanoi Problem, Efficiency of Recursion.

Queues: The Queue and Its Sequential Representation: C implementation of Queues, Insertion, Deletion and Display operations, Types of Queues (Linear, Circular).

Text Book 2: Chapter 2, 3, 4 (selected topics)

9 Hours

UNIT III

Dynamic Memory Allocation: malloc(), calloc(), realloc(), free().

Linked lists: Linked List terminology, Operations on list: Inserting, Deleting, Modifying, and Displaying. Applications of Linked List: Stacks, Queue implementation, Addition of two polynomials, implementing header nodes, examples of list operation, list implementation of priority queues, header nodes.

Text Book 1: Chapter 13.1-13.6

Text Book 2: Chapter 4 (selected topics)

9 Hours

UNIT IV

Other List Structures: Circular lists: primitive operations on circular lists, Stack as a Circular list, Queue as a Circular list, Josephus problem, Doubly linked lists: Primitive operations, Implementation of Doubly linked list, Double Ended Queues, Examples.

Text Book 2: Chapter 4 (selected topics)

8 Hours

UNIT V	
Trees: Tree terminology, Representation and Operations on Trees, Types of Trees, Binary Search Tree, Binary Tree Traversals, Applications of Binary Trees: Constructing BST, General Expressions as Trees, Evaluating an Expression Tree. Graphs: Graph terminology, Representation of graphs, Breadth First Search, Depth First Search. Text Book 2: Chapter 5, 8 (selected topics)	
8 Hours	

TEXT BOOKS		
1	Yedidyah Langsam, Moshe J. Augenstein, Aaron M. Tenenbaum	Data structures using C and C++, PHI/Pearson, 2nd Edition, 2015. (Unit I to V)
2	E. Balagurusamy	Programming in ANSI C, 8th Edition, Tata McGraw-Hill Publications, 2019. (Unit I)

REFERENCE BOOKS		
1	Horowitz, Sahni and Anderson-Freed	Fundamentals of Data Structures in C, 2nd Edition, Universities Press Pvt. Ltd., 2011

Course Outcomes:	
Upon completion of this course the student will be able to:	
CO1	Apply the concepts of pointers and structures to develop solutions for computational problems.
CO2	Develop programs for performing operations on stack and queue data structures using static memory allocation technique.
CO3	Develop various types of linked lists using dynamic memory allocation technique.
CO4	Apply the knowledge of Stacks, Queues and linked lists to solve computational problems.
CO5	Implement non-linear data structures such as trees, graphs and their applications using dynamic/static memory allocation technique.

Course Articulation Matrix

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3							1	1			3		
	CO2	3		2					1	1			3		
	CO3			3					1	1			3		
	CO4	3		2					1	1			3		
	CO5			2					1	1			3		
AVG		3		2					1	1			3		

1: Low association 2: Moderate association 3: High association

Data Structures Laboratory

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P3ISL1	SEE Marks:	50

Course objectives:

This course will enable students to:

1	Write programs for implementing Linear data structures such as stacks, queues and their applications using static memory allocation.
2	Implement Linear data structures such as linked lists using dynamic memory allocation.
3	Explore the applications of linked lists and implement the same.
4	Develop programs for implementing Non-Linear data structures such as trees and graphs.

Experiment No.	Lab Exercises
1	Modern web browsers maintain a history of visited pages and allow users to move backward through previously accessed pages. This functionality is commonly implemented using a stack data structure. Design and implement a Browser Navigation History Management System using an array-based stack of integers, where each integer represents a unique webpage ID. The system should support the following operations: • Push: Add a newly visited webpage ID to the browsing history. • Pop: Remove the most recently visited webpage ID when the user clicks the “Back” button. • Display: Show all webpage IDs currently stored in browsing history. The program should provide a menu-driven interface and demonstrate stack operations efficiently.
2	Scientific calculators, compilers, and digital assistants internally process mathematical expressions in different formats for easier computation and optimization. Develop a Smart Calculator System in C that can handle user-entered arithmetic (infix) expressions and perform the following tasks: a. A compiler translating human-readable expressions into executable (postfix) expression. b. Convert a standard mathematical expression entered in infix notation into prefix notation for faster recursive evaluation. c. Evaluate a mathematical expression provided in prefix or postfix notation and display the computed result.
3	In automated parking facilities, vehicles are assigned parking slots in a cyclic manner to maximize slot utilization. A circular queue helps efficiently

	manage available slots without wasting memory. Develop a Smart Parking Slot Allocation System using a circular queue of integers, where each integer represents a parking slot number. The system should support the following operations: • Insertion: Add a newly available parking slot into the system. • Deletion: Allocate the next available parking slot to an arriving vehicle. • Display: Show all currently available parking slots in cyclic order. This system should demonstrate how circular queues are applied in resource management systems.
4	Educational institutions frequently update student records as new students enroll, existing records are modified, or reports are generated. A singly linked list is suitable for managing such dynamic data efficiently. Design and implement a Student Enrollment Record Management System using a singly linked list with integer data, where each integer represents a student roll number. The application should support the following functionalities: • Insert at End: Add a newly enrolled student record at the end of the list. • Delete First Node: Remove the earliest student record when a student withdraws. • Insert at Specified Position: Insert a student record at a particular rank or position in the enrollment list. • Display: Show all student roll numbers currently stored. • Reverse List: Reverse the order of records for administrative processing or reporting. The program should be menu-driven and simulate real-world student record handling.
5	In operating systems, CPU processes are scheduled efficiently using circular queues to utilize resources optimally and avoid wastage of unused resources. Develop a CPU Task Scheduling System using a circular queue (linked list implementation), where each task is represented by an integer (task ID). The system should support: • Insertion (Enqueue): Add a new task to the scheduling queue. • Deletion (Dequeue): Execute and remove the next task from the queue. • Display: Show all tasks currently waiting for execution. The circular nature of the queue should ensure that freed spaces are reused efficiently, simulating real-time system scheduling.
6	An e-commerce company maintains two sorted product catalogs from different vendors. To analyze inventory overlap and uniqueness, the company needs a system to combine and compare these catalogs efficiently. Develop and implement a Product Catalog Comparison System using two ordered singly linked lists, where each node stores a product ID (integer).

	The system should support: • Insert into Catalog 1 • Insert into Catalog 2 • Union Operation: Generate a combined catalog containing all unique product IDs from both vendors. • Intersection Operation: Identify common products available in both catalogs. • Display: Show the contents of Catalog 1, Catalog 2, and the resulting list.
7	A library requires a dynamic system to maintain book records efficiently, with frequent insertions, deletions, updates, and tracking of total inventory. Develop and implement a Library Inventory Management System using a doubly linked list with a header node, where the header stores the count of books. The system should support: • Insert a book record at the end • Insert a new record next to a specified book • Delete the first record if pointer to last node is available • Delete a record by book ID • Display all records • Swap n^{th} and m^{th} records The header node must update the total count after every operation.
8	An emergency response center manages high-priority and normal-priority requests differently. High-priority requests may be inserted at the front, while regular requests are added at the rear. Develop an Emergency Dispatch Queue System using a deque (double-ended queue) implemented with a doubly linked list. The system should perform: • Insertion: Add requests at either front or rear. • Deletion: Remove requests from either front or rear. • Display: Show all pending requests. This simulates real-time double-ended scheduling.
9	A company stores employee IDs in a structured format to enable fast searching, insertion, and deletion. Design and implement an Employee Record Search System using a Binary Search Tree (BST). The system should support: • Insert: Add a new employee ID. • Traversal: Display records in inorder, preorder, and postorder. • Delete: Remove a given employee ID. This demonstrates efficient hierarchical storage and retrieval.
10	Scientific applications and compilers use expression trees to represent and evaluate mathematical expressions efficiently. Develop an Automated

	Formula Evaluation Engine that: - Constructs an expression tree from a valid postfix expression. - Evaluates the expression tree to compute the final result. This system simulates internal processing used in compilers, calculators, and symbolic computation tools.
11	Modern navigation systems must determine the routes between different locations in a city. Roads connect different places, and each road may have a distance or travel cost associated with it. - Implement the graph traversal techniques for finding the connectivity from a particular location to all other locations in a city using Breadth First Search (BFS). - Develop a program to check whether all the cities are connected or not using Depth First Search algorithm.

TEXT BOOKS

1.	Yedidyah Langsam, Moshe J. Augenstein, Aaron M. Tenenbaum	Data structures using C and C++, PHI/Pearson, 2nd Edition, 2015. (Unit II to V)
2.	E. Balagurusamy	Programming in ANSI C, 8th Edition, Tata McGraw-Hill Publications, 2019. (Unit I)

REFERENCE BOOKS

1	Horowitz, Sartaj Sahni and Anderson-Freed	Fundamentals of Data Structures in C, 2 nd Edition, Universities Press Pvt. Ltd., 2011
---	---	---

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Develop C programs by applying C programming techniques like pointers and structures to develop solutions for particular problems.
CO2	Develop Linear data structures like Stack, Queue using static memory allocation technique and explore their applications.
CO3	Develop Linear data structures like Linked Lists using dynamic memory allocation technique.
CO4	Develop programs for performing various operations on non-linear data structures like Trees and graphs by using dynamic/static memory allocation.

Course Articulation Matrix

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
Cos	CO1	3	2	3		3							3		
	CO2	3	2	3		3							3		
	CO3	3	2	3		3							3		
	CO4	3	2	3		3							3		
AVG		3	2	3									3		

1: Low association 2: Moderate association 3: High association

Data Visualization

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P3ISA1	SEE Marks:	50

Course objectives:

This course will enable students to:

1	To understand the fundamentals of data analytics and visualization using tools such as R, Tableau, and Power BI.
2	To apply statistical and programming techniques for data manipulation, analysis, and interpretation.
3	To develop skills in creating interactive visualizations such as charts, heatmaps, and dashboards.
4	To perform data preprocessing, transformation, and cleaning using appropriate tools and techniques.
5	To enable learners to analyze real-world datasets and derive meaningful business insights.

Instruction: Part A programs may be executed in either R or Python.

Experiment No.	List of Experiments
	PART A
1	a) Develop a program to accept a sentence as input and determine the number of words, digits, uppercase letters, and lowercase letters present in the given sentence. b) Develop a program to demonstrate data subsetting operations for an Employee DataFrame containing Employee Name, Salary, and Department. Also compute the total salary and average salary for each employee based on the given data.
2	Develop a program to create a Students DataFrame using vectors such as Student ID (Sid) and Student Name (S-name), and perform the following operations: a) Insert multiple student records into the DataFrame. b) Add a new column for student address. c) Remove empty row from the DataFrame. d) Merge two DataFrames containing student details based on Student ID.
3	Develop a Program for Data Manipulation that Performs the Following Operations. a. Read multiple datasets from different files or data sources. b. Merge or join the datasets using common variables or keys. c. Perform aggregation operations such as calculating sums, averages, and counts based on groups or categories. d. Filter the dataset according to specific conditions or criteria. e. Create new variables or transform existing variables using functions or mathematical operations

4	Develop program to create and display a Bar Chart, Horizontal Bar Chart, and Pie Chart using the number of books written by different authors as a vector. Analyze the output generated from the charts.
5	Develop Program to Perform the Following Operations a. Create a vector of multiples of 5 using the sequence function from 5 to 60 and display the values. Also, create a filtered vector containing values greater than 15 and less than 45. b. Create a list containing a vector, a matrix, and another nested list. Display the class of each element in the list. Determine the number of objects in the list and find the length of the first vector in the list. c. Create matrices and perform matrix addition, subtraction, and multiplication of a matrix with its transpose. d. Calculate the column sums, row means, and total sum of a matrix. Also, sort the matrix elements column-wise in ascending order.
6	Develop program to perform Hierarchical Clustering and generate Dendrograms to visualize the clustering structure of the given dataset. a) Create a Hierarchical Dendrogram and analyze the clustering pattern obtained. b) Create a Cluster-based Dendrogram by applying different linkage methods and write the analysis of the results.
7	Develop a program using ggplot2 and the built-in mtcars dataset to create a Correlation HeatMap showing the relationship between variables such as mileage, horsepower, weight, and engine size. Analyze the correlated heatmap plot obtained.
8	A city pollution monitoring system wants to analyze air quality data to identify relationships between variables. Develop a program using ggplot2 to create an Interactive HeatMap using an air quality dataset, and write the analysis for the generated heatmap plot.
	PART B
1	Using Tableau, create an interactive dashboard for GlobalMart Retail to analyze sales performance based on Region, Category, and Customer Type using the given dataset containing Order ID, Order Date, Sales, and Profit. Include charts for sales trends, profit analysis, and customer-wise sales analysis.
2	Using Tableau, design an interactive dashboard for FreshBasket Superstores to analyze sales performance across different regions and product categories. The dashboard should include filters, calculated fields, and parameters to enable user interaction and provide insights into overall business performance.
3	Using Tableau and the built-in iris dataset, create an interactive Heatmap visualization to analyze the relationship among the variables such as sepal length, sepal width, petal length, and petal width. Apply appropriate filters and formatting to make the heatmap interactive and interpret the patterns observed in the dataset.
4	Design an interactive sales analysis report using the given retail dataset in Microsoft Power BI. The report should include a column chart for category-wise total sales, a pie chart showing region-wise order distribution, and a table visual displaying order details with a calculated total column. Add a region slicer to filter all visuals dynamically and ensure proper layout, formatting, and cross-filtering between visuals on the report page.
5	Design a data transformation workflow using Microsoft Power BI for the employee dataset using Power Query Editor. The dataset contains EmployeeID, FirstName, LastName, JoiningDate (text format dd/mm/yyyy), Salary (text with

	\$ symbols), Department, and Status (with null or blank values). Perform the following operations: load the dataset into Power Query Editor, convert JoiningDate to Date type, clean and convert Salary to Decimal Number after removing special characters, replace null/blank values in Status with “Active”, and create a custom column Full Name by combining FirstName and LastName. Finally, load the cleaned data and verify the transformations in Data view.
--	---

Note: Students should execute one program from Part A and one program from Part B in examination.

TEXT BOOKS		
1	Stephen Few	Information dash board design, Second Edition, Analytics Press, 2021

REFERENCES	
1	https://www.tableau.com/big-book-dashboards
2	http://book.flowingdata.com/
3	https://www.w3schools.com/python/matplotlib_intro.asp
4	https://www.geeksforgeeks.org/r-language/types-of-data-visualization

Course Outcomes:	
Upon completion of this course the student will be able to:	
CO1	Develop R/Python programs to perform data analysis, statistical operations, and visualizations.
CO2	Design interactive dashboards using Tableau for business data analysis.
CO3	Apply clustering, heatmaps, and correlation analysis techniques to interpret real-time datasets.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3	2	-	-	-	-	-	-	-	-	-	-	-	3
	CO2	3	2	-	-	2	-	-	-	-	-	-	-	-	3
	CO3	3	2	-	-	2	-	-	-	-	-	-	-	-	3
AVG		3	2	-	-	2	-	-	-	-	-	-	-	-	3

1: Low association 2: Moderate association 3: High association

TypeScript

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P3ISA2	SEE Marks:	50

Course objectives:

This course will enable students to:

1	Apply TypeScript data types and structures.
2	Develop and manage applications using advanced TypeScript features.
3	Integrate TypeScript with modern development environments and frameworks.
4	Implement best practices in TypeScript development.

Experiment No.	List of Experiments
1	A software company named TechSolutions Pvt. Ltd. wants to automate its employee salary processing system. Currently, the HR department calculates employee salaries manually, which is time-consuming and may lead to calculation errors. To improve efficiency, the company has decided to develop an Employee Payroll Management System using TypeScript and object-oriented programming concepts. The system should maintain employee details such as: - Employee ID - Employee Name - Basic Salary In addition to the basic salary, the company provides the following allowances to employees: HRA (House Rent Allowance) – 20% of the basic salary DA (Dearness Allowance) – 10% of the basic salary The payroll system should automatically calculate the employee's total salary by adding these allowances to the basic salary. The program should: - Create a class named Employee to represent employee details. - Use a constructor to initialize employee information such as employee ID, employee name, and basic salary. - Implement a method to calculate the total salary including HRA and DA. - Display employee details along with salary breakdown and total salary. - Create multiple employee objects to demonstrate the working of the payroll system.
2	A company named SmartTech Solutions has developed an online employee portal where employees can securely log in to access their profile, attendance records,

	salary details, and company announcements. To ensure system security, the portal must verify user login credentials before granting access. The company wants to develop a Login Authentication System using TypeScript and interfaces to validate users based on their username and password. The system should compare the entered login credentials with predefined authorized credentials stored in the system. The program should: 1. Create an interface to define the structure of user login information. 2. Store user credentials such as: ○ Username ○ Password 3. Implement a class that uses the interface and performs user authentication. 4. Verify whether the entered username and password match the stored login credentials. 5. Display appropriate messages such as: ○ “Login Successful” when valid credentials are entered ○ “Invalid Username or Password” when incorrect credentials are provided 6. Allow testing with multiple users or login attempts to demonstrate the authentication process.
3	An online shopping company named EasyMart wants to develop a digital shopping cart system for its e-commerce website. Customers visiting the website should be able to add multiple products to their shopping cart, view the selected items, and calculate the total bill amount before placing an order. Currently, the company is managing cart operations manually during testing, which is inefficient and difficult to maintain. To automate this process, the company has decided to develop a Shopping Cart System using TypeScript and object-oriented programming concepts. The system should allow customers to: • Add products to the shopping cart • View product details • Calculate the total cost of all selected items Each product should contain the following details: • Product Name • Product Price The program should: 1. Create a class named Product to store product information such as: ○ Product Name ○ Product Price 2. Create another class named ShoppingCart to manage shopping cart

	operations. 3. Implement methods to: ○ Add products to the shopping cart ○ Display all product details in the cart ○ Calculate the total bill amount of purchased items 4. Store multiple products inside the shopping cart using arrays or collections. 5. Display the final shopping cart summary along with the total cost of all products added to the cart. 6. Demonstrate the system by adding multiple products to the cart and generating the final bill.
4	A software company named DataConnect Solutions is developing a web application that communicates with multiple online services such as user management systems, payment gateways, product databases, and reporting services through APIs. Each API returns different types of data such as text messages, numerical values, user objects, or lists of records. The development team wants to create a reusable and flexible system to handle all API responses efficiently without writing separate code for each data type. To achieve this, the company has decided to use Generics in TypeScript for designing a common API response handling mechanism. The program should: 1. Create a generic class named <code>ApiResponse<T></code> where T represents the type of data returned by the API. 2. Store the following API response details: ○ Status Code ○ Message ○ Response Data 3. Use constructors to initialize API response values dynamically. 4. Implement methods to: ○ Display API response details ○ Return API response data ○ Handle success and error messages based on status codes 5. Demonstrate the flexibility of generics by creating multiple API response objects using different data types such as: ○ String responses (e.g., "Login Successful") ○ Number responses (e.g., total sales count) ○ Object responses (e.g., employee or customer details) ○ Array responses (e.g., list of products or users) 6. Display all response details using console output.
5	A company named SecureTech Solutions is developing an internal management system for its organization. The system will be used by different types of users such as administrators, managers, employees, and guests. Since each user should have different levels of access to company resources, the organization wants to implement a Role-Based Access Control (RBAC) System. For example:

	• Administrators should have complete access to the system. • Managers should be able to manage employees and view reports. • Employees should only access their assigned tasks and information. • Guests should have limited view-only access. To improve security and maintain proper authorization, the company has decided to implement the RBAC system using Enums and Classes in TypeScript. The program should: 1. Create an enum named Role to define different user roles such as: ○ Admin ○ Manager ○ Employee ○ Guest 2. Create a class named User to store user details such as: ○ User ID ○ User Name ○ User Role 3. Use constructors to initialize user information and assign roles. 4. Implement methods to: ○ Display user details ○ Check the assigned role ○ Display access permissions based on the role 5. Define different access permissions for each role, such as: ○ Admin → Full system access ○ Manager → Manage employees and reports ○ Employee → Limited access to assigned tasks ○ Guest → View-only access 6. Create multiple user objects with different roles to demonstrate how access permissions vary depending on the assigned role.
6	A software development company named NextGen Technologies manages multiple projects simultaneously. Project managers need an efficient system to assign tasks to team members, track task progress, and monitor completed and pending work. Currently, task tracking is done manually using spreadsheets, which makes it difficult to manage project activities and monitor task status effectively. To improve project management and team productivity, the company wants to develop a Task Management System using TypeScript, interfaces, and object-oriented programming concepts. The system should allow users to: • Create new tasks • Store task details • Update task progress • Display completed and pending tasks Each task should contain the following information:

	• Task ID • Task Title • Task Description • Task Status (Pending, In Progress, Completed) The program should: 1. Create an interface named Task to define the structure of task information such as: ○ Task ID ○ Task Title ○ Task Description ○ Task Status 2. Create a class named TaskManager to manage all task-related operations. 3. Implement methods to: ○ Add new tasks ○ Display all tasks ○ Update the status of a task ○ Display completed tasks and pending tasks separately 4. Store multiple tasks using arrays or collections. 5. Display task information clearly along with the current task status. 6. Demonstrate the working of the system by creating multiple tasks with different statuses.
7	A supermarket chain named FreshMart Stores manages hundreds of products in its warehouse and retail outlets every day. The store manager needs an efficient system to track available stock, update item quantities, and search for products quickly. Currently, inventory records are maintained manually, which often leads to stock mismatches, delayed updates, and difficulty in monitoring available items. To improve inventory tracking and stock management, the company wants to develop an Inventory Management System using TypeScript and object-oriented programming concepts. The system should manage inventory items by storing details such as: • Item ID • Item Name • Quantity Available in Stock The program should: 1. Create a class named Item to store inventory details such as: ○ Item ID ○ Item Name ○ Quantity Available 2. Create another class named InventoryManager to handle inventory-related operations. 3. Implement methods to: ○ Add new items to the inventory ○ Update the quantity of existing items

	○ Display available stock details ○ Search for specific items in the inventory 4. Store multiple inventory items using arrays or collections. 5. Display all inventory information clearly along with stock availability details. 6. Demonstrate the system by adding multiple inventory items and updating stock quantities.
8	A national bank named SecureBank Ltd. wants to develop a simple digital banking system to manage customer accounts and banking transactions efficiently. Customers should be able to deposit money, withdraw funds, and check their current account balance through the system. Currently, some branch operations are handled manually, which can lead to transaction errors and incorrect balance updates. To automate banking operations and improve accuracy, the bank has decided to develop a Banking Transaction System using TypeScript and object-oriented programming concepts. The system should manage customer account details such as: • Account Number • Account Holder Name • Account Balance The application should support basic banking operations including: • Deposit transactions • Withdrawal transactions • Balance inquiry The program should: 1. Create a class named BankAccount to store account-related information such as: ○ Account Number ○ Account Holder Name ○ Account Balance 2. Use constructors to initialize customer account details. 3. Implement methods to: ○ Deposit money into the account ○ Withdraw money from the account ○ Display account details and current balance 4. Validate withdrawal operations to ensure that customers cannot withdraw more money than the available account balance. 5. Automatically update and maintain the account balance after every transaction. 6. Demonstrate the system by performing multiple deposit and withdrawal transactions for different customers.

9	A software company named CloudTech Solutions develops large-scale web and mobile applications for different clients. During application execution, the development team needs a common logging system to record various types of information such as system messages, error reports, transaction details, API responses, user activities, and debugging information. Since the application handles different types of data including text messages, numbers, boolean values, arrays, and objects, the company wants to create a reusable and flexible Logger Utility System using Generics in TypeScript. The goal is to avoid writing separate logging classes for every data type and improve code reusability and maintainability. The system should: 1. Create a generic class named <code>Logger<T></code> where T represents the type of data or message being logged. 2. Use constructors or methods to dynamically accept log messages of different data types. 3. Implement methods to: ○ Add log messages ○ Display log details ○ Store and retrieve logged data 4. Demonstrate the working of the logger utility using multiple data types such as: ○ String messages (e.g., "Server Started") ○ Numeric values (e.g., transaction IDs or response codes) ○ Boolean values (e.g., login status) ○ Arrays (e.g., list of errors or user records) ○ Objects (e.g., employee or customer details) 5. Ensure the logger utility is reusable, flexible, and type-safe using TypeScript generics. 6. Display all logged information clearly using console output.
10	A software company named SmartCalc Technologies is developing a calculator application for students, accountants, and office employees to perform daily arithmetic calculations efficiently. As the application grows, the development team wants the code to be well-organized, reusable, and easy to maintain. Instead of writing all calculator operations in a single file, the company has decided to use modular programming in TypeScript by separating each arithmetic operation into different modules. This approach will help developers manage the code more efficiently and allow future expansion of the application with advanced mathematical features. The calculator application should support the following operations: • Addition • Subtraction • Multiplication • Division

	The program should: 1. Create separate TypeScript modules for different arithmetic operations such as: - o Addition - o Subtraction - o Multiplication - o Division 2. Use the export keyword to make functions or classes accessible from each module. 3. Create a main TypeScript file that uses the import keyword to access and execute functions from different modules. 4. Accept input values and perform calculator operations by displaying the computed results. 5. Handle invalid operations such as division by zero using conditional statements. 6. Demonstrate the working of the calculator by performing multiple arithmetic calculations.
--	---

TEXT BOOKS

1	Adam Freeman	Essential TypeScript 5, Third Edition, Manning, 3rd ,Edition, 2023
2	Josh Goldberg	Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript, O'Reilly Media, Inc. 1st Edition, 2022

LINKS FOR REFERENCE

1	https://www.typescriptlang.org/docs/handbook/intro.html
2	https://www.typescriptlang.org/docs/handbook/typescript-tooling-in-5-minutes.html
3	https://www.w3schools.com/typescript
4	https://www.geeksforgeeks.org/typescript/typescript-tutorial
5	https://github.com/JoshuaKGoldberg/Best-Books-For-TypeScript

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Develop applications using fundamental concepts of TypeScript language.
CO2	Create reusable and modular components using features of TypeScript.
CO3	Develop applications using TypeScript to solve real-world problems.

Course Articulation Matrix

	Pos												PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
Cos	-	-	2	-	-	2	-	-	2	2	-	-	-	3	
	-	-	3	-	-	3	-	-	2	2	-	-	-	3	
	-	-	3	-	-	3	-	-	2	2	-	-	-	3	
AVG		-	3	-	-	3	-	-	2	2	-	-	-	3	

1: Low association 2: Moderate association 3: High association

UNIX and Shell Programming

Contact Hours/ Week:	2	Credits:	01
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P3ISA3	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Describe the architecture of the Unix Operating System and demonstrate various UNIX commands along with their options and arguments.
2.	Apply suitable commands and filters for file processing.
3.	Solve Text processing problems using Regular Expressions.
4.	Develop shell scripts to solve the given problem.

Experiment No.	List of Experiments -Part A
1.	a) Introduction to Unix Operating System and demonstrate the basic general shell commands such as date, cal, who, uname, echo, bc, man, history, exit. • Display the time in GMT(Greenwich Mean Time)/UTC(Coordinated Universal Time) time zone. • Display the calendar of the complete current year with the current date highlighted. • Display list of users logged in to the system. • Display all system information such as the kernel name, hostname, kernel version, operating system. • Display a string on the terminal. b) Illustrate the basic directory commands like pwd, mkdir, cd, rmdir. • Create directories with name dir1, dir2, dir3 in root directory • Create directories d11, d12 under dir1 directory • Create directories d21, d22 under dir2 directory • Create files f11, f12 under dir1 directory • Create files f31, f32, file33 under dir3 directory • Navigate to directory dir3 and create folder d31 and files f31, f32 • Demonstrate navigation from one directory to other using absolute path and relative path
2.	Understand the basic file commands like cat, touch cp, mv, rm, ls, wc, cmp, chmod. • To view multiple files • Find out number of lines, word count, byte and characters count in a file. • Read and write file permission for Owner, and Read-only for the group and other. • Change the group ownership of a file or directory.

	• Compare between two files and report the location of the first mismatch.
3.	Query a data file using filter commands in UNIX such as head, tail, cut, tr, nl, pr, join. • To view the first 20 lines of a file named "example.txt". • To view the last 26 lines of a file named "example1.txt". • Compare the ends of multiple files. • To print the version number of the command. • Combine the contents of these two files. • To convert characters from lower case to upper case.
4.	Search for a regular expression in a file using grep, sed command. • Find the number of lines that matches the given string. • Display the lines that are not matched with the specified search string. • Match the lines which start with the given string. • Match the lines which end with the given string. • Replaces the word "unix" with "linux" in the file. • Replace the first, second occurrence of a pattern in a line.
5.	Learn various features and controls of VI editor like edit commands, navigation commands, and the ex-mode. • Opening an existing file with 'file_name' = jayesh. • Copy lines or words from one place and paste them in another place. • Search some text for the string "Seeksforseeks". • Delete the lines from 1 to 5. • Delete the lines from present cursor position to 3 lines. • Cut selected lines and paste them at a different position in the file. • Search for and replace a specific word throughout the file. • Display line numbers in the VI editor.
6.	List background processes currently running in the system. • Display processes in hierarchical tree format. • Find the process consuming the highest CPU usage. • Find the process consuming the highest memory usage. • Display all system processes, including daemon processes. • Search for a specific process by process name. • Display only the PID and command name of running processes.
List of Experiments -Part B	
1.	Understand shell programming and the conditional statements in it. • Conditional statement to check whether two numbers are equal or not equal. • Conditional statement to check whether the given number is even or odd. • Count the number of users logged in.
2.	Application of multi-way branching in shell programming. • Write a shell script using multi-way branching (case statement) to display the headquarters address of a given car company based on the car name entered by the user.

	Design a simple calculator to perform basic arithmetic operations.
3.	Illustrate string comparison and file handling in shell programming - Demonstration string comparison in shell programming. - Write a shell script to copy contents from one file to another. - Compare whether the strings are equal or not.
4.	Illustrate the different test commands in shell programming. - Test the validity of a command. - Check the type of file and the permissions related to a file.
5.	Illustrate the different loops in shell programming. - Read a file with a for loop. - Read only a particular part of a file using while loop.
6.	a) Write a shell script to add two numbers and debug the execution using bash -x and set -x. b) Demonstrate syntax checking in shell scripting using bash -n by creating a script with syntax errors.

TEXT BOOKS

1.	Sumitabha Das	Unix- Concepts and Applications, 4 th Edition, Tata McGraw-Hill, 2017.
----	---------------	---

Course Outcomes:

Upon completion of this course the student will be able to:

CO1:	Analyze the fundamental components and functionalities of the Unix operating system using various unix commands.
CO2:	Design solutions for Text processing problems using Regular Expressions .
CO3:	Develop shell scripts to perform text editing and to automate system-level tasks.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	1													1
	CO2	1	2	2											1
	CO3	1	2												1
AVG		1	2	2											1

1: Low association 2: Moderate association 3: High association

Prompt Engineering

Contact Hours/ Week:	2	Credits:	
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P3ISA4	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Understand LLM Architectures
2.	Discuss various prompting techniques
3.	Develop structured prompts for specific topics
4.	Utilize prompt chaining and agentic workflows to solve complex problems
5.	Illustrate the various evaluation metrics

	Lab Exercises
Lab 1	Develop a set of common prompts and use ChatGPT, Claude, and Gemini to generate responses. Compare the outputs based on correctness, completeness, response time, creativity, and token usage. Prepare a comparative report highlighting the strengths and limitations of each model.
Lab 2	Investigate the influence of inference parameters such as temperature, top-p, maximum tokens, and context window on the responses generated by an LLM. Analyze how variations in these parameters affect response quality, consistency, factual accuracy, and creativity for different tasks.
Lab 3	Design structured prompts by incorporating role, context, instruction, output format, and constraints for tasks such as essay writing, email generation, code explanation, or report generation. Compare the outputs with those obtained using unstructured prompts and evaluate improvements in quality and consistency.
Lab 4	Implement Zero-shot, Few-shot, and Role Prompting techniques to perform a text classification or document summarization task. Compare the generated outputs using evaluation metrics such as accuracy, coherence, relevance, and response style.
Lab 5	Develop Chain-of-Thought prompts to solve mathematical, logical, and analytical reasoning problems. Compare the solutions obtained with direct-answer prompting and analyze differences in reasoning quality, correctness, and interpretability.
Lab 6	Use an AI coding assistant such as ChatGPT, Claude, or GitHub Copilot to generate programs in Python, C, or Java for a given problem. Evaluate the generated code by performing debugging, explanation, optimization, documentation generation, and refactoring, and compare it with manually written code.
Lab 7	Design prompts that generate outputs in structured formats such as JSON, XML, Markdown, CSV, and tables. Validate the generated outputs for correctness and demonstrate their integration into a simple software application or parser.
Lab 8	Design and test adversarial prompting techniques to identify vulnerabilities in Large Language Models (LLMs), and refine prompts to improve robustness, safety, and resistance against malicious or misleading inputs.
Lab 9	Evaluate LLM responses on factual, ethical, and socially sensitive prompts to identify hallucinations, bias, misinformation, and unsafe content. Design responsible prompting strategies that reduce these issues and improve trustworthy AI behavior.
Lab 10	Analyze hallucinations, bias, and jailbreak vulnerabilities in LLM responses, and design responsible prompting strategies to improve safety and ethical AI usage.

Lab 11	Develop a context-aware question-answering system by providing external documents or knowledge sources to an LLM. Compare the generated responses with and without contextual information, and evaluate improvements in factual accuracy and reduction of hallucinations.
Lab 12	Develop a simple AI-assisted coding workflow capable of generating source code, documentation, test cases, debugging suggestions, and code explanations for programming problems. Evaluate the system using AI assurance metrics such as correctness, robustness, explainability, safety, usability, and reliability.

TEXT BOOKS:

1.	Nathan Hunter	The Art of Prompt Engineering with ChatGPT, Apress, 2023 · ISBN 978-1-4842-9962-3
2.	James Phoenix & Mike Taylor	Prompt Engineering for Generative AI, O'Reilly, 2024 · ISBN 978-1-0981-5343-5

REFERENCE BOOKS:

1.	Valentina Alto	Building LLM-Powered Applications, Packt, 2023 · ISBN 978-1-8350-8369-0
2.	Suhas Pai	Designing Large Language Model Applications, O'Reilly, 2024
3.	Web Resource	docs.anthropic.com/en/docs/build-with-claude/prompt-engineering Anthropic Prompt Engineering Guide
4.	Web Resource	platform.openai.com/docs/guides/prompt-engineering OpenAI Prompt Engineering

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Discuss how LLMs process and respond to prompts to perform coding and content generation tasks.
CO2	Design different prompt engineering techniques to generate accurate and relevant AI responses.
CO3	Apply AI tools to generate and debug programs in Python, C, and Java.
CO4	Evaluate AI-generated outputs using quality measures and structured prompts.
CO5	Analyze AI systems for hallucinations, bias, safety, and security issues using responsible AI practices.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
Cos	CO1	3		2		3									3
	CO2	3		2		3									3
	CO3	3		2		3									3
	CO4	3		2		3									3
	CO5	3		2		3			3	3					3
AVG		3		2		3			3	3					3

Foundations of Engineering Mathematics-I

(For III Semester Lateral Entry students only)

Contact Hours/ Week:	3:0:0:0	Credits:	00
Total Lecture Hours:	40 Hours	CIE Marks:	100
Course Code:	P3MATDIP	SEE Marks:	00

Course objectives:

This course will enable students to:

1.	Familiarize the importance of differential calculus.
2.	Familiarize the importance of calculus associated with one variable and multivariable arising in engineering.
3.	Analyze and solve engineering problems by applying Ordinary Differential Equations.

Unit I: Differential Calculus-I

Polar curves: angle between the radius vector and tangent, angle of intersection of polar curves. Pedal equation for polar curves. Curvature and radius of curvature in pedal form. (Without proof)

(8 Hours)

Unit II: Partial differentiation

Partial derivatives, Total derivatives, differentiation of composite functions. Jacobians.

(8 Hours)

Unit III: Series Expansion and Application

Taylor's and Maclaurin's series expansions for one variable **(statement only)**. Indeterminate forms L' Hospital Rule (without proof)
(Only $\frac{0}{0}, \frac{\infty}{\infty}, 0 \times \infty$ and $\infty - \infty$ form)

(8 Hours)

Unit IV: Differential Equations of I Order

Solution of first order and first degree differential equations: Variable Separable, Linear differential equations and Exact differential equations

(8 Hours)

Unit V: Differential Equations of higher order

Solution of second and higher order equations with constant coefficients by inverse differential operator method, method of variation of parameters.

(8 Hours)

TEXT BOOKS

1	Erwin Kreyzig	Advanced Engineering Mathematics", 10 th edition, Wiley Publications, 2016. ISBN: 9788126508273.
2	B. S. Grewal	"Higher Engineering Mathematics", 43 rd edition, Khanna Publications, 2015. ISBN:9788174091955.

REFERENCE BOOKS		
1	Maurice D. Weir, Joel R. Hass and George B. Thomas	“Thomas’ calculus: Early Transcendental”, 12 th edition, Pearson Education, 2016. ISBN: 978078024269.
2	Ramana .B.V	Higher Engineering Mathematics, McGraw-Hill Education, 11th Ed., 2017, ISBN: 9780070634190.
3	C.Ray Wylie, Louis. C. Barrett	Advanced Engineering Mathematics, McGraw – Hill Book Co., New York, 6th Ed., 2017, 9780070722064.
4	Louis A. Pipes and Lawrence R. Harvill	“Applied Mathematics for Engineers and Physicists”, 3 rd edition, McGraw Hill, 2014. ISBN: 9780486779515.
5	Peter V.O ’Neil	“Advanced Engineering Mathematics”, 7 th edition, CENGAGE Learning India Pvt. Ltd. Publishers, 2012. ISBN; 9788131503102.

**SCHEME and SYLLABUS
OF
IV SEMESTER B.E.**

**INFORMATION SCIENCE AND ENGINEERING
FOR THE A.Y: 2026-27**

SCHEME OF TEACHING AND EXAMINATION : IV Semester

Sl. No.	Course and Course Code		Course Title	Teaching / Paper setting Dept.	Teaching and Learning scheme					Examination				Credits
					Lecture	Tutorial	Practical/ Drawing	TW&SL	Total Hours / Sem.	Duration in hrs.	CIE Marks	SEE Marks	Total Marks	
					L	T	P	SL						
1.	ASC/ PCC	P4MAT1	Discrete Mathematics	Maths / Specific department	42	28	0	48	120	3	50	50	100	4
2.	IPCC	P4ISI1	Artificial Intelligence and Machine Learning	ISE	42	0	28	50	120	3	50	50	100	4
3.	PCC	P4IS1	Data Communication	ISE	42	0	0	48	90	3	50	50	100	3
4.	PCC	P4IS2	Design and Analysis of Algorithms	ISE	42	0	0	48	90	3	50	50	100	3
5.	PCCL	P4ISL1	Design and Analysis of Algorithms Laboratory	ISE	0	0	28	2	30	2	50	50	100	1
6.	AEC	P4ISAX	Ability Enhancement Course Laboratory	ISE	0	0	28	2	30	2	50	50	100	1
7.	BSC	P4ISB1	Bioinformatics for Engineers	ISE	15	0	0	15	30	1	50	50	100	1
8.	PCC	P4IS3	Agentic Artificial Intelligence	ISE	0	0	28	2	30	1	50	50	100	1
9.	SDC	SDCIS4	Environmental Science Project	ISE	0	0	0	30	30	2	50	50	100	1
10.	NCMC	PMC1	National Service Scheme (NSS)	NSS coordinator	0	0	0	28	28	--	100	--	100	PP
		PMC2	Physical Education (PE) (Sports and Athletics)	Ph.Edn. Director										
		PMC3	Yoga (Yoga Teacher)											
		PMC4	Music (Music Teacher)											
			Total							550	450	1000	19	
		AAP	AICTE Activity Points (Regular students have to earn 100 activity points from I to VI semesters and Lateral Entry students have to earn 75 activity points from III to VI semesters)	Compulsory requirement for the award of a degree										
11.	NCMC	P4MATD IP	Mathematics course for Lateral Entry Students	Maths Dept.	14	0	0	14	28	3	100	---	100	PP

Note:

ASC: Applied Science Course **PCC:** Professional Core Course, **IPCC:** Integrated Professional Core Course, **PCCL:** Professional Core Course laboratory, **AEC:** Ability Enhancement Course, **SDC:** Skill Development Course **NCMC:** Non Credit Mandatory Course **L:** Lecture, **T:** Tutorial, **P:** Practical **SL:** Self Learning **TW&SL:** Team works & Self Learning **CIE:** Continuous Internal Evaluation, **SEE:** Semester End Evaluation.

Ability Enhancement Course (Laboratory)			
P4ISA1	Advanced RISC Microcontroller	P4ISA2	Internet of Things
P4ISA3	Mobile Application Development		

Environmental Science Project: The Environmental Science Project is an applied learning component designed to develop students' awareness, understanding, and responsibility toward the environment. It provides an opportunity to study real-world environmental issues and apply scientific and engineering principles to design feasible and sustainable solutions.

The topics under environment include, but not limited to, climate change, biodiversity, air and water pollution, land use, excess use of natural resources, earthquakes, rise in the earth's temperature, power generation, soil erosion, environment issues related program, etc.

The project involves problem identification, field surveys, case studies, data collection, environmental audits, analysis, and proposal of remedial or preventive measures aimed at improving biodiversity, air quality, and thermal comfort, etc. Students can take up project individually or in a group not exceeding 4 students. Students can opt for Interdisciplinary Project based on their interest.

The evaluation shall be done as per the following;

CIE: The CIE marks shall be awarded by a committee consisting of the Head of the concerned Department and two senior faculty members of the Department, one of whom shall be the Guide. The CIE marks awarded for the project work, shall be based on the rubrics.

SEE: SEE will be conducted by the two examiners appointed by the Institute. The SEE marks awarded for the project work shall be based on the rubrics.

Discrete Mathematics (Computer Science and allied branches)

Contact Hours/ Week:	3L+2T	Credits:	04
Total Lecture Hours:	42 Hours	CIE Marks:	50
Total Tutorial Hours:	28	SEE Marks:	50
Course Code	P4MAT1		

Course objectives:

This course will enable students to:

- | | |
|----|--|
| 1. | Develop the sense of reasoning through mathematical logic. |
| 2. | Illustrate the applications of relation along with their properties in engineering. |
| 3. | Develop the deep understanding of algebraic structures through group and its properties. |
| 4. | Expose to ancient the techniques in number theory with a special emphasis on primes and the remainder. |

Module I: Mathematical Logic

Logic, Laws of logic, Logical equivalence using the Laws of Logic, Logical Implications and Rules of Inference, Quantifiers, Definitions, Proofs of theorems.

(8 Hours)

Module II: Relations

Relations, Properties of Relations, Computer Recognition- Zero-One Matrices and Digraphs, Partial order relation -Poset and Hasse-Diagrams, Equivalence Relation and Partitions, Extremal elements of a Poset, Lattice.

(8 Hours)

Module III: Groups

Binary Operations and Properties, Definition of a Group, Examples and Elementary properties, Abelian Groups, Homomorphism, Isomorphism and Cyclic Groups, Cosets and Lagrange's Theorem, Normal subgroups.

(8 Hours)

Module IV: Number Theory – Divisibility Theory in Integers

Introduction, The division algorithm, greatest common divisor, Euclidean Algorithm, The Diophantine equation $ax + by = c$, Fundamental theorem of arithmetic, The Goldbach conjecture.

(8 Hours)

Module V: Theory of congruence

Basic properties of congruence, Binary and decimal representation of integers, Chinese remainder theorem, Fermat's Theorem, Wilson Theorem, The Fermat-Kraitchik Factorization method.

(8 Hours)

TEXT BOOKS		
1	Ralph P. Grimaldi	“Discrete and Combinatorial Mathematics”, 5 th Edition, Pearson Education, 2019. ISBN(13) : 978-9353433055.
2	Bernard Kolman, Robert Busby and Sharon C. Ross	“Discrete Mathematical Structures”, 6 th edition, Pearson Education, 2012. ISBN(13) : 978-0132297516
3	David M Burton	“Elementary Number Theory”, 7 th Edition, McGraw Hill Education, 2013. ISBN(13) : 978-0073383149

REFERENCE BOOKS		
1	Kenneth H. Rosen, and Kamala Krithivasan	“Discrete Mathematics and its Applications”, 8th Edition, McGraw Hill, 2021. ISBN(13) : 978-9390727353.
2	J.P.Tremblay and R. Manohar	“Discrete Mathematical Structures With Applications to Computer Science”, McGraw Hill Higher Education, 1975, ISBN(13) : 978-0070651425.
3	M. Ram Murthy and Jody Esmonde,	“Problems in Algebraic number theory”, 2nd Edition, Springer, 2006. ISBN(13) : 978-0387221823.

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Apply laws of logic to derive logical implications by using quantifiers.
CO2	Demonstrate the representation of relation as zero-one matrix and Hasse diagram.
CO3	Apply the concept of groups and subgroups to verify the properties of algebraic structures.
CO4	Demonstrate the use of Euclidean algorithm and its applications.
CO5	Perform congruence arithmetic with primes and composites.

Course Articulation Matrix

	POs												PSOs	
		1	2	3	4	5	6	7	8	9	10	11	1	2
COs	CO1	3	3											
	CO2	3	3											
	CO3	3	3											
	CO4	3	3											
	CO5	3	3											

Artificial Intelligence and Machine Learning (IPCC)

Contact Hours/ Week:	3L+2P	Credits:	4
Total Lecture Hours:	42	CIE Marks:	50
Total Practical Hours:	28	SEE Marks:	50
Sub. Code:	P4ISI1		

Course objectives:

This course will enable students to:

1.	Understand fundamental concepts in Artificial Intelligence.
2.	Explore basic principles of AI such as problem solving, searching, knowledge representation and learning.
3.	Explain the role of agents in AI and structure of the agents.
4.	Analyze and identify significant characteristics of data sets for measurements.
5.	Understand and implement the training, testing, and validation phases of supervised and unsupervised learning algorithms.

UNIT I

Introduction: What is AI? Acting humanly: The Turing Test approach, Thinking humanly: The cognitive modelling approach, Thinking rationally: The “laws of thought” approach, Acting rationally: The rational agent approach.

Intelligent Agents: Agents and Environments, Rationality, Omniscience, learning, and autonomy, The nature of Environments: Specifying the task environment, Properties of task environments, The Structure of Agents; Agent programs, Simple reflex agents, Model-based reflex agents, Goal-based agents, Utility-based agents, Learning agents.

Solving Problems By Searching: Problem-solving agents; Well-defined problems and solutions, Formulating problems, Example problems; Toy problems, Real-world problems.

9 Hours

UNIT II

Solving Problems By Searching (Contd.): Searching for solution; Infrastructure for search algorithms, Measuring problem-solving performance, Uninformed search strategies, Uniform-cost search, Depth-limited search.

Adversarial Search: Games, Optimal Decisions in Games; The minimax algorithm.

9 Hours

UNIT III

Introduction: If Data had Mass, The Earth Would Be A Black Hole, Learning; Machine Learning, Types of Machine Learning, Supervised Learning; Regression, Classification, The Machine Learning Process.

Preliminaries: Some Terminology; Weight Space, The Curse of Dimensionality, Knowing What You Know; Overfitting, Training, Testing, and Validation Sets, The Confusion Matrix, Accuracy Metrics, The Receiver Operator Characteristic (ROC) Curve, Unbalanced Datasets, Measurement Precision, Testing Machine Learning Algorithms, Turning Data into Probabilities; Some Basic Statistics.

8 Hours

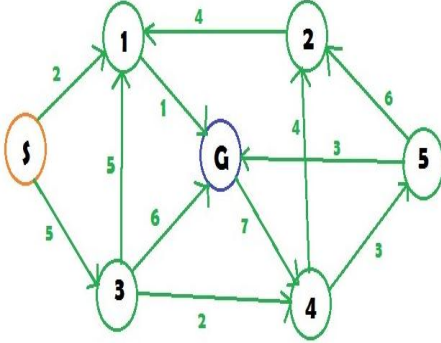
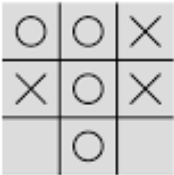
UNIT IV	
Dimensionality Reduction: Linear Discriminant Analysis (LDA), Principal Components Analysis (PCA),	
LEARNING WITH TREES: Using Decision Trees, Constructing Decision Trees, Classification and Regression Trees (CART);	
8 Hours	

UNIT V	
Probabilistic Learning: K-Nearest Neighbour Algorithm	
Unsupervised Learning-Clustering: Introduction, Hierarchical Clustering, Agglomerative Clustering, The single Linkage Algorithm, The complete linkage Algorithm, Partitional Clustering, The k-means Algorithm,	
8 Hours	

TEXT BOOKS		
1	Stuart J. Russell and Peter Norvig	Artificial Intelligence, A Modern Approach, 3 rd Edition, Pearson India Education Services, 2015 (UNIT I and II).
2	Stephen Marsland	Machine Learning, An Algorithmic Perspective, 2 nd Edition, CRC Press, 2015 (UNIT III, IV and V).
3	Christopher Bishop	Pattern Recognition and Machine Learning, CBS Publishers & Distributors- New Delhi.2006.

REFERENCE BOOKS		
1	Elaine Rich, Kevin Knight:	Artificial Intelligence, 3 rd Edition, Tata McGraw Hill, 2009.
2	Tom M Mitchell	Machine Learning, McGraw-Hill, Inc. New York, NY, USA. 2013.

Experiment No.	List of Experiments
1.	Write two Python programs to calculate the factorial of a given number n using following approaches: a) Iterative function b) Recursive function
2.	Write a Python function that checks whether a given integer is a prime number or not.
3.	Write a function that takes two lists as input, merges them, and returns a new list with unique elements only, maintaining the original order of first occurrence.
4.	Write a Python Program to perform the following searching functions: a. Linear Search b. Binary Search
5.	Write three separate Python functions to perform the following operations on 2D matrices: a) Addition b) Subtraction c) Transpose

6	Write a Python functions using Pandas to implement the following operations: - Create a DataFrame from a dictionary containing student names and their marks. - Add a new column for grades (e.g., A/B/C based on marks). - Filter out students with marks below 50. - Sort the DataFrame by marks in descending order.
7	Write a python program to implement the following Uninformed search algorithms: - Breadth First Search algorithm - Depth First Search Algorithm
8	You are given a road network where each road segment between two locations has an associated cost (e.g., time taken, distance travelled, or fuel consumption). You need to implement the Uniform Cost Search (UCS) algorithm to find the least-cost path from a starting location to a destination. The road network is represented as a graph where: - Nodes represent intersections or locations. - Edges represent the road segments connecting those locations, and each edge has a cost associated with it (e.g., travel time or distance). 
9	Write a python program to implement the Tic-Tac-Toe game and demonstrate the steps. Consider 3X3 board and minimum of two players. 
10	Demonstrate the implementation of the supervised probability based naïve Bayesian classifier and demonstrate the performance of the model with clear interpretation of confusion matrices and other performance metrics. Use IRIS dataset for demonstration of the Model performance.
11	Implement a Simple supervised Linear Regression model using the training data set. Assume the appropriate dependent and independent variables for regression analysis. Predict the scores on the test data. Print the MSE and R Squared Score (Coefficient of determination). Interpret the results. Consider Diabetes dataset downloaded from sklearn repository.
12	Implement a supervised Logistic Regression model using the training data set. Assume the appropriate dependent and independent variables for regression analysis. Predict the scores on the test data. The output should include Confusion Matrix, Accuracy, Error rate, Interpret the results. Consider an USER-DATA dataset downloaded from Kaggle repository.

13	Demonstrate the implementation of Linear discriminant analysis (LDA) technique for dimensionality reduction. Consider IRIS dataset from Kaggle repository.
14	Demonstrate the implementation of Principal component analysis (PCA) technique for dimensionality reduction. Consider IRIS dataset from Kaggle repository.
15	A telecommunications company is looking to predict customer churn (whether a customer will leave the service or not). Currently, the company has a huge dataset containing customer details, including their usage patterns, subscription plans, and support tickets. The company is struggling to identify the key factors contributing to churn, and therefore, struggles to take proactive measures to retain customers. Use the ID3 (Iterative Dichotomiser 3) decision tree algorithm to predict whether a customer will churn or stay, based on features such as: • Customer Age • Subscription Plan • Monthly Spend • Number of Support Tickets • Customer Satisfaction Score Demonstrate the performance of the model. Interpret the significance of Entropy and Information gain on model performance. Plot the complete tree constructed.

Course Outcomes:

Upon completion of this course the student will be able to:

CO1.	Apply knowledge representation techniques and problem solving strategies to Artificial Intelligence (AI) applications.
CO2.	Illustrate the structure of the agents and different types of agents used in AI.
CO3.	Apply fundamental concepts of machine learning to solve real time problems.
CO4.	Illustrate the significance of dimensionality reduction techniques for supervised and unsupervised learning.
CO5.	Design applications to solve real world problems using machine learning algorithms such as classification, regression, and clustering.

Mapping of Course Outcomes (COs) to Program Outcomes (POs) & Program Specific Outcomes (PSOs)

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	2													2
	CO2	2	2												2
	CO3			2		3				2	3				2
	CO4			2	2										2
	CO5	2													2
AVG		2	2	2	2	3				2	3				2

1: Low association 2: Moderate association 3: High association

Data Communication

Contact Hours/ Week:	3	Credits:	3
Total Lecture Hours:	42	CIE Marks:	50
Course Code:	P4IS1	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Understand the basic concepts of data communications.
2.	Analyse and comprehend the layered architecture of TCP/IP protocol and OSI model.
3.	Understand physical layer characteristics and usage in communication networks.
4.	Learn the importance of data link layer and services offered for reliable communication in the Internet.
5.	Explore the data link control and MAC protocols in real world scenario.

UNIT I	
Data Communications and Network Models Data Communications: Components, Data Representation, Data Flow. Networks: Network Criteria, Physical Structures. Network types: LAN, WAN, Internet. Network Models Protocol Layering: Scenarios, Principles of Protocol Layering, Logical Connections. TCP/IP Protocol Suite: Layered Architecture, Layers In the TCP/IP Protocol Suite, Description of Each Layer, Encapsulation and Decapsulation, Addressing, Multiplexing and Demultiplexing. The OSI model: OSI versus TCP/IP, Lack of OSI model's success.	
8 Hours	
UNIT II	
Introduction to physical layer Data and Signals: Analog and Digital Data, Analog and Digital Signals, Periodic and non-periodic. Periodic Analog signals: Sign wave, phase, wavelength, Time and frequency domains, composite signals, Bandwidth. Digital signals: Bit rate, bit length, Digital signal as a composite analog signal, Transmission of Digital signals. Transmission Impairment: Attenuation, Distortion, Noise. Data rate limits: Noiseless channel, noisy channel. Performance: Bandwidth, throughput, latency, bandwidth-delay product, jitter.	
9 Hours	
UNIT III	
Digital transmission Digital-to-digital conversion: line coding, Line coding schemes. Analog to digital conversion: PCM, DM. Transmission modes: Parallel Transmission, serial transmission. Analog Transmission Digital-to-Analog Conversion: Aspects of Digital to Analog conversion, Amplitude Shift Keying, Frequency Shift keying, Phase Shift keying, Quadrature Amplitude Modulation.	
9 Hours	
UNIT IV	
Introduction to Data link Layer Introduction: Nodes and links, services, two categories of links, Two sub layers. Error Detection & Correction Introduction: Types of Errors, Redundancy, Detection Versus Correction. Cyclic Codes: Cyclic Redundancy Check, Polynomials, Cyclic Code Encoder Using Polynomials.	

Checksum: Concept, other approaches to the checksum

8 Hours**UNIT V****Data link Control (DLC)**

DLC Services: Framing, Flow and Error Control, Connectionless and Connection-Oriented. Data-link layer Protocols: Simple protocol, Stop-and-wait protocol, piggybacking. HDLC: Configuration and Transfer Modes, Framing.

Media Access Control (MAC)

Random Access: Aloha, CSMA, CSMA/CD, CSMA/CA. Controlled Access: Reservation, Polling, Token Passing. Channelization: FDMA, TDMA, CDMA.

8 Hours**TEXT BOOK**

1	Behrouz A. Forouzan	Data Communications and Networking, Tata McGraw-Hill, 5 th Edition, 2013.
---	---------------------	--

REFERENCE BOOKS

1	Alberto Leon Garcia	Communication Networks: Fundamental Concepts & key Architectures, Tata McGraw – Hill, 10 th Edition, 2004.
2	William Stallings	Data and Computer Communication, Pearson Education, 10 th Edition, 2013.

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Identify the various data and signal types with transmission principles, topologies and layered architecture of computer networks.
CO2	Analyze the physical layer characteristics, its performance parameters, transmission impairments and their impact on data communication.
CO3	Apply the different error detection and correction, modulation, and multiplexing techniques to solve problems in data communications.
CO4	Explain different digital and analog transmission techniques for different networking applications.
CO5	Apply medium access control and data link control protocols to design reliable and efficient data communication over shared networks.

Course Articulation Matrix

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3												3	
	CO2		2											3	
	CO3			2										3	
	CO4			2										3	
	CO5													3	
AVG		3	2	2										3	

1: Low association 2: Moderate association 3: High association

Design and Analysis of Algorithms

Contact Hours/Week:	3	Credits:	3.0
Total Lecture Hours:	42	CIE Marks:	50
Total Practical Hours:	-	SEE Marks:	50
Course Code:	P4IS2		

Course objectives:

This course will enable students to:

1. Explain algorithm design and asymptotic complexity analysis fundamentals.
2. Develop algorithm complexity analysis and correctness proof skills.
3. Compare the performances of various sorting and searching algorithms
4. Develop problem-solving abilities using advanced design strategies such as divide-and-conquer, transform-and-conquer, dynamic programming, and greedy methods.
5. Familiarize the computational complexity concepts, space-time optimization strategies, and backtracking techniques for handling NP-Complete problems.

UNIT I

Introduction: Notion of algorithm, Fundamentals of Algorithmic Problem Solving, Fundamentals of the Analysis of Algorithm Efficiency: Analysis frame work, Asymptotic Notations and Basic Efficiency Classes, Mathematical Analysis of Non-recursive and Recursive Algorithms. [Chapters: 1.1, 1.2, 2.1-2.4]

09 Hours

UNIT II

Brute force: Selection Sort and Bubble Sort, Sequential Search and Brute-Force String Matching. Divide and Conquer: Mergesort, Quicksort, Binary Search. [Chapters: 3.1, 3.2, 4.1-4.3]

08 Hours

UNIT III

Decrease and conquer: Insertion Sort, Depth First Search, Breadth First Search, Topological Sorting, Algorithms for Generating Combinatorial Objects.

Transform and conquer: Presorting, Balanced Search Trees: AVL Tree, Heaps and Heapsort. [Chapters: 5.1 – 5.4, 6.1, 6.3 (only AVL Trees), 6.4]

09 Hours

UNIT IV

Dynamic programming: Computing a Binomial Coefficient, Warshall's and Floyd's Algorithms, The Knapsack Problem.

Greedy Technique: Prim's Algorithm, Kruskal's Algorithm, Dijkstra's Algorithm. [Chapters: 8.1, 8.2, 8.4, 9.1-9.3]

08 Hours

UNIT V

Space And Time Tradeoffs: Sorting by Counting, Input Enhancement in String Matching: Horspool's Algorithm.

Limitations Of Algorithm Power: P, NP and NP-Complete Problems. Coping with the Limitations of Algorithm Power: Backtracking: N-Queens, Hamiltonian Circuit Problem, Subset-Sum Problem. [Chapters: 7.1, 7.2, 10.3, 11.1]

08 Hours

Self-Study Component:

The following topics must be studied by the students:

1. Algorithms for Generating Combinatorial Objects.
2. Implementation of types of Sorting by Counting algorithms.

NOTE: Self-study topics are assessed only in CIE-Quizzes/ Assignments

TEXT BOOKS:

1.	Anany Levitin.	Introduction to The Design & Analysis of Algorithms. 2 nd Edition, Pearson Education. 2011.
----	----------------	--

REFERENCE BOOKS:

1.	Ellis Horowitz, Satraj Sahni and Rajasekharan.	Fundamentals of Computer Algorithms, University Press Pvt. Ltd, 2 nd Edition, 2009.
2.	Cormen, Leiserson, Rivest, and Stein	Introduction to Algorithms" (CLRS), 4 th Edition. The MIT Press. Cambridge, Massachusetts London, 2022

Course Outcomes:

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

CO1:	Illustrate fundamental concepts of algorithms and different design paradigms.
CO2:	Apply brute-force and divide-and-conquer techniques to solve problems using recursive and non-recursive algorithms with complexity analysis.
CO3:	Apply decrease and conquer and transform and conquer to find solutions along with complexity analysis.
CO4:	Apply dynamic programming and greedy techniques to solve optimization problems along with complexity analysis.
CO5:	Illustrate the principles of space–time trade-offs and backtracking techniques for solving N-Queens and subset-sum problems.

Mapping of Course Outcomes (COs) to Program Outcomes (POs) & Program Specific Outcomes (PSOs)

		POs											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3	2										2		
	CO2		3	2		2				2			2		
	CO3		3	2		2				2			2		
	CO4		3	2						2			2		
	CO5		3			1				1			2		
AVG		3	3	2		2				2			2		

1: Low association 2: Moderate association 3: High association

Design and Analysis of Algorithms Laboratory

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P4ISL1	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Implement fundamental algorithms such as searching, sorting, and graph algorithms using programming languages.
2.	Analyze time and space complexity of algorithms through practical experimentation.
3.	Apply algorithm design techniques like divide and conquer, greedy, dynamic programming, and backtracking.
4.	Evaluate algorithm performance using different input cases and empirical analysis.
5.	Design efficient solutions using appropriate data structures for real-world problems.

Experiment	Lab Exercises
1	- Given an unsorted array <code>students_roll[]</code> representing student roll numbers and an integer <code>target_roll</code> representing the student roll number to be searched, determine whether the student is present in the list. Return the index if found; otherwise, return -1. Use the linear search algorithm for this problem statement. - A library stores book IDs in ascending order for faster searching. When a student requests a book, the librarian needs to check whether the given book ID is available. Write a program to find the book ID using Binary Search and display its position if found; otherwise, print that it is not available.
2	- A company stores employee salaries in random order in its records. The HR department wants to arrange the salaries in ascending order for easier analysis. Write a program to sort the salaries using Merge Sort. - Given an unsorted array <code>temperature[]</code>, sort the array in ascending order using an efficient sorting algorithm. Use an in-place sorting algorithm like Quick Sort.
3	- A retail store maintains a daily record of product prices that need to be displayed in ascending order for customer convenience. Since the data is updated incrementally (one product at a time), develop a program using the insertion sort algorithm to efficiently sort the list of product prices as new entries are added. - In a software development company, multiple tasks must be completed where some tasks depend on others (e.g., testing can only begin after coding is complete). Represent these task dependencies as a Directed Acyclic Graph (DAG). Develop a program to determine a valid order of execution of tasks using the topological sorting (source removal) method, ensuring that all prerequisite constraints are satisfied.
4	In a social networking platform, users are connected through friendships. Given a user, it is required to identify all other users who can be reached directly or indirectly through these connections. Represent the network as a directed graph and develop a program using Breadth First Search (BFS) to list all users reachable from a given

	starting user. b. In a computer network within an organization, systems are interconnected through communication links. It is important to verify whether all systems are connected to ensure proper communication. Represent the network as a graph and develop a program using Depth First Search (DFS) to check whether the entire network is connected or not.
5	In an e-commerce platform, thousands of customer orders are recorded daily with different order values. For reporting and analytics, it is necessary to arrange these orders in ascending order of their value to identify trends such as minimum and maximum purchases. Develop a program to sort a list of N order values using the Heap Sort algorithm. The program should first transform the given array into a heap (max-heap) and then repeatedly extract the maximum element to produce a sorted list. Ensure that the solution efficiently handles large datasets and demonstrates the concept of the Transform and Conquer technique.
6	A logistics company is preparing a delivery truck that has a maximum weight capacity. The company has a list of packages, where each package has a specific weight and a monetary value. Due to space and safety constraints, each package can either be fully loaded or not loaded at all (packages cannot be split). The goal of the company is to maximize total profit from the selected packages without exceeding the truck's weight capacity. Given: • n packages, each with a weight $w[i]$ and value $v[i]$ • A truck with maximum weight capacity W Develop a program using Dynamic Programming to determine the maximum total value that can be obtained by selecting a subset of packages such that the total weight does not exceed W.
7	A university maintains a system that records course prerequisite relationships. For example, if Course A is a prerequisite for Course B, and Course B is a prerequisite for Course C, then Course A is indirectly a prerequisite for Course C. The university administration wants to know all possible direct and indirect prerequisite relationships between every pair of courses. Given a directed graph represented with an adjacency matrix, where each node represents a course and an edge represents a prerequisite relationship Develop a program using Warshall's Algorithm to compute the transitive closure of the graph, indicating whether a course is reachable (directly or indirectly) from another course.
8	A city's traffic management system maintains a network of locations connected by roads. Each road has a travel cost or distance. The city planners want to determine the shortest travel distance between every pair of locations to optimize navigation and emergency response routes.

	Given: • A weighted graph where vertices represent locations • Edge weights represent distances between locations • Absence of a direct connection represented by infinity (∞) Develop a program using Floyd's Algorithm to compute the shortest paths between all pairs of vertices.
9	A town planning department is designing an electric power distribution network for a newly developed residential area. Each house must be connected directly or indirectly to every other house through power lines. The cost of laying power lines between different houses varies based on distance and terrain. To minimize the overall infrastructure cost, the planners want to connect all houses such that: • Every house is reachable from any other house • The total cost of laying power lines is minimum • No unnecessary cycles are formed Given a connected, undirected, weighted graph where vertices represent houses and edges represent possible power line connections with associated costs Develop a program using Prim's Algorithm (using Greedy Technique) to find the Minimum Cost Spanning Tree (MST) of the graph.
10	A telecommunications company plans to lay fiber-optic cables to connect multiple branch offices across a region. Several possible cable routes exist between offices, each with a different installation cost. The company's objective is to ensure that: • All offices are connected • The total installation cost is minimized • No redundant communication loops are present The routes are considered independently and selected based solely on their cost. Given an undirected, weighted graph where vertices represent branch offices and edges represent possible cable routes with associated costs Develop a program using Kruskal's Algorithm (using Greedy Technique) to determine the Minimum Cost Spanning Tree by selecting edges in increasing order of weight while avoiding cycles.
11	A digital forensics team is developing a text-analysis tool to scan large log files for suspicious keywords or patterns. These log files can be extremely long, and performing character-by-character matching would be inefficient and time-consuming. To improve performance, the team decides to use an efficient string matching technique that preprocesses the search pattern and reduces unnecessary character comparisons, achieving a better time-space trade-off. Given: • A text string representing the contents of a log file

	• A pattern string representing a keyword to be searched Develop a program using Horspool's Algorithm to efficiently find all occurrences of the pattern in the given text. The algorithm should preprocess the pattern to create a shift table, enabling faster pattern shifts during the search phase.
12	A financial planning application is designed to help users analyze their expenses. The user provides a list of individual expense amounts, and the system must determine if there exists one or more combinations of expenses whose total matches a specified budget limit. Since all possible combinations must be explored and invalid partial solutions should be discarded early, a backtracking approach is suitable for solving this problem efficiently. Given: • A set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers • A positive integer d representing the target sum Write a program using Backtracking technique to find all possible subsets of S whose elements sum exactly to d. Example: If $S = \{1, 2, 5, 6, 8\}$ and $d = 9$ The valid subsets are: • $\{1, 2, 6\}$ • $\{1, 8\}$ Expected outcome of your program should be to: • Generate all subsets whose sum equals d • Display each valid subset • Indicate if no such subset exists
13	A chess tournament organizer is designing a puzzle game based on the Queen's movement rules in chess. The objective is to place queens on a chessboard such that no two queens threaten each other. Since placing a queen in one row affects choices in subsequent rows, and invalid placements must be undone, the problem naturally fits a backtracking solution. Given a chessboard of size $N \times N$, write a program using Backtracking technique to place N queens on the board such that: • No two queens share the same row • No two queens share the same column • No two queens lie on the same diagonal The expected outcome of your program should be to: • Print all valid arrangements of queens on the chessboard • Indicate if no solution exists for the given N

TEXT BOOKS:

1.	Anany Levitin.	Introduction to The Design & Analysis of Algorithms. 2 nd Edition, Pearson Education. 2011.
----	----------------	--

REFERENCE BOOKS:

1.	Ellis Horowitz, Satraj Sahni and Rajasekharan.	Fundamentals of Computer Algorithms, University Press Pvt. Ltd, 2 nd Edition, 2009.
2.	Cormen, Leiserson, Rivest, and Stein	Introduction to Algorithms" (CLRS), 4 th Edition. The MIT Press. Cambridge, Massachusetts London, 2022

Course Outcomes:

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

CO1	Implement basic algorithms such as searching and sorting.
CO2	Apply algorithm design techniques such as divide-and-conquer, greedy, and dynamic programming.
CO3	Analyze the time and space complexity of algorithms through practical execution.
CO4	Implement and test graph algorithms, such as traversal and shortest path problems.
CO5	Design and evaluate efficient algorithmic solutions for given problems.

Course Articulation Matrix

		POS											PSOS		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COS	CO1			3		2							3		
	CO2			3		2							3		
	CO3			3		2				2			3		
	CO4			3		2				2			3		
AVG				3		2				2			3		

1: Low association 2: Moderate association 3: High association

Advanced RISC Microcontroller

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P4ISA1	SEE Marks:	50

Course objectives:

This course will enable students to:

1	Explore the Keil μ vision4 tool to develop embedded applications.
2	Develop and test embedded C programs using LPC2148.
3	Conduct the experiments on an LPC2148 evaluation board with external/internal I/O devices.

Experiment No.	List of Experiments
1	Interface the Logic Controller Interface and Write a C program to count the number of switches is ON and display count on LED's
2	Interface the Logic Controller Interface and Write a C program to read 8 switch status and display its complement on LEDs
3	Interface the Logic Controller Interface and Write a C program to perform BCD up/down counter on LEDs.
4	Display the FIRE and HELP on a 7-segment LED interface, with an appropriate delay in between the digits.
5	Display the Hex digits 0 to F on a 7-segment LED interface, with an appropriate delay in between the digits.
6	Interface a Stepper motor and rotate it in clockwise and anticlockwise direction.
7	Interface the LCD and display "Welcome to SIT"
8	Interface a 3x8 keyboard and display the key pressed on an LCD.
9	Interface a DAC and generate the following waveforms: - Triangular - Square
10	Interface a DAC and generate the following waveforms: - Halfwave rectifier - Sin waveform

REFERENCE BOOKS

1	LPC214x User Manual – http://www.keil.com/dd/docs/datashts/philips/(LPC2148, GPIO, Registers, Embedded components selected)	
2	Shibu K V	Introduction to Embedded Systems, 2 nd Edition, Tata McGraw Hill Education Private Limited, 2017.
3	Steve Furber	ARM System on Chip Architecture, 2 nd Edition, Pearson Education Limited, 2000.

Course Outcomes: Upon completion of this course the student will be able to:	
CO1	Develop Embedded C programs on Keil Microvision software.
CO2	Analyze the interface circuits and interface with microcontroller, develop C code to use LPC2148 Microcontroller GPIOs.
CO3	Develop C code to perform the different operations on the interface circuits.
CO4	Develop C code to generate different waveforms using DAC.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	3				3									1
	CO2		3												1
	CO3			3		3									1
	CO4			3		3									1
AVG		3	3	3		3									1

1: Low association 2: Moderate association 3: High association

Internet of Things

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P4ISA2	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Understand the fundamentals of Arduino UNO, ESP32, sensors, actuators, and embedded system interfacing.
2.	Develop IoT-based applications for monitoring environmental and real-time parameters using various sensors.
3.	Implement cloud-based communication and remote monitoring systems using IoT platforms such as Blynk.
4.	Design smart automation systems integrating sensors, displays, actuators, and wireless communication technologies.

Lab Exercises	
1	Develop and implement a Push Button controlled LED system using the Arduino UNO board to demonstrate digital input and output operations. The program should enable turning the LED ON and OFF using a push button switch.
2	Design a system to interface two different actuators with the Arduino UNO board and develop a program to control their operation. Analyze the behavior of the actuators under different input conditions.
3	Develop an ultrasonic distance measurement system using the ESP32 microcontroller and monitor the measured distance values through the Serial Monitor for real-time observation and analysis.
4	Develop a system to Interface the DHT11 temperature and humidity sensor with a microcontroller and develop a program to acquire, display, and evaluate environmental temperature and humidity readings in real time.
5	Develop an automatic lighting control system using an LDR sensor and Arduino UNO, where the LED switches ON automatically when the ambient light intensity falls below a predefined threshold value.
6	Develop a Rain Detection and Indication system using a Rain Sensor, Arduino UNO, and water source to identify rainfall conditions and provide suitable indications through output devices.
7	Develop a system to interface MQ-7 and MQ-135 gas sensors with the Arduino UNO board to measure Carbon Monoxide (CO) and Ammonia (NH ₃) gas concentrations and analyze the application of the system in IoT-based air pollution monitoring and control.
8	Develop a program using ESP32 to display the text "Hello World" on a 1.8-inch TFT Display, demonstrating graphical display interfacing and text rendering techniques.
9	Propose and demonstrate a system to Interface the DHT11 sensor with the ESP32 board and develop an IoT application to upload and monitor temperature and humidity sensor data on the Blynk Cloud platform in real time.
10	Design and implement a cloud-based actuator control system using ESP32 and Blynk Cloud, enabling remote monitoring and control of actuators through IoT technology.

TEXT BOOKS

1	Arshdeep Bahga, Vijay Madiseti	Arshdeep Bahga, Vijay Madiseti, “ Internet of Things- A Hands On Approach”, Universities press, 2014.
---	--------------------------------	---

REFERENCE BOOKS

1	David Hanes	David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, Jerome Henry,"IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things", 1 stEdition, Pearson Education (Cisco Press Indian Reprint). (ISBN: 978-9386873743)
2	Srinivasa K G	Srinivasa K G, “Internet of Things”, CENGAGE Learning India, 2017

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Demonstrate the interfacing and programming of sensors, actuators, and input/output devices using Arduino UNO and ESP32 boards.
CO2	Develop embedded and IoT-based applications for environmental monitoring and automation using different sensors and modules.
CO3	Implement cloud-connected IoT systems for real-time data monitoring and remote actuator control.
CO4	Design and evaluate smart IoT solutions for real-world applications using modern embedded system tools and technologies.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	2				2							2		
	CO2		2	2									2		
	CO3		2		1	2							2		
	CO4			2		2				1			2		

1: Low association 2: Moderate association 3: High association

Mobile Application Development

Contact Hours/ Week:	2	Credits:	1
Total Lecture Hours:	28	CIE Marks:	50
Course Code:	P4ISA3	SEE Marks:	50

Course objectives:

This course will enable students to:

1	Explore mobile application development frameworks for Android.
2	Understand fundamental concepts of Android operating system.
3	Design real time mobile applications to solve the problems on android studio.

Experiment No.	List of Experiments
1	a) Display “Welcome to Android Laboratory”, toast message with the help of direct and indirect method invocation. b) Create an application to design a Visiting Card. The Visiting card should have a company logo at the top right corner. The company name should be displayed in Capital letters, aligned to the center. Information like the name of the employee, job title, phone number
2	Create an android application to Display Explicit and implicit Intents.
3	Develop an Android application using controls like Button, TextView, EditText for designing a calculator having basic functionality like Addition, Subtraction, Multiplication, and Division.
4	Create a SIGN Up activity with Username and Password. Validation of password should happen based on the following rules: • Password should contain uppercase and lowercase letters. • Password should contain letters and numbers. • Password should contain special characters. • Minimum length of the password (the default value is 8).
5	Develop an Android application to read student data from user and display it in the form of toast message using input controls like Radio Button and Checkbox.
6	Develop an application to set an image as wallpaper. On click of a button, the wallpaper image should start to change randomly every 30 seconds.
7	Develop an Android application to demonstrate the use of ScrollView and ListView controls. The application should contain a ListView implemented using an ArrayAdapter with an ArrayList of company objects as the data source. Each item displayed in the ListView must include the following company details: • Company Name • Company Address • Annual Revenue The application should also use a ScrollView to display scrollable content within the user interface.
8	Write a program to create an activity with two buttons START and STOP. On

	pressing of the START button, the activity must start the counter by displaying the numbers from One and the counter must keep on counting until the STOP button is pressed. Display the counter value in a TextView control.
9	Create an activity like a phone dialer with CALL and SAVE buttons. On pressing the CALL button, it must call the phone number and on pressing the SAVE button it must save the number to the phone contacts.
10	Develop an Android application to fetch and display the current temperature and weather condition of a city using a Weather API.
	Open-ended Experiments Students should form a group of two members. Each group must identify and select a problem statement related to mobile application development. Based on the selected problem statement, the students are required to design and develop a mobile application. Each student should individually demonstrate the developed application and clearly explain their individual contribution to the project.

TEXT BOOKS

1	Ian F. Darwin	Android Cookbook, 2nd Edition, Publisher(s): O'Reilly Media, 2017.
2	John Horton	Android Programming for Beginners, Packt Publishing, 2015.

REFERENCES BOOKS

1. <https://developer.android.com/training/basics/firstapp>
2. <https://www.mygreatlearning.com/academy/learn-for-free/courses/android-application-development>
3. <https://www.w3schools.blog/android-tutorial>

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Design applications for Intents, Web based applications and Location based Applications.
CO3	Develop various Android applications related to layouts & rich user interactive interfaces.
CO4	Develop user interface for mobile applications using widgets with event handling.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	-	-	3	-	3	-	-	-	-	-	-	-	3	-
	CO2	3	2	-	-	3	-	-	-	-	-	-	-	3	-
	CO3	-	3	3	-	3	-	-	-	3	-	-	-	3	-
AVG		3	3	3	-	3	-	-	-	3	-	-	-	3	-

1: Low association 2: Moderate association 3: High association

Bioinformatics for Engineers

Contact Hours/ Week:	1	Credits:	1
Total Lecture Hours:	15	CIE Marks:	50
Course Code:	P4ISB1	SEE Marks:	50

Course objectives:

This course will enable students:

1.	Understand the fundamentals, scope, applications, and biological databases of bioinformatics.
2.	Develop the skills to retrieve, interpret, and manage biological data using standard biological databases and file formats.
3.	Apply bioinformatics tools and sequence analysis techniques to solve problems in genomics, proteomics, and biomedical research.

UNIT I

Unit 1: Bioinformatics and Biological databases: Definition, goals, scope, applications, limitations, and emerging trends. Definition of Sequence: Nucleic acid and Protein Sequences. Introduction to sequence databases: Primary, secondary, and specialised databases. Databases and File formats: NCBI, EMBL, DDBJ, UniProt, and Protein Data Bank (PDB).

5 Hours

UNIT II

Unit 2: Sequence Alignment: Evolutionary Basis, Homology versus Similarity, Similarity versus Identity, Global alignment, Local alignment, Dynamic programming Method (For both Local and Global Alignment, i.e., Needleman-Wunsch & Smith-Waterman), Gap Penalties. Scoring Matrices: Nucleotide scoring matrices: Transitions and Transversions, Amino acid scoring matrices: PAM, BLOSUM, Comparison between PAM and BLOSUM,

5 Hours

UNIT III

Unit 3: Sequence Analysis and Applications: Pairwise sequence alignment and Multiple sequence alignment. Alignment applications (PSA and MSA), Molecular phylogenetics, Tree Terminologies, Cladograms and Phylograms, Applications of phylogenetic analysis. Applications of Bioinformatics in the area of Pharmacy, Drug Discovery, OMICS.

5 Hours

TEXT BOOKS:

1. Basant K Tiwary, Bioinformatics and Computational Biology: A Primer for Biologists, Springer Verlag, 2021, 978-981-16-4240-1
2. Jin Xiong Essentials Bioinformatics, Cambridge University Press, 2006, 9789335657325, 3rd Edition
3. Jeremy Ramsden, *Bioinformatics: An Introduction*, 4th Edition, Springer, 2023.

Web links (e-Resources):

1. <https://www.geeksforgeeks.org/biology/bioinformatics-application/>
2. <https://www.ncbi.nlm.nih.gov>
3. <https://blast.ncbi.nlm.nih.gov/Blast.cgi>
4. <https://www.uniprot.org>
5. <https://www.rcsb.org>

6. <https://www.ebi.ac.uk/training>
7. <https://www.cancer.gov/ccg/research/genome-sequencing/tcga>
8. <https://www.ncbi.nlm.nih.gov/sra/>
9. <https://www.ebi.ac.uk/pride/>

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Explain the fundamental concepts, scope, applications, and biological databases used in bioinformatics.
CO2	Retrieve , interpret, and manage biological sequence and structured data using standard biological databases and data formats.
CO3	Apply bioinformatics tools for sequence analysis and interpret their applications in genomics, proteomics, and biomedical research.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
COs	CO1	2		3											2
	CO2	2	2												2
	CO3			3											2
AVG		2	2	3											2

Agentic Artificial Intelligence

Contact Hours/ Week:	2	Credits:	1
Total Practical Hours:	28	CIE Marks:	50
Course Code:	P4IS3	SEE Marks:	50

Course objectives:

This course will enable students to:

1.	Introduce the fundamental concepts of agentic AI, including autonomy, tool use, and LLM-based decision-making.
2.	Develop practical skills in building AI agents using function calling, reasoning loops, planning, and memory.
3.	Enable students to ground agent responses in external data sources through simple retrieval techniques.
4.	Familiarize students with agent frameworks and basic multi-agent collaboration.
5.	Provide hands-on experience in designing, implementing, testing, and demonstrating a small team-based agentic AI project.

	Lab Exercises
Lab 1	Write a Python program that sends a user query to an LLM API and prints the response. Modify it so that queries containing specific keywords ("calculate", "time", "weather") are handled by direct Python logic (eval, datetime, a mock weather function) instead of being passed to the LLM. Test both versions on the same 5 queries and compare outputs.
Lab 2	Define two Python functions, add(a, b) and get_today_date(). Using an LLM's function-calling feature, accept a natural-language query and let the LLM select and invoke the correct function with correct arguments. Print the selected function name, arguments, and result.
Lab 3	Implement an agent (without any framework) that solves a multi-step task — e.g., find the largest of 3 numbers and report if it is even or odd. Print each cycle's Thought, Action, and Observation separately before proceeding to the next step.
Lab 4	Accept a user goal (e.g., "Plan my morning routine"), use an LLM to decompose it into an ordered step list, and execute the steps sequentially, printing each as "Doing: <step>". Persist completed steps to a file so a second run of the program shows prior progress.
Lab 5	Create 3–5 short .txt note files on different topics. Write a program that reads all files, selects the most relevant one for a user's question using keyword matching, and sends only that file's content with the question to the LLM for an answer.
Lab 6	Reimplement the Lab 3 agent using a beginner-level agent framework (e.g., LangChain) for the same task. Record and compare lines of code between the from-scratch and framework versions.
Lab 7	Implement two agents: Agent A generates one short idea on a given topic; Agent B reviews Agent A's idea and returns one line of feedback. Automate the handoff from Agent A's output to Agent B's input.
Lab 8	Extend an existing agent with: (a) rejection of empty/nonsensical input before calling the LLM, and (b) rejection of LLM output that exceeds a word limit or contains a blocklisted word. Test with 5 valid and 5 invalid inputs.
Lab 9	Form a team of 2–3 students and select one real-world student problem — a timetable assistant, an expense note-taker, or a study reminder. Identify and document at least three concepts from prior labs (e.g., tool use, planning, notes-based Q&A, memory) that will be combined in the solution. Prepare a design document specifying: (a) the single end-to-end feature to be implemented, (b) the tools/functions required, (c) the data to be stored and its format, and (d) a flow diagram illustrating user input → agent

Lab 10A	Implement the core end-to-end feature defined in Lab 10A — the single flow from user input to correct agent output, using the tools/functions and data storage specified in the design. Keep scope limited to this one feature only; no additional functionality should be attempted at this stage.
Lab 10B	Test the prototype from Lab 10B against 5 normal inputs and 3 edge-case/invalid inputs (e.g., empty input, unrealistic values, out-of-scope questions). Record actual vs. expected behavior for each, fix at least 2 identified issues, and re-test to confirm the fixes.
Lab 10C	Deliver a live demonstration (5 minutes per team) of the completed agent performing its end-to-end feature, followed by a short-written report covering: which lab concepts were combined, one challenge faced during implementation, and one improvement the team would make with more time.

TEXT BOOKS:

1.	Jayaram Nanduri and Anand Oka	Agentic AI Unleashed: A guide to designing, building, and deploying autonomous AI systems, 2026, ISBN-13 978-9378544392, Kindle Publisher
2.	Dr Manish Gupta	Building Applications with AI Agents: A comprehensive guide to AI agents for beginners and practitioners , 2025, ISBN-13 978-9365894790

REFERENCE BOOKS:

1.	Chip Huyen	AI Engineering: Building Applications with Foundation Model
----	------------	---

Course Outcomes:

Upon completion of this course the student will be able to:

CO1	Differentiate between LLM-based chatbots and autonomous AI agents by implementing and comparing response-generation and action-execution programs.
CO2	Develop AI agents using function calling, structured reasoning, memory, and goal decomposition to perform multi-step task execution and tool selection.
CO3	Design retrieval-augmented AI agents that utilize external knowledge sources, and compare custom-built agents with framework-based implementations.
CO4	Construct multi-agent AI systems with role-based collaboration, communication handoff, and input/output guardrails for reliable task execution.
CO5	Design an end-to-end agentic AI application that integrates multiple agent design concepts to solve a real-world problem.

Course Articulation Matrix

		Pos											PSOs		
		1	2	3	4	5	6	7	8	9	10	11	1	2	3
Cos	CO1	3		2		3									3
	CO2	3		2		3									3
	CO3	3		2		3									3
	CO4	3		2		3									3
	CO5	3		2		3			3	3		2			3
AVG		3		2		3			3	3		2			3

Foundations of Engineering Mathematics-II (For IV Semester Lateral Entry students only)

Contact Hours/ Week:	3:0:0:0	Credits:	00
Total Lecture Hours:	40 Hours	CIE Marks:	100
Course Code:	P4MATDIP	SEE Marks:	00

Course objectives:

This course will enable students to:

1.	Develop the numerical schemes to solve algebraic, transcendental and differential Equations.
2.	Familiarize the importance of Integral calculus and Vector calculus.
3.	Analyze the engineering problems applying Partial Differential Equations

Unit I: Elementary Numerical Methods – I

Solution of polynomial and transcendental equations - Newton-Raphson method (only formula) – Illustrative examples. Finite differences. Interpolation/extrapolation using Newton's forward and backward difference formulae, Newton's divided difference and Lagrange's formulae (All formulae without proof).

(8 Hours)

Unit II: Numerical Integration & Numerical Solution of Differential Equations

Numerical integration: Trapezoidal, Simpson's $1/3^{\text{rd}}$, Simpson's $3/8^{\text{th}}$ rule and Weddle's rule. Numerical solution of ordinary differential equations of first order and first degree - Taylor's series method, Modified Euler's method, Runge-Kutta method of fourth order and Milne's predictor-corrector method.

(8 Hours)

Unit III: Vector Calculus

Vector Differentiation: scalar and vector fields, Gradient, directional derivative, divergence, curl-physical interpretation; solenoidal and irrotational vector fields-illustrative problems.

(8 Hours)

Unit IV: Partial Differential Equations

Formation of PDEs by elimination of arbitrary constants and functions. Solution of non-homogeneous PDE by direct integration. Homogeneous PDEs involving derivatives with respect to one independent variable only.

(8 Hours)

Unit V: Integral Calculus

Multiple integrals –Evaluation of double and triple Integrals, **Beta and Gamma functions:** Definition, relation between Beta and Gamma functions and simple problems.

(8 Hours)

TEXT BOOKS

1	Erwin Kreyzig	Advanced Engineering Mathematics", 10 th edition, Wiley Publications, 2016. ISBN: 9788126508273.
---	---------------	---

2	B. S. Grewal	“Higher Engineering Mathematics”, 43 rd edition, Khanna Publications, 2015. ISBN:9788174091955.
3	M.K. Jain, S.R.K. Iyengar and R.K. Jain	Numerical Methods for Scientific and Engineering Computation, New Age International Publishers, 8 th Ed., 2022. ISBN: 9788183893732

REFERENCE BOOKS

1	Maurice D. Weir, Joel R. Hass and George B. Thomas	“Thomas’ calculus: Early Transcendental”, 12 th edition, Pearson Education, 2016. ISBN: 978078024269.
2	Ramana .B.V	Higher Engineering Mathematics, McGraw-Hill Education, 11 th Ed., 2017, ISBN: 9780070634190.
3	C.Ray Wylie, Louis. C. Barrett	Advanced Engineering Mathematics, McGraw – Hill Book Co., New York, 6 th Ed., 2017, 9780070722064.
4	Louis A. Pipes and Lawrence R. Harvill	“Applied Mathematics for Engineers and Physicists”, 3 rd Edition, McGraw Hill, 2014. ISBN: 9780486779515.
5	Peter V.O 'Neil	“Advanced Engineering Mathematics”, 7 th edition, CENGAGE Learning India Pvt. Ltd. Publishers, 2012. ISBN; 9788131503102.