SNDT Women’s University, Mumbai

Bachelor of Science (Information Technology)

(B. Sc. - I.T.)

Syllabus

as per NEP 2020

w.e.f.

A.Y.: 2024-2025.

1, Nathibai Thackersey Road, Mumbai- 400020

www.sndt.ac.in

Undergraduate Programmes
2023 May
Tentative Template

Terminologies

Abbreviation	**Full-form**	**Remarks**	**Related to Major and Minor Courses**
Major(Core)	Main Discipline		
Major(Elective)	Elective Options		related to the Major Discipline
Minor Stream	Other Disciplines(Inter/Multidisciplinary) not related to the Major	either from the same Faculty or any other faculty	
OEC	Open Elective Courses/Generic		Not Related to The Major and Minor
VSEC	Vocational and Skill Enhancement Courses		
VSC	Vocational Skill Courses		Not Related to The Major and Minor
SEC	Skill Enhancement Courses		Not Related to The Major and Minor
AEC	Ability Enhancement Courses	Communication skills, critical reading, academic writing, etc.	Not Related to the Major and Minor
VEC	Value Education Courses	Understanding India, Environmental science/education, Digital and technological solutions, Health & Wellness, Yoga education, sports, and fitness	Not Related to the Major and Minor

IKS	Indian Knowledge System	I. Generic IKS Course: basic knowledge of the IKS II. Subject Specific IKS Courses: advanced information pertaining to the subject: part of the major credit.	Subject Specific IKS related to Major
OJT	On-Job Training(Internship/Apprenticeship)	Corresponding to the Major Subject	Related to the Major
FP	Field projects	Corresponding to the Major Subject	Related to the Major
CC	Co-curricular Courses	Health and Wellness, Yoga education sports, and fitness, Cultural Activities, NSS/NCC and Fine/Applied/Visual/Performing Arts	Not Related to the Major and Minor
CE	Community Engagement and service		Not Related to the Major and Minor
RP	Research Project	Corresponding to the Major Subject	Related to the Major

Programme Template:

Programme Degree		Bachelor of Science (B.Sc.)
Parenthesis if any (Specialization)		Information Technology (I. T.)
Preamble (Brief Introduction to the programme)		The Bachelor of Science (Information Technology) (BSc-IT) program is designed to prepare students for dynamic and rapidly evolving careers in the field of information technology. This program embodies a commitment to excellence in education, innovation, and the practical application of IT knowledge. With a strong emphasis on both theory and practical skills, the BSc-IT program equips students with the tools and knowledge they need to succeed in the digital age. The mission of the BSc-IT program is to nurture and develop a new generation of IT professionals who are not only well-versed in the latest technological advancements but also possess the critical thinking and problem-solving skills required to meet the ever-changing demands of the IT industry. Through a holistic and hands-on approach, we aim to empower our students to become leaders and innovators in the world of technology. The BSc-IT program is committed to foster a learning environment that empowers students to be leaders and problem solvers in the ever-evolving world of information technology.
Programme Specific Outcomes(PSOs)		After completing this programme, Learner will
	1.	Understand fundamental concepts of information technology, including operating systems, programming languages, data structures, and algorithms.
	2.	Demonstrate a comprehensive understanding of hardware and software components in computer systems.
	3.	Develop software applications and solutions to address real-world problems using appropriate programming languages and tools
	4.	Compare and contrast different software development methodologies and their suitability for various projects.
	5	Apply data analysis techniques to extract meaningful insights from datasets
	6.	Evaluate the security vulnerabilities in IT systems and propose measures to enhance
	7.	Design and develop innovative IT solutions, including web applications, mobile apps, and databases.

Eligibility Criteria for Programme		**B.Sc.IT –First Year:** A Women candidate must have passed the Higher Secondary School Certificate (Std. XII) examination conducted by the different Divisional Boards of the Maharashtra State Board of Secondary and Higher Secondary Education in any stream with 45% marks (40% for candidates belonging to Reserved category). . OR Must have passed an examination of any other recognized Board or Body Recognized as equivalent. Students who have not done mathematics at 12th Std. are needed to take a bridge course in mathematics and pass in university conducted test before semester 1 examination. OR Must have passed any three year Government recognized Diploma programme in Second Class.
Intake (For SNDTWU Departments and Conducted Colleges)		60

- *External Examination does not always mean Theory paper. It may practical examination, Product submission, projects, etc. checked by external examiners.*
- *Internal evaluation should not be written Theory papers like Unit tests.*
  Internal marks will be acquired through practical, small group or individual Projects, activities, presentations, seminars, workshops, products, assignments, application-based work, reports, etc.
- Practical may be part of the main courses along with theory modules instead of having separate courses of practical work.

Structure with Course Titles
(Options related to our area of study to be provided with "OR" for baskets of different types)

SN	Courses	Type of Course	Credits	Marks	Int	Ext
	Semester I					
1.1	C Programming	Major(Core)	4	100	50	50
1.2	Computer Fundamentals and Operating System	Major(Core)	2	50	0	50
1.3	Open Elective Course-I	OEC	4	100	50	50
1.4	Office Automation Tools	VSC	2	50	50	0
1.5	Swayam/Chetana/MOOC	SEC	2	50	50	0
1.6		AEC	2	50	0	50
1.7		IKS	2	50	0	50
1.8		VEC	2	50	50	0
1.9	* Co-curricular Course-I	CC	2	50	50	0
			22	**550**	**300**	**250**
	*** 1.6,1.7,1.8 and 1.9 Co-Curricular Course (Health & Wellness, Yoga education, sports & fitness, Cultural activities, NSS, NCC and Fine/applied/visual/performing arts) will be provide by the University.**					
	Semester II	**Type of Course**	**Credits**	**Marks**	**Int**	**Ext**
2.1	Data Structure & Algorithms	Major(Core)	4	100	50	50
2.2	Computer Organization and Architecture	Major(Core)	2	50	0	50
2.3	Object Oriented Programming using C++	Minor Stream	2	50	0	50
2.4	Open Elective Course-II	OEC	4	100	50	50
2.5	Linux Operating system	VSC	2	50	0	50
2.6	Vedic Mathematics	SEC	2	50	50	0
2.7		AEC	2	50	0	50
2.8		VEC	2	50	0	50
2.9	* Co-curricular Course – II	CC	2	50	50	0
			22	**550**	**250**	**300**
	*** 2.7,2.8 ,2.9 Co-Curricular Course (Health & Wellness, Yoga education, sports & fitness, Cultural activities, NSS, NCC and Fine/applied/visual/performing arts) will be provide by the University**					

Exit with UG Certificate in Information Technology with 10 extra credits (44+10 credits)

SN	Courses	Type of Course	Credits	Marks	Int	Ext
	Semester III					
3.1	Java Programming	Major(Core)	4	100	50	50
3.2	Database Management System	Major(Core)	4	100	50	50
3.3	Web Technology	Minor Stream	4	100	50	50
3.4	Open Elective Course-III	OEC	2	50	0	50
3.5	Data Analytics using Excel	VSC	2	50	50	0
3.6		AEC	2	50	0	50
3.7	Field Projects	FP	2	50	50	0
3.8	* Co-curricular Course	CC	2	50	50	0
			22	**550**	**300**	**250**
SN	**Courses**	**Type of Course**	**Credits**	**Marks**	**Int**	**Ext**
	Semester IV					
4.1	Python Programming	Major(Core)	4	100	50	50
4.2	Advanced Java	Major(Core)	4	100	50	50
4.3	Introduction to Microprocessor 8086	Minor Stream	4	100	50	50
4.4	Open Elective Course-IV	OEC	2	50	0	50
4.5	Analytical Skills using Statistics	SEC	2	50	0	50
4.6		AEC	2	50	0	50
4.7	Green Computing	CEP	2	50	50	0
4.8	* Co-curricular Course	CC	2	50	50	0
			22	**550**	**250**	**300**

Exit with UG Diploma with 10 extra credits (44+10 credits)

OEC (Open Elective Course)	
Open Elective Course-I	Web Technology
Open Elective Course-II	Multimedia System
Open Elective Course-III	Cyber Security
Open Elective Course-IV	Management Information System

SN	Courses	Type of Course	Credits	Marks	Int	Ext
	Semester V					
5.1	Computer Networks	Major(Core)	4	100	50	50
5.2	Advanced Web Development using ASP.net & C#	Major(Core)	4	100	50	50
5.3	Software Engineering	Major (Core)	2	50	0	50
5.4	Elective –I	Major (Elective)	4	100	50	50
5.5	Multimedia and animation	Minor Stream	4	100	50	50
5.6	Data Visualization using Tableau	VSC	2	50	50	0
5.7	Field Project/Internship	FP/CEP	2	50	50	0
			22	**550**	**300**	**250**
	Semester VI					
6.1	R programming	Major(Core)	4	100	50	50
6.2	Mobile Application Development using Android	Major(Core)	4	100	50	50
6.3	Software Testing	Major (Core)	2	50	0	50
6.4	Elective –II	Major (Elective)	4	100	50	50
6.5	Block Chain Technology	Minor Stream	4	100	50	50
6.6	Internship/Apprenticeship	OJT	4	100	50	50
			22	**550**	**250**	**300**

Exit with Degree (3-year)

4-YearDegree with Honors

SN	Courses	Type of Course	Credits	Marks	Int	Ext
	Semester VII					
7H.1	Machine Learning	Major(Core)	4	100	50	50
7H.2	Natural Language Processing	Major(Core)	4	100	50	50
7H.3	UI/UX Design	Major(Core)	4	100	50	50
7H.4	Data Warehousing	Major(Core)	2	50	0	50
7H.5	Elective –III	Major (Elective)	4	100	50	50
7H.6	Research Methodology	Minor Stream(RM)	4	100	50	50
			22	**550**	**250**	**300**
SN	**Courses**	**Type of Course**	**Credits**	**Marks**	**Int**	**Ext**
	Semester VIII					
8H.1	Deep Learning	Major(Core)	4	100	50	50
8H.2	Internet Of Things	Major(Core)	4	100	50	50
8H.3	Cloud Computing	Major(Core)	4	100	50	50
8H.4	Data Mining	Major(Core)	2	50	0	50
8H.5	Elective –IV	Major (Elective)	4	100	50	50
8H.6	OJT	OJT	4	100	50	50
			22	**550**	**250**	**300**

Major Elective	
Open Elective Course-I	Artificial Intelligence
Open Elective Course-II	Management Information System
Open Elective Course-III	Research Methodology
Open Elective Course-IV	Management Information System

4-YearDegreewith Research

SN	Courses	Type of Course	Credits	Marks	Int	Ext
	Semester VII					
7R.1		Major(Core)	4	100	50	50
7R.2		Major(Core)	4	100	50	50
7R.3		Major(Core)	2	50	0	50
7R.4		Major (Elective)	4	100	50	50
7R.5		Minor Stream(RM)	4	100	50	50
7R.6		Research Project	4	100	100	0
			22	**550**	**300**	**250**
	Semester VIII					
8R.1		Major(Core)	4	100	50	50
8R.2		Major(Core)	4	100	50	50
8R.3		Major(Core)	2	50	0	50
8R.4		Major (Elective)	4	100	50	50
8R.5		Research Project	8	100	100	100
			22	**550**	**250**	**300**

Course Syllabus

Semester: I

1.1 Major (Core)

Course Title	**C PROGRAMMING**
Course Credits	**2**
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply logic to create programs in C.
	2. Analyze and understand computer programming language concepts.
	3. Evaluate and interpret the use of pointers, including their declarations, initialization, and operations.
	4. Design and develop applications using basic programming constructs, facilitating the transition to other languages.
Module1 (Credit1)	
Learning Outcomes	**After learning this module, learners will be able to**
	1. Learn steps in problem solving using C.
	2. Understand structure, keywords, operators, and functions of C programming.
	3. Evaluate the concept of I/O functions, header files, and preprocessor directives.
	4. Design and apply concepts of the C language.
Content Outline	• **Introduction to problem solving**: Concept: Steps in problem solving - (Define Problem, Analyze Problem, Explore Solution), Problem solving techniques - (Trial & Error, Brain Storming, Divide & Conquer), Algorithms and Flowcharts (Definitions, Characteristics, Advantage & Disadvantages, Symbols, Examples), Pseudo-code (Definition, Conditional statements, Loops),etc. • **Overview of programming languages**: Definition of the program, Concept- Source code, Object code, Compilation, Interpretation, Execution, Input and Output, Debugging etc., Expressions, control structures; sub routines, Storage management; scoping rules; bindings for names, Storage types: Automatic, external, register and static variables • **Introduction to 'C' Language:** History of C Programming, Structures of „C" , Programming, Simple example, Basic Input/ Output, Function as building blocks.LanguageFundamentals:Characterset,CTokens,Keywords, Identifiers,Variables, Constant, Data Types, Comments

	• **Operators**: Types of operators, Precedence and Associativity, Expression. Statement and types of statements, Built-in Operators and function. Console based I/O and related built in I/O Function: printf(), scanf(), getch(), getchar(), putchar(), etc; Concept of header files, Preprocess or directives: #include,#define, Conditional statements and Loops
Module2 (Credit1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Gain proficiency in writing C programs to solve various problems.
	2. Learn the syntax and semantics of the C language, including its specific features such as pointers and memory management.
	3. Analyze the difference between structure and union.
	4. Design and handle operations of the files.
Content Outline	• **Control structures** ➢ Decision making structures: If, If-else, Nested If –else, Switch, Loop Control structures ➢ While, Do-while, For, Nested for, while, do-while loop, jumping statements: break, continue, go to, exit. • **Functions:** Definition, Basic types of function, Declaration and definition, Function call, Types of function, Parameter passing, Call by value, Call by reference, Scope of variables, Recursion, String: Declaration, string Functions, String Manipulations • **Pointers:** Introduction to pointers, Pointer notation, Pointer arithmetic, Null Pointer • **Arrays:** Definition, Declaration, Initialization, Bounds checking, One-Dimensional Array, Two-Dimensional Array, Passing array to a function, pointer to Array • **Structure and Union:** Introduction to Structure, Definition, Declaration of Structure Variables, Dot Operator, Nested Structure, Array of Structure, pointer to structure, Introduction to Union, Difference between Structure and Union • **File Handling:** Concept of File, Definition, File operations (create, open, read, move, write, close), File opening Mode, Closing a file, Input / output operations, Creating and reading a file, Command Line Argument

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)
Module 1

1. Create a flowchart and algorithm for a simple problem (e.g., calculating the factorial of a number).
2. Write a pseudo-code for the above problem.
3. Convert the pseudo-code into a C program.
4. Demonstrate the use of basic input/output functions such as printf() and scanf() in the program.
5. Include the use of variables, constants, and data types.
6. Write a C program to demonstrate the use of different operators (arithmetic, relational, logical, bitwise, etc.).
7. Create examples to illustrate the precedence and associativity of operators and evaluate expressions.

8. Include conditional statements and loops in the program to show complex expressions and their evaluations.
9. Demonstrate the use of console-based I/O functions such as printf(), scanf(), getch(), getchar(), putchar(), etc.
10. Illustrate the use of header files and preprocessor directives (#include, #define).

Module 2

1. Write C programs using different control structures (if, if-else, switch, while, do-while, for loops). Include programs that utilize nested loops and jumping statements (break, continue, go to).
2. Create programs that use functions to perform various tasks. Include examples of parameter passing (call by value and call by reference).
3. Write a program that includes recursion and demonstrates string manipulation using string functions.
4. Develop a program that uses pointers for arithmetic operations and demonstrates the concept of null pointers.
5. Write a program to handle arrays (one-dimensional and two-dimensional) and pass them to functions. Include pointer to array operations.
6. Write programs to perform basic file operations (create, open, read, write, close). Include programs that demonstrate reading from and writing to files.
7. Implement a program that uses command-line arguments for file operations.
8. Compare and contrast the use of structures and unions in handling data through a practical example in a program.

References:-

1. Schildt, H. (2000). C: The Complete Reference (4th ed.). Tata McGraw-Hill Education Pvt.Ltd.
2. Ramkumar, & Agrawal. (1996). Programming in ANSI C. Tata McGraw-Hill.
3. Kanetkar, Y. P. (2008). Let Us C. Infinity Science Press.

1.1 Major Core

Course Title	**C PROGRAMMING (LAB)**
Course Credits	**2**
Course Outcomes	**After completion of this Course, the students will be able to**
	1. Apply algorithms by writing C code.
	2. Analyze and trace the execution of C programs.
	3. Evaluate the use of pointers, arrays, and the pre-processor in programs.
	4. Design programs that utilize derived data types and implement simple file operations.
Module1 (Credit 1)	
Learning Outcomes	**After learning this Module, learners will be able to**
	1. Apply operators to write simple programs.
	2. Analyze and use control, iterative, and jumping statements.
	3. Evaluate the use of break and continue statements.
	4. Design programs with header files and preprocessor directives.

Content Outline	• **Simple Program** • **Implementation of Operators:** Built in Operators and function, Arithmetic, Logical, Relational, bitwise, Precedence And Associativity, composite statements. Unary, binary and ternary operators. • **Concept of header files**, Preprocessor directives: #include, #define. And macros implementations, Implementation of Storage types: Automatic external, register and static variables • **Console based I/O and related built in I/O function**: printf(), scanf(), getch(),getchar(), putchar(); • **Control Statement:** Decision Making Statements, if, Nested if, if-else, Nested if-else, if-else-if, switch, etc. The Conditional Expression; • **Iterative Statements**- The for loop, . The while loop, The do-while loop; Jumping Statements- The goto & label, The break & continue, The exit()function
Module 2 (Credit 1)	
Learning Outcomes	**After learning this Module, learners will be able to**
	1. Apply functions in programs.
	2. Analyze the declaration, initialization of pointers, and passing arrays to functions.
	3. Evaluate the definitions and declarations of structure variables in programs.
	4. Design programs effectively using functions, pointers, and structures.
Content Outline	• **Implementation of Functions:** Defining and accessing, passing arguments, Function prototypes, function calling mechanism, call by value, call by reference, recursive function; String Manipulations • **Pointer Declaration and Initialization** of Pointer variables, pointer Arithmetic, Pointers and Character Strings Implementation of 1-D and multidimensional Array, One-Dimensional Array, Two-Dimensional Array, Passing array to a function, pointer to Array. • **Programs Using Structure and Union:** Defining and Declaring Structure Variables, .Dot Operator, Nested Structure, Array of Structure, pointer to structure, Examples of Union. • **Programs using I/O Operations File Handling**: File Operations (Create, open, read, move, write, close) **Input/output operations on file Character by** –(fgetc, fputc), Reading and writing files

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1

Write a C program that implements a simple calculator. The program should prompt the user to enter two numbers and an operator (+, -, *, /). Based on the operator entered, perform the corresponding arithmetic operation and display the result.

1. Use appropriate control flow statements (if, else if, else or switch) to determine the operation to be performed based on the operator entered by the user.
2. Implement error handling to handle division by zero.
3. Utilize console-based input/output functions like printf(), scanf() for user interaction.
4. Make use of header files and preprocessor directives to organize your code and define any necessary constants.

Module 2

Write a C program that demonstrates the use of functions, pointers, and structures to manage student records. Each student record should contain the following information:

name, roll number, marks in three subjects.

1. Define a structure to represent a student record with appropriate data members.
2. Implement functions to perform the following tasks:
3. Input student details (name, roll number, marks).
4. Calculate the total marks and average marks of a student.
5. Display student details along with total and average marks.
6. Use pointers to pass structures to functions wherever necessary.
7. Ensure proper memory allocation and deallocation.
8. Implement file handling operations to read and write student records to a file using input/output functions like fscanf(), fprintf(), fopen(), fclose(), etc.

References:

1. Balagurusamy, E. (1990). C programming. Tata McGraw Hill.
2. Schildt, H. (2000). C: The Complete Reference (Fourth Edition). Tata McGraw-Hill Education Pvt. Ltd.
3. Ramkumar & Agrawal. (1996). Programming in ANSI C. Tata McGraw Hill.
4. Kanetkar, Y. P. (2008). Let Us C. Infinity Science Press.

Semester: I

1.2. Major (Core)

Course Title	**COMPUTER FUNDAMENTALS AND OPERATING SYSYTEM**
Course Credits	**2**
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply knowledge to set up and connect various computer peripherals and interfaces.
	2. Analyze and perform conversions between different number systems and execute binary arithmetic operations.
	3. Evaluate and troubleshoot basic networking concepts and protocols.
	4. Design and optimize operating system components to enhance their functions and overall system performance.
Module 1 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply concepts of number systems and computer arithmetic to solve computational problems.
	2. Analyze the differences and functionalities of system and application software, and various types of computers.
	3. Evaluate the characteristics and interpretations of data, including memory devices and computer languages.
	4. Design efficient process scheduling and synchronization mechanisms within various types of operating systems.
Content Outline	• **Number Systems**: Binary, Octal, Decimal, Hexadecimal and Their inter conversion, Computer Arithmetic. • **Computer Software:** System and Application Software. • **Type of Computers:** Digital, Analog, Hybrid Computers • **Definition:** Data, Information; Characteristics and Interpretation, Data & it's logical & physical Concepts, Definition of Computer, Features, Block Diagram of Computer System, Computer Generations • **Primary Memory Devices:** RAM, ROM, PROM,EPROM, CACHE Memory, Registers. • **Computer Languages:** Machine, Assembly, High Level • **Operating System:** Purpose of Operating Systems, OS Structure, Services of Operating System,Types of Operating System • **Processes**: Concept, process states, Scheduling, Operations on Processes, Cooperating Process, Process Synchronization.
Module 2 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply CPU scheduling algorithms (FCFS, SJF, RR, Priority) to optimize process scheduling.
	2. Analyze various memory management techniques, including swapping, contiguous memory allocation, paging, and segmentation.
	3. Evaluate different page replacement policies (LRU, OPT, SC, FIFO, NRU, MRU) to enhance memory management efficiency.
	4. Design memory management strategies incorporating advanced scheduling and page replacement algorithms to improve overall system performance.

Content Outline	• **CPU Scheduling:** Concept, Scheduling Criteria, Scheduling Algorithms (FCFS, SJF, RR, Priority). • **Memory Management:** Concept, Swapping, Contiguous Memory Allocation, Paging, Segmentation. • **Page Replacement policies**: Least Recently used(LRU) Optimal (OPT), Second Chance (SC), First in First Out (FIFO), Not recently used (NRU), Most Recently Used(MRU).

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

1. – **Module 1: Comprehensive Computer Systems Analysis and Implementation**
   - **Number Systems and Computer Arithmetic**
     - Convert the following numbers from one number system to another:
       Binary to Decimal: 101011
       Decimal to Hexadecimal: 175
       Octal to Binary: 257
     - Perform the following binary arithmetic operations:
       1011 + 1101
       1001 - 0110
   - **Computer Software**
     Compare and contrast system software and application software by listing at least 3 examples of each. Explain the primary functions of each type of software.
   - **Types of Computers**
     Describe the characteristics and uses of digital, analog, and hybrid computers. Provide one real-world example for each type.
   - **Data and Computer Systems**
     - Define the terms data and information. Explain the difference between the two with examples.
     - Draw a block diagram of a computer system and label its main components.
     - Describe the features and characteristics of each generation of computers from the first to the current generation.
   - **Primary Memory Devices**
     Explain the differences between RAM, ROM, PROM, EPROM, and CACHE memory. Include their primary uses and characteristics.
   - **Computer Languages**
     - Differentiate between machine language, assembly language, and high-level languages. Provide one example for each and explain their typical uses in programming.
   - **Operating Systems**
     - Describe the primary purpose and services provided by an operating system.
     - Explain the structure of an operating system with a diagram.
     - List and briefly describe different types of operating systems (e.g., batch, time- sharing, distributed).
   - **Processes and Scheduling**
     - Explain the concept of a process and its states. Illustrate the process state diagram.
     - Describe different CPU scheduling algorithms (FCFS, SJF, RR, Priority). Provide a scenario where each algorithm would be most appropriate.
     - Explain process synchronization and provide an example where it is crucial in an operating system.

2. – **Module 2: To apply, analyze, evaluate, and design CPU scheduling algorithms, memory management techniques, and page replacement policies through a comprehensive simulation and analysis activity.**

   CPU Scheduling:

   - **Define and explain the concepts of CPU scheduling and scheduling criteria.**
     Implement and compare the following scheduling algorithms: FCFS, SJF, RR, and Priority.

        - Task: Simulate a scenario with a set of processes, each having different burst times, arrival times, and priorities. Evaluate the performance of each scheduling algorithm based on average waiting time, turnaround time, and CPU utilization.
        - Deliverable: Provide a detailed report with your simulation code, a comparison of the results, and an analysis of which algorithm performs best under different conditions.
- **Memory Management:**
  Explain the concepts of memory management, including swapping, contiguous memory allocation, paging, and segmentation.
    - Task: Simulate a memory management system that includes swapping and both contiguous and non-contiguous allocation methods (paging and segmentation).
    - Deliverable: Create a memory map for each allocation method. Provide your simulation code, memory maps, and an analysis of the advantages and disadvantages of each method.
- **Page Replacement Policies:**
  Define and explain the following page replacement policies: LRU, OPT, SC, FIFO, NRU, and MRU.
    - Task: Simulate a scenario where a sequence of page requests is given. Implement each of the page replacement policies and evaluate their performance based on the number of page faults.
    - Deliverable: Provide your simulation code, a comparison of the number of page faults for each policy, and an analysis of which policy is most effective under different conditions.

References:

1. Stallings, W. (2021). Computer organization and architecture: Designing for performance (11th ed.). Pearson Education.
2. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts (10th ed.). Wiley.
3. Tanenbaum, A. S., & Austin, T. (2012). Structured computer organization (6th ed.). Pearson Education.
4. Patterson, D. A., & Hennessy, J. L. (2017). *Computer organization and design: The hardware/software interface* (5th ed.). Morgan Kaufmann.
5. Stallings, W. (2018). *Operating systems: Internals and design principles* (9th ed.). Pearson Education.

Semester: I

1.3 Open Elective Course-I (OEC)

Course Title	**WEB TECHNOLOGY (HTML, CSS, PHP, JAVASCRIPT)**
Course Credits	2
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply HTML5 and CSS to create and style web pages effectively.
	2. Analyze and implement JavaScript to develop interactive web pages.
	3. Evaluate and develop dynamic, database-driven web pages using PHP.
	4. Design and implement comprehensive client-side and server-side scripting programs.
Module 1 (Credit 1)	
Learning Outcomes	**After learning the module , learners will be able to**
	1. Apply HTTP concepts and security measures to manage client-server interactions, including persistent connections and cookies.
	2. Analyze and utilize HTML5 elements and tags to create structured and semantic web pages.
	3. Evaluate the effectiveness of HTML5 web forms in capturing various types of user input, ensuring usability and accessibility.
	4. Design and implement multimedia content and interactive form elements in web pages to enhance user experience.
Content Outline	• **Introduction to web and Security Concepts** HTTP:Overview – HTTP Basics, Client request, Server response; HTTP Headers; Session Management – Persistent connections, Cookies. General concepts on web server: virtual hosting, General concepts of caching proxy server , Web security, Digital signatures, Digital Certificates, Encryption, and Authentication • **HTML5** Basics of HTML elements and Tags. Introduction of HTML5 (evolutions, limitation of HTML4, advantages of HTML5, Overview of HTML5) • **Page Layout of Semantic Elements** Header,Navigation, Section & Articles, Footer Organizing Text in HTML, Links and URLs in HTML, Tables in HTML, Images on a Web Page, Image Formats, Image Maps, Colors, FORMs in HTML, Frames in HTML Working with Multimedia -Inserting Audio and Video on a web page,audio video file format. • **HTML5 Web Forms** HTML 5 Global Attributes Displaying a Search Input Field, Contact Information Input Fields, Utilizing Date and Time Input Fields, Number Inputs, Selecting from a Range of Numbers, Selecting Colors, Creating an Editable Drop-Down, Requiring a Form Field, Autofocusing a Form Field, Displaying Placeholder Text.
Module 2 (Credit 1)	
Learning Outcomes	**After learning the module , learners will be able to**
	1. CSS syntax, selectors, and properties effectively to style web pages, including background settings, fonts, text styles, and

	element positioning.
	2. Analyze and implement JavaScript programming fundamentals such as variables, operators, control flow statements, and core JavaScript objects (Array, Boolean, Date, Function, etc.), along with handling events and using browser objects.
	3. Evaluate the integration of PHP with HTML for server-side scripting, understanding syntax, variables, and passing information between web pages.
	4. Design and implement advanced JavaScript functionalities including the Document Object Model (DOM) manipulation, form validation techniques, and utilizing cookies for enhanced web application interactivity and functionality.
Content Outline	• **CSS:** Understanding the Syntax of CSS, CSS Selectors, Inserting CSS in an HTML Document, CSS properties to work with- background of a Page, CSS Fonts and Text Styles, positioning an element • **JavaScript:** Using JavaScript in an HTML Document, Programming Fundamentals of JavaScript - Variables, Operators, Control Flow Statements, Popup Boxes, Core JavaScript (Properties and Methods of Each) : Array, Boolean, Date, Function, Math, Number, Object, String, regExp, Events and Event Handlers, Browser Objects - Window, Navigator, History, Location, Document, Cookies, Document Object Model, Form Validation using JavaScript. • **INTRODUCTION TO PHP AND SQL**: - Server-side web scripting, Installing PHP, SQL, Adding PHP to HTML, Syntax and Variables, Passing information between pages.

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1:

1. Web and Security Concepts:
   - Explain the basics of HTTP, including client requests and server responses. Illustrate with an example of HTTP headers used in a web request.
   - Describe session management techniques such as persistent connections and cookies. Create a simple HTML page demonstrating the use of cookies.
2. HTML5 Basics:
   - Write an HTML5 document that includes various HTML elements and tags. Highlight the advantages of HTML5 over HTML4.
   - Create a simple web page layout using semantic elements such as header, navigation, section, articles, and footer.
3. Page Layout of Semantic Elements:
   - Develop a web page that organizes text, includes links, and embeds tables and images. Use different image formats and demonstrate the use of image maps.
   - Create a form in HTML5 with various input fields, including search, contact information, date and time, number inputs, and color picker. Include attributes like autofocus, placeholder text, and required fields.
4. Working with Multimedia:
   - Add audio and video elements to a web page. Provide examples of different audio and video file formats that can be used.

Module 2:

1. CSS:
   - Create an HTML document and apply various CSS styles. Use different selectors and properties to style the background, fonts, text, and position elements.
   - Develop a responsive web page layout using CSS for positioning elements.

2. JavaScript:
    - Write JavaScript code to demonstrate the use of variables, operators, and control flow statements. Include examples with popup boxes.
    - Create a web page with JavaScript that manipulates core objects like Array, Boolean, Date, Function, Math, Number, Object, String, and RegExp. Include event handlers and form validation.
3. Introduction to PHP and SQL:
    - Install PHP and configure it with a web server. Write a PHP script that integrates with an HTML form to capture user input.
    - Develop a simple web application that uses PHP to interact with a SQL database. Implement functionality to add, retrieve, update, and delete records.

References:-

1. Powell, T. (2000). Web design: The complete reference. Tata McGraw-Hill.
2. Powell, T. (2008). HTML and XHTML: The complete reference. Tata McGraw-Hill.
3. Powell, T., & Schneider, F. (2004). JavaScript 2.0: The complete reference (2nd ed.). Tata McGraw-Hill.
4. Holzner, S. (2017). PHP: The complete reference. Tata McGraw-Hill.
5. Lecky-Thompson, G. W. (2009). Web programming. Cengage Learning.

1.3 OEC

Course Title	**WEB TECHNOLOGY (LAB)**
Course Credits	2
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply HTML5 and CSS to design and style web pages.
	2. Analyze and implement JavaScript to create interactive web pages.
	3. Evaluate and develop dynamic, database-driven web pages using PHP.
	4. Design and implement comprehensive client-side and server-side scripting programs.
Module 1 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply basic HTML tags to design web pages with organized text, links, tables, images, forms, and multimedia elements.
	2. Analyze and utilize HTML5 semantic elements (navigation, section, articles, footer, etc.) to structure web page content effectively.
	3. Evaluate and implement CSS syntax and properties to enhance web page aesthetics, including backgrounds, fonts, text styles, and element positioning.
	4. Design interactive and multimedia-rich web pages by integrating HTML and CSS, ensuring proper use of image maps, audio, and video file formats.
Content Outline	• **Use of Basic Tags, Image maps, Tables, Forms and Media** Design webpages using the given tools in HTML Navigation, Section & Articles, Footer, aside and more. Organizing Text in HTML, Links and URLs in HTML, Tables in HTML, Images on a Web Page, Image Formats, Image Maps, Colors, FORMs in HTML, Frames in HTML Interactive Elements, Working with Multimedia - Audio and Video File Formats, HTML elements for inserting Audio / Video on a web page • **CSS**

	Syntax of CSS, CSS Selectors, Inserting CSS in an HTML Document, CSS properties to work with background of a Page, CSS properties to work with Fonts and Text Styles, CSS properties for positioning an element
Module 2 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply control flow and looping statements in JavaScript to create dynamic and interactive web pages.
	2. Analyze and utilize core JavaScript properties and methods (Array, Boolean, Date, Function, Math, Number, Object, String, RegExp) along with events and event handlers to enhance web page functionality.
	3. Evaluate and implement PHP in web development, including syntax, variables, and the integration of PHP with HTML for passing information between pages.
	4. Design and develop database-driven web applications using PHP and SQL, demonstrating the installation and configuration of PHP and the creation of dynamic content.
Content Outline	• **Java Script** : Control and looping statements and Java Script reference, Using JavaScript design, a web page; Control Flow, Statements, Design a web page demonstrating different conditional statements. Design a web page demonstrating different looping statements; Popup Boxes, Core JavaScript (Properties and Methods of Each) : Array, Boolean, Date, Function, Math, Number, Object, String, regExp, Events and Event Handlers • **PHP & SQL** Demonstrate program in PHP, Installing PHP, SQL, Adding PHP to HTML, Syntax and Variables, Passing information between pages.

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1:
Create a comprehensive web page incorporating basic HTML tags, image maps, tables, forms, and multimedia elements, and style it using CSS.
1. HTML Structure:
- Design a web page using HTML that includes a header, navigation bar, sections, articles, aside, and footer.
- Organize text with proper HTML tags.
- Insert links and URLs.
- Create tables to display data.
- Add images in various formats and create image maps.
- Develop forms to capture user input with various input fields.
- Include interactive elements using frames.
- Embed audio and video files using HTML multimedia elements.

2.CSS Styling:
- Apply CSS to style the web page.
- Use CSS selectors and properties to customize the background, fonts, text styles, and positioning of elements.
- Ensure the web page is visually appealing and follows a consistent design theme.

Module 2:
Develop a dynamic web application using JavaScript for client-side scripting and PHP with SQL for server-side scripting.
1.JavaScript Functionality:

- Design a web page with JavaScript to demonstrate different control flow statements (if-else, switch) and looping statements (for, while).
- Implement popup boxes and core JavaScript objects (Array, Boolean, Date, Function, Math, Number, Object, String, RegExp).
- Add event handlers to manage user interactions and form validations.

2.PHP and SQL Integration:

- Install PHP and configure it with your web server.
- Write PHP scripts to process form data and interact with a SQL database.
- Create a database and implement CRUD (Create, Read, Update, Delete) operations.
- Pass information between web pages using PHP.

References:-

1. Lecky-Thompson, G. W. (2009). Web programming. Cengage Learning.
2. Powell, T. (2000). Web design: The complete reference. Tata McGraw-Hill.
3. Powell, T. (2008). HTML and XHTML: The complete reference. Tata McGraw-Hill.
4. Powell, T., & Schneider, F. (2004). JavaScript 2.0: The complete reference (2nd ed.). Tata McGraw-Hill.
5. Holzner, S. (2017). PHP: The complete reference. Tata McGraw-Hill.

Semester – I

1.4 VSC

Course Title	**OFFICE AUTOMATION TOOLS**
Course Credits	**2**
Course Outcomes	**After going through the course, learners will be able to**
	1. Apply office automation concepts and technologies to integrate office tools and equipment, enhancing communication efficiency in the workspace.
	2. Analyse and evaluate proficiency in Writer and Calc, demonstrating mastery in text formatting, styles, graphics, tables, formulas, functions, and data analysis techniques.
	3. Design and manage complex documents and data using advanced features such as master documents, fields, forms, mail merge, data linking, collaboration tools, reviewing capabilities, and macros.
	4. Create and customize presentations in Impress, applying design principles to master slide creation, text and graphic formatting, slide transitions, animations, and various exporting options.
Module 1 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply the concepts of office automation by integrating office tools, equipment, and technologies to optimize workspace communication and efficiency.
	2. Analyse and evaluate proficiency in using the Writer tool for text formatting, style application, graphics and table integration, mail merge operations, and document customization.
	3. Design and manage complex documents by applying advanced features like templates, master documents, fields, forms, and creating tables of contents, indexes, and bibliographies.
Content Outline	• **Concept of Office Automation:** Purpose of an office, activities in an office ,structure of an office, office manual, document flow management in an office, need for office automation and its advantages and disadvantages, Office automation tools. • **Office Automation Technology:** Office equipment, Workstation communication and convergence of Technologies • **Writer** -Introducing Writer -Working with Text - • **Formatting Pages** - Printing, Faxing, Exporting, and Emailing • **Introduction to Styles** - Working with Styles - • **Working with Graphics** - Working with Tables – Working with Templates in Writer - Using Mail Merge – Creating Tables of Contents, Indexes, and Bibliographies - • **Working with Master Documents** - Working with Fields - Using Forms in Writer- Customizing Writer
Module 2 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply Calc skills for proficient data management, analysis, and sharing, utilizing formulas, functions, and macros to enhance productivity.
	2. Analyse and evaluate proficiency in creating and customizing presentations in Impress, employing text and graphic formatting, slide transitions, animations, and various exporting options.

Content Outline	• **Calc**: Introducing Calc, Entering, Editing, and Formatting Data, Using Charts and Graphs, Using Styles and Templates, Using Graphics in Calc, Printing, Exporting, and E-mailing, Formulas and Functions, Using the Data Pilot, Data Analysis, Linking Calc Data, Sharing and Reviewing, Calc Macros • **Impress:** Guide Introducing Impress, Using Slide Masters, Styles, and Templates, Adding and Formatting Text, Adding and Formatting Pictures, Managing Graphic Objects, Formatting Graphic Objects, Spreadsheets, Charts, and Other Objects, Slides, Notes, and Handouts, Slide Shows : Transitions, Animations, Printing, Emailing, Exporting, and Saving Slide Shows, Setting Up and Customizing Impress

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1:

- Research and identify three different office automation tools available in the market.
- Evaluate their advantages and disadvantages in terms of optimizing communication and efficiency in a workplace setting.
- Choose one tool and create a presentation using Impress to explain its features and benefits to your colleagues.

Module 2:

- Using Writer, create a complex document that includes text formatting, graphics, tables, and styles.
- Apply mail merge functionality to personalize documents for a hypothetical mailing list of 20 recipients.
- Evaluate the effectiveness of Writer's features for document customization and mail merge operations in a short reflective report (500 words).

- Design a comprehensive office manual template in Writer that includes sections for document flow management, office structure, and activities.
- Incorporate advanced features such as master documents, fields, and forms to create a dynamic and interactive document.
- Produce a table of contents, an index, and a bibliography within the manual template to demonstrate proficiency in document design and management.

References

1. Weverka, P. (2015). Office 2016 All-In-One For Dummies. Wiley.
2. Libre Office Documentation Team. (2016). Libre Office: Writer, Calc, Math Formula Book. Friends of OpenDocument, Inc.
3. Leete, G., Finkelstein, E., & Leete, M. (2003). OpenOffice.org For Dummies. Wiley.
4. Weverka, P. (2018). Office 2019 All-in-One For Dummies. Wiley.
5. Libre Office Documentation Team. (2016). Libre Office 5.2 Writer Guide. Friends of OpenDocument, Inc.
6. Bain, M. A. (2006). Learn OpenOffice.org Spreadsheet Macro Programming: OOo Basic and Calc automation. A press.

Semester: I

1.5. Skill Enhancement Course (SEC)

SWAYAM/CHETNA/MOOCS course

Semester: I

1.6 AEC (Ability Enhancement Course)

Semester: I

1.7 IKS (Indian Knowledge System)

Semester: I

1.8 VEC (Value Education Course)

Semester: I

1.9 Co-Curricular Course-I (CC)

Course Syllabus

Semester: II

2.1. Major (Core)

Course Title	**DATA STRUCTURES AND ALGORITHMS**
Course Credits	**2**
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply abstract data structures using arrays and linked lists.
	2. Analyze the efficiency and performance of algorithms.
	3. Evaluate data structures like arrays, linked lists, stacks, queues, binary trees, and graphs for different applications.
	4. Design and apply appropriate searching and sorting techniques.
Module 1 (Credit 1)	
Learning Outcomes	**After learning this module, learners will be able to**
	1. Apply various operations on linear data structures such as arrays, linked lists, stacks, and queues.
	2. Analyze algorithm characteristics, including space and time complexity, using asymptotic notation.
	3. Evaluate the advantages and disadvantages of different types of linked lists and their operations.
	4. Design and implement stack and queue structures, and apply their operations in different contexts.
Content Outline	• **Algorithm Analysis:** Algorithm Characteristics, Space complexity, Time complexity. Asymptotic notation (Big O, 0, Omega and Theta) • **Introduction:** Introduction to data structure, Classification of data structure, Operations performed on data structures, Linear data structure, arrays, operations on an array • **Linked List:** Introduction, Key terms, Advantages & disadvantages, Linear linked lists - Types (Singly, Doubly, Circular) Operations (Inserting, Deleting nodes) • **Stack:** Introduction, Stack implementation, Operations on stack (Push Pop), Implementation of stack using pointer, Applications of stack, Infix prefix, postfix notations • **Queue :** Introduction and Queue implementation, Operations
Module 2 (Credit 1)	
Learning Outcomes	**After learning this module, learners will be able to**
	1. Apply various traversal techniques to binary trees and graphs.
	2. Analyze different sorting algorithms such as Bubble Sort, Quick Sort, and Heap Sort.
	3. Evaluate the efficiency of search algorithms like Linear Search and Binary Search.
	4. Design binary trees and graphs using array and linked list

	representations.
Content Outline	• **Trees:** Introduction, terminology, Binary tree, Strictly Binary tree, Complete Binary tree, Binary tree representation as Array and Linked lists, Traversal (Inorder, Preorder, Post order), Binary Search Tree • **Graphs:** Introduction, terminology, Graph representation, Applications of graph, Graph traversal (BFS, DFS, Shortest path), Spanning tree, Minimum spanning tree • **Sorting & Searching:** Bubble Sort, Selection Sort, Quick Sort, Heap Sort, Insertion Sort. Linear Search , Binary Search

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1 : Implementing and Analyzing Linear Data Structures

1. Implement the following linear data structures using your preferred programming language:
   - Array
   - Singly Linked List
   - Doubly Linked List
   - Circular Linked List
   - Stack
   - Queue
2. Write functions for basic operations on these data structures:
   - For arrays: insertion, deletion, and traversal.
   - For linked lists: insertion, deletion, and traversal.
   - For stacks: push and pop.
   - For queues: enqueue and dequeue.
3. Algorithm Analysis:
   Analyze the time complexity of the implemented operations using asymptotic notation (Big O, Omega, Theta).

Module 2: Traversal and Sorting Techniques

1. To apply traversal techniques to binary trees and graphs, analyze sorting algorithms, evaluate search algorithms, and design tree and graph structures.
2. Implement the following types of binary trees:
   - Binary Tree
   - Strictly Binary Tree
   - Complete Binary Tree
   - Represent these trees using arrays and linked lists.
3. Implement the traversal methods for binary trees:
   - Inorder
   - Preorder
   - Postorder
4. Implement graph representations using adjacency matrix and adjacency list.
   Implement graph traversal algorithms:
   - Breadth-First Search (BFS)
   - Depth-First Search (DFS)
5. Sorting Algorithms:Implement the following sorting algorithms:
   - Bubble Sort
   - Selection Sort
   - Quick Sort
   - Heap Sort
   - Insertion Sort

6. Search Algorithms:Implement Linear Search and Binary Search.

References:-

1. Trembley, P., & Sorrenson, R. (2005). Data Structure. McGraw Hill Publication.
2. Lipschuiz, S. (2014). Data structures (Schaum's Outline Series). McGraw Hill Publication.
3. Horowitz, E., & Sawhney, S. (2008). Fundamentals of Computer Algorithms. Computer Sci.P
4. Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1982). Data Structures and Algorithms. Pearson
5. Abhyankar, A. (2016) Data Structures and Files. Nirali Prakashan, Educational Publishers
6. Baluja, G. S. (2022). Data Structures Through C. VISIONIAS
7. Loomis, M. E. S. (1989). Data Management and File Structures (2nd ed.). Prentice Hall.
8. Samanta, D. (2009). Classical Data Structures. PHI, New Delhi**.**

2.1Major (Core)

Course Title	**DATA STRUCTURES AND ALGORITHMS (LAB)**
Course Credits	**2**
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply abstract data structures using arrays and linked lists.
	2. Analyze the efficiency and performance of algorithms.
	3. Evaluate data structures like arrays, linked lists, stacks, queues, binary trees, and graphs for different applications.
	4. Design and apply appropriate searching and sorting techniques.
Module 1 (Credit 1)	
Learning Outcomes	**After learning this module, learners will be able to**
	1. Apply various operations on linear data structures such as arrays, linked lists, stacks, and queues.
	2. Analyze algorithm characteristics, including space and time complexity, using asymptotic notation.
	3. Evaluate the advantages and disadvantages of different types of linked lists and their operations.
	4. Design and implement stack and queue structures, and apply their operations in different contexts.
Content Outline	• **Arrays:** Implementations of Array and Operations- Insertion, deletion of an element from one dimensional array, Traversing of array • **Linked List:** Implementation of List and Linked List and Operations- Inserting, Deleting of nodes etc • **Stack:** Stack Implementation, Operations on stack (Push Pop). Implementation of stack • **Queue:** Implementation of Queue Implementation, Operations on queue(Insertion and deletion)
Module 2 (Credit 1)	
Learning Outcomes	**After learning this module, learners will be able to**
	1. Apply various traversal techniques to binary trees and graphs.
	2. Analyze different sorting algorithms such as Bubble Sort, Quick Sort, and Heap Sort.
	3. Evaluate the efficiency of search algorithms like Linear Search and Binary Search.
	4. Design binary trees and graphs using array and linked list representations.

Content Outline	**Chapter 5 : Trees** Implementation of tree as Array and Linked lists and Traversal (Inorder, Preorder, Postorder) **Chapter 6: Graphs** Implementation of Graph traversal (BFS, DFS) **Chapter 7 Sorting and Searching** Implementation of searching (Sequential, Binary search) Sorting (Bubble sort, Selection sort, Insertion Sort.)

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1 : Implementing and Analyzing Linear Data Structures

1. Implement the following linear data structures using your preferred programming language:
    - Array
    - Singly Linked List
    - Doubly Linked List
    - Circular Linked List
    - Stack
    - Queue
2. Write functions for basic operations on these data structures:
    - For arrays: insertion, deletion, and traversal.
    - For linked lists: insertion, deletion, and traversal.
    - For stacks: push and pop.
    - For queues: enqueue and dequeue.
3. Algorithm Analysis:
   Analyze the time complexity of the implemented operations using asymptotic notation (Big O, Omega, Theta).

Module 2: Traversal and Sorting Techniques

1. To apply traversal techniques to binary trees and graphs, analyze sorting algorithms, evaluate search algorithms, and design tree and graph structures.
2. Implement the following types of binary trees:
    - Binary Tree
    - Strictly Binary Tree
    - Complete Binary Tree
    - Represent these trees using arrays and linked lists.
3. Implement the traversal methods for binary trees:
    - Inorder
    - Preorder
    - Postorder
4. Implement graph representations using adjacency matrix and adjacency list.
   Implement graph traversal algorithms:
    - Breadth-First Search (BFS)
    - Depth-First Search (DFS)
5. Sorting Algorithms:Implement the following sorting algorithms:
    - Bubble Sort
    - Selection Sort
    - Quick Sort
    - Heap Sort
    - Insertion Sort
6. Search Algorithms:Implement Linear Search and Binary Search.

References:-

1. Trembley, P., & Sorrenson, R. (2005). Data Structure. McGraw Hill Publication.
2. Lipschuiz, S. (2014). Data structures (Schaum's Outline Series). McGraw Hill Publication.
3. Horowitz, E., & Sawhney, S. (2008). Fundamentals of Computer Algorithms. Computer Sci.P
4. Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1982). Data Structures and Algorithms. Pearson
5. Abhyankar, A. (2016) Data Structures and Files. Nirali Prakashan, Educational Publishers
6. Baluja, G. S. (2022). Data Structures Through C. VISIONIAS
7. Loomis, M. E. S. (1989). Data Management and File Structures (2nd ed.). Prentice Hall.
8. Samanta, D. (2009). Classical Data Structures. PHI, New Delhi**.**

Semester: II

2.2. Major (Core)

Course Title	**COMPUTER ORGANIZATION AND ARCHITECTURE**
Course Credits	**2**
Course Outcomes	**After Completion of this Course, students will be able to**
	1. Apply basic concepts to understand the structure of computers.
	2. Analyze the working of different interrupts and mapping techniques, and study register organization.
	3. Evaluate different addressing modes and their functionalities.
	4. Design and demonstrate the working of the central processing unit, including RISC and CISC architectures.
Module 1 (Credit 1)	
Learning Outcomes	After learning the module, learners will be able to
	1. Apply knowledge to understand the basic structure of a computer.
	2. Analyze memory organization and its components.
	3. Evaluate different memory types and their roles within a computer system.
	4. Design and implement memory organization techniques in computing scenarios.
Content Outline	• **Basic Structure of computers:** Basic organization of computer, Intel 8086 Architecture, Basic Measures of Computer Performance, CPU: Registers, Computer Function: Instruction Cycle, Interrupts, Interconnection Structures, Bus Interconnection, Peripheral Component Interconnection (PCI). • **Memory Organization:** Classifications of primary and secondary memories. Types of RAM (SRAM, DRAM, SDRAM, DDR, SSD) and ROM, Characteristics of memory, Memory hierarchy: cost and performance measurement. • **Cache Memory:** Principles, Elements of cache design (Size, Mapping, Replacement, Write policies, Block size) Virtual Memory Concept.
Module 2 (Credit 1)	
Learning Outcomes	After learning the module, learners will be able to
	1. Apply various I/O techniques, including programmed I/O, interrupt-driven I/O, and direct memory access, to manage external devices and I/O modules.
	2. Analyze the instruction sets, operand types, operations on operands, and addressing modes of the 8086 processor.
	3. Evaluate the characteristics and performance differences between RISC and CISC architectures, including their pipelining capabilities.
	4. Design processor and register organizations based on the principles of RISC and CISC instruction execution.
Content Outline	• **Input/Output:** External devices, I/O Modules, Programmed I/O, Interrupted-Driven I/O, Direct Memory Access. • **Central Processing Unit:** Instruction sets: Instruction characteristics, Types of operands, Types of operations on operands, addressing modes of 8086 processor, Processor Organization, Register organization.

	• **RISC:** Instruction Execution, RISC Characteristics, and RISC Pipelining, RISC Vs. CISC, Reduced Instruction Set Computers (RISCs), Introduction to CISC. CISC Characteristics

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1:

1. Study the basic organization of a computer system, focusing on Intel 8086 Architecture.
2. Explore the fundamental measures of computer performance and the role of CPU registers in the instruction cycle.
3. Classify primary and secondary memories, and analyze their classifications and characteristics.
4. Evaluate different types of RAM (SRAM, DRAM, SDRAM, DDR) and ROM, considering their cost and performance metrics.
5. Study the principles of cache memory.
6. Design elements of cache memory, including size, mapping, replacement policies, write policies, and block size considerations.
7. Discuss the concept of virtual memory and its implementation in modern computing.
8. Design a memory hierarchy system that optimizes both cost and performance for a specific computing scenario.
9. Implement the designed memory organization techniques in a simulation environment using appropriate tools or programming languages.

Module 2 :

1. Implement various I/O techniques such as programmed I/O, interrupt-driven I/O, and direct memory access (DMA) to manage external devices and I/O modules.
2. Analyze the effectiveness and suitability of each technique in different computing scenarios.
3. Analyze the instruction sets, including characteristics, types of operands, and operations on operands of the Intel 8086 processor.
4. Evaluate the addressing modes and their implications for program execution efficiency.
5. Compare and contrast RISC and CISC architectures in terms of instruction execution efficiency and complexity.
6. Evaluate the advantages and disadvantages of RISC pipelining and its impact on processor performance.
7. Design a processor architecture based on either RISC or CISC principles, considering instruction set design and register organization.
8. Discuss how the designed architecture enhances performance and efficiency in specific computing tasks.

Reference Books:

1. Stallings, W. (2013). Computer organization and architecture: Designing for performance (10th ed.). Pearson.
2. Hayes, J. P. (1988). Computer architecture and organization. McGraw-Hill.
3. Hall, D. V. (2005). Microprocessor and interfacing (2nd ed.). Tata McGraw-Hill.
4. Brey, B. B. (2009). The Intel microprocessors 8086/8088… (4th ed.). PHI.
5. Tanenbaum, A. S. (2016). Structured computer organization (6th ed.). Pearson.
6. Mano, M. (2007). Computer system architecture (3rd ed.). Pearson.
7. Hwang, K., & Briggs, F. A. (1986). Computer architecture and parallel processing. McGraw-Hill.
8. Chaudhuri, P. P. (2004). Computer organization and design. Prentice Hall India.
9. Usha, M., & Shrikant, T. S. (2014). Computer system architecture and organization. Wiley India.

Semester: II

2.3 Minor Stream

Course Title	**Object Oriented Programming Using C++**
Course Credits	**2**
Course Outcomes	**After going through the course, learners will be able to**
	1. Apply object-oriented programming concepts in C++.
	2. Analyze problems and develop C++ programs to solve them.
	3. Evaluate the use of file input/output in C++.
	4. Design solutions using object-oriented programming principles in C++.
Module 1(Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply fundamental programming concepts including variables, data types, control structures, functions, arrays, and objects.
	2. Analyze and implement object-oriented programming concepts like objects, classes, and defining functions and variables.
	3. Evaluate the use of object-oriented programming in solving programming problems.
	4. Design solutions using a combination of fundamental programming and object-oriented concepts.
Content Outline	• **Evolution of OOP:** Advantages and disadvantages of OOP over its predecessor paradigms. • **Characteristics of Object-oriented Programming:** Abstraction, Encapsulation, Data hiding, Inheritance, Polymorphism, Code Extensibility and Reusability, User defined Data Types. • C++Program Structure, Simple Input/ Output Program, Program Comments, Identifiers, Literals, String, Character, Integer, Floating Point, Constants, Keywords, Data Types, Operators in C++, Control Structures in C++. • **Object and Classes:** Core object concepts, Encapsulation, Abstraction, Polymorphism, • Classes, Messages Association, Interfaces, Implementation of class in C++, C++ Objects as physical object, C++ object as data types constructor Object as function arguments. • **Functions and Variables:** Functions: Declaration and Definition, Variables: Definition Declaration, and Scope, Dynamic Creation and Derived Data, Arrays and Strings in C++.
Module 2(Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply concepts of constructors, inheritance (including its types), and polymorphism in C++.
	2. Analyze file input/output handling in C++ and class templates.
	3. Evaluate the effectiveness of constructors, inheritance, and polymorphism in solving programming tasks.
	4. Design solutions involving file input/output operations and the utilization of class templates in C++.
Content Outline	• **Inheritance**: Concept of Inheritance, Derived class and base class, Types of Inheritance, Functions and Friend Functions. • **Constructors**: Multiple Constructors and Initialization, Using estructors to Destroy Instances. • **Polymorphism**: Syntax for Operator overloading, overloading of unary and binary operators. • **File input and output:** Reading a File, Managing I/O

	Streams, opening a File – Different Methods, Checking for Failure with File Commands • **Class templates:** Implementing a class template, implementing class template member functions, Using a class template, Function template.

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)
Module 1:

- Develop a console-based application in C++ that demonstrates the implementation of fundamental programming concepts and object-oriented principles.
- Create a C++ program that includes classes representing real-world entities (e.g., a student, a car, a bank account).
- Implement basic functionality within these classes, such as setting and retrieving object attributes, defining member functions for data manipulation, and demonstrating encapsulation and abstraction.
- Utilize inheritance to create derived classes that inherit properties and behaviors from base classes, showcasing the concept of code reusability.
- Incorporate polymorphism by defining virtual functions and overriding them in derived classes to demonstrate runtime polymorphism.
- Implement file input/output operations to store and retrieve data related to objects, showcasing the handling of persistent data using C++.
- Document your code thoroughly and provide comments to explain the purpose and functionality of each component.

Module 2 :

- Design and implement a template-based class hierarchy in C++ for managing a generic data structure.
- Define a base template class that represents a generic data structure (e.g., a linked list, a stack, a queue).
- Implement derived template classes that inherit from the base class and specialize it to handle specific data types or functionalities (e.g., a linked list of integers, a stack of strings).
- Utilize constructor overloading to provide flexibility in initializing instances of the template classes.
- Implement operator overloading to enable intuitive manipulation of objects within the class hierarchy (e.g., addition, subtraction for mathematical operations).
- Use file input/output operations to demonstrate the serialization and deserialization of objects of the template classes.
- Test your implementation with various data types and scenarios to ensure correctness and functionality.
- Provide comprehensive documentation explaining the design decisions, implementation details, and usage instructions for the template-based class hierarchy.

References:

1. Balaguruswamy, E. (2008). Object Oriented Programming with C++. Tata McGraw–Hill
2. Education.
3. Venugopal, K. R. (1997). Mastering C++. Tata McGraw-Hill Education.
4. Stroustrup, B. (1997). C++ Programming Language (3rd ed.). Addison Wesley.
5. Chandra, B. (1998). A Treatise On Object Oriented Programming using C++. Narosa Publications.
6. Schildt, H. (2001). The Complete Reference CN. Tata McGraw-Hill.

Semester II

2.4. OEC (Open Elective Course-II)

Course Title	**Multimedia System**
Course Credits	**4**
Course Outcomes	**After going through the course, learners will be able to**
	1. Apply creativity, organization, and communication effectively in project stages.
	2. Analyse and optimize text, images, sound, and video for different mediums.
	3. Evaluate mastery of multimedia software tools for image editing, sound editing, animation, and video production.
	4. Design efficient file management strategies using compression techniques like CODECs, GIF, JPEG, MPEG to control file sizes.
Module 1 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply understanding of multimedia significance and project development stages.
	2. Analyse font types, utilize text editing tools, and design web-specific multimedia content.
	3. Evaluate production of still images using bitmap and vector drawing techniques, and understanding of color theory and image file formats.
	4. Design application of digital audio concepts, video technology workings, and animation principles in multimedia projects effectively.
Content Outline	• **INTRODUCING MULTIMEDIA:** Multimedia- Definitions, Use of Multimedia, Introduction To Making Multimedia: The Stages of a Multimedia Project, Need, Creativity, Organization, Communication. • **Tex**t- About Fonts and Faces, Cases, Serif Versus Sans Serif, Using Text in Multimedia, Computers and Text, Font editing and design tools, Hypermedia and Hypertext. Designing for the World Wide Web-Developing for the Web, Text for the Web, Images for the Web, Sound for the Web, Animation for the Web. • **IMAGES:** Images: Making Still Images, Bitmaps, Vector Drawing, 3-D Drawing and Rendering, Color, Understanding Natural Light and Color, Computerized Color, Color Palettes, Image File Formats.
Module 2 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply multimedia system sound principles, including digital audio, MIDI, and various audio file formats.
	2. Analyse differences between MIDI and digital audio technologies in multimedia contexts.
	3. Evaluate methods for audio CD playback, recording, and voice recognition systems.
	4. Design and implement video technologies, including broadcast standards, digital video formats, and techniques for optimizing video files and animations for multimedia applications.
Content Outline	• **Sound:** Multimedia System Sounds, Digital Audio, MIDI Audio, Audio File Formats, MIDI vs Digital Audio, Audio CD Playback.

	Audio Recording. Voice Recognition & Response. • **Video:** How Video Works, Broadcast Video Standards: NTSC, PAL, SECAM, ATSC DTV, Analog Video, Digital Video, Digital Video Standards – ATSC, DVB, ISDB, Video recording & Shooting Videos, Video Editing, Optimizing Video files for CD-ROM, Digital display standards. • **Animation**: Principle of Animations. Animation Techniques, Animation File Formats
Module 3 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply image editing techniques using selection tools, layers, masks, and channels in software applications.
	2. Analyse the differences between Macintosh and Windows hardware platforms, including connectivity options like IDE, SCSI, ATA, USB, and Firewire.
	3. Evaluate storage devices and multimedia input/output devices suitable for multimedia projects.
	4. Design strategies for producing web-ready files and integrating typographic design and vector drawing into multimedia content.
Content Outline	• **IMAGE EDITING:** Image Editing software: selection tools, working with layers, masks and channels, correcting and enhancing photographs, typographic design and vector drawing, working with 3D images, producing files for the web. • **Hardware:** Macintosh versus Windows, Connections: IDE, SCSI, UIDE, ATA, USB, Firewire etc. Storage devices, Input , Output devices for Multimedia Projects
Module 4 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply mastery of image editing software for diverse content creation.
	2. Analyse hardware differences to select appropriate devices.
	3. Evaluate proficiency in various multimedia software tools.
	4. Design effective application of compression principles for web development.
Content Outline	• **Multimedia Software Tools:** Text Editing & Word processing tools, OCR S/W, Painting &Drawing Tools, 3D Modelling & Animation Tools, Image editing tools, Sound Editing tools, Animation, Video & Digital movie tools, Overview of various types of Multimedia Authoring tools. • **Compression**: CODEC, Types of Compression & redundancies, GIF, JPEG & MPEG Standards Overview, Fractals • **Multimedia tools for WWW & Designing for WWW:** Plug Ins, Text, Images, Sound &Animation for the Web

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1:

1. Assignment: Multimedia Project Proposal:
   - Develop a proposal for a multimedia project, including objectives, target audience, and content outline.
   - Present the proposal to the class, highlighting the need for creativity and effective communication.
2. Assignment: Font Analysis and Design:
   - Analyze different font types and their usage in multimedia.

- Design a multimedia presentation focusing on font selection and its impact on communication.

Module 2:

1. Assignment: Image Creation and Editing Task:
    - Create still images using bitmap and vector drawing techniques.
    - Edit and enhance the images using image editing software, considering color theory and file formats.
2. Assignment: Audio Recording and Editing Exercise:
    - Record and edit audio clips using sound editing software.
    - Explore various audio file formats and techniques for enhancing audio quality.

Module 3:

1. Assignment: Video Production and Optimization Project:
    - Produce a short video clip, applying principles learned about video production, editing, and optimization.
    - Optimize the video for CD-ROM distribution and digital display standards.
2. Assignment: Image Editing and Web Production Task:
    - Use image editing software to produce web-ready graphics, focusing on selection, layering, and optimization techniques.
3. Assignment: Hardware and Device Comparison:
    - Research and compare Macintosh and Windows systems, as well as various hardware connections and storage devices.
    - Present findings in a comparative analysis report.

Module 4:

1. Assignment: Multimedia Software Exploration:
    - Explore various multimedia software tools, including text editing, painting, 3D modeling, animation, and sound editing.
    - Create a multimedia project using a combination of these tools.
2. Assignment: Compression Techniques Analysis:
    - Investigate different compression techniques and standards such as CODECs, GIF, JPEG, MPEG, and Fractals.
    - Compare and contrast techniques and present findings in a multimedia presentation.

References:

1. Buford, J. F. K. (2002). Introduction to Multimedia Systems.Pearson.
2. Vaughan, T. (1999). Introduction to Multimedia. McGraw-Hill Osborne Media
3. Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing.Pearson.
4. Pratt, W. K. (1991). Digital Image Processing. John Wiley & Sons
5. Adobe Creative Team. (2021). Adobe Photoshop Classroom in a Book. Pearson.

Semester: II

2.5 VSC

Course Title	**LINUX OPERATING SYSYTEM**
Course Credits	**2**
Course Outcomes	**After going through the course, learners will be able to**
	1. Apply Linux environment skills effectively for various tasks and operations.
	2. Analyse and utilize basic Linux commands and shell scripting techniques.
	3. Evaluate file system creation, directory management, and related operations using Linux programs.
	4. Design and implement solutions using a diverse set of standard Linux programming and development tools.
Module 1 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply basic Linux commands proficiently for various tasks and operations.
	2. Analyse the significance of Linux architecture and its key features.
	3. Evaluate Linux utilities for creating and managing simple file processing operations effectively.
	4. Design and demonstrate memory management techniques in file handling, including file and region locking mechanisms.
Content Outline	• **Introduction to Linux** ◦ Introduction to Unix architecture, ◦ General-purpose utilities ◦ All basic commands introduction. ◦ Usage of commands. • **Files and Directories:** File Concept, File types, File System Structure, File metadata- Inodes, kernel support for files, file System calls for file I/O operations- open, create, read, write, close. • **Directories-** mkdir, rmdir, chdir, obtaining current working directorygetcwd, directory contents, scanning directories- opendir, readdir, closedir, rewind dir functions.
Module 2 (Credit 1)	
Learning Outcomes	**After learning the module, learners will be able to**
	1. Apply the concept of filters in Linux to manipulate data effectively.
	2. Analyse different editors available in Linux and their functionalities.
	3. Design and implement shell scripts capable of performing complex tasks in a shell programming environment.
	4. Evaluate the role and functionalities of the Linux kernel in system operations.

Content Outline	• **The vi Editor-** Introduction to the vi editor of Linux.all commands. • **Introduction to filters**- Simple Filters, Filters using regular expressions - use of grep command • **Introduction to shell concept and writing shell script-** What is kernel, What is Shell,what is terminal,Advantage and disadvantage, first simple Shell Programme.

Assignments/Activities towards Comprehensive Continuous Evaluation (CCE)

Module 1:

1. Basic Linux Commands Proficiency:
   - Practice using a range of basic Linux commands for file and directory management, navigation, and system operations.
   - Apply these commands to perform tasks such as file creation, modification, deletion, and directory navigation.
2. Analysis of Linux Architecture:
   - Analyze the architecture of Unix/Linux systems, including its components and functionalities.
   - Evaluate the significance of key features like multi-user capability, security models, and file system structure in Unix/Linux.
3. Evaluation of Linux Utilities:
   - Evaluate Linux utilities for file processing operations, including creation, deletion, copying, and searching files and directories.
   - Analyze the efficiency and effectiveness of these utilities in real-world scenarios.
4. Design and Demonstrate Memory Management Techniques:
   - Design a system for managing file memory, including techniques like file and region locking mechanisms.
   - Implement and demonstrate these techniques using Linux system calls and utilities.

Module 2:

1. Application of Filters:
   - Apply filters in Linux to manipulate data effectively, including simple filters and those using regular expressions.
   - Use commands like grep for text pattern searching and filtering.
2. Analysis of Linux Editors:
   - Analyze different editors available in Linux, focusing on the vi editor.
   - Explore and practice all essential commands and functionalities of the vi editor for text editing and manipulation.
3. Design and Implementation of Shell Scripts:
   - Design shell scripts capable of performing complex tasks in a shell programming environment.
   - Develop scripts that automate file management, data processing, or system administration tasks.
4. Evaluation of Linux Kernel Functionalities:
   - Evaluate the role and functionalities of the Linux kernel in system operations, including process management, memory management, and device management.
   - Discuss the advantages and disadvantages of kernel-level operations and their impact on system performance.

References:-

1. Card, R., Dumas, E., & Mevel, F. (2003). The Linux kernel book. Wiley.
2. Das, S. (2017). Unix concepts and applications (4th ed.). TMH.
3. Suehring, S. (2002). MySQL Bible. John Wiley & Sons.
4. Lerdorf, R., & Tatroe, K. (2002). Programming PHP. O'Reilly Publications.
5. Collings, T., & Wall, K. (Wiley). (2003). Red Hat Linux network and system administration (3rd ed.).
6. Mathews, N. (2007). Beginning Linux programming (4th ed.). Wrox Press.
7. Koparkar, P. (2001). Unix for you. Tata McGraw-Hill.
8. Kanetkar, Y. P. (2013). Unix shell programming (1st ed.). BPB Publications.

Semester: II

2.6 SEC

Course Title	**VEDIC MATHEMATICS**
Course Credits	**2**
Course Outcomes	**After completion of this Course, the students will be able to**
	• Apply Ancient Vedic Mathematics techniques to solve mathematical problems efficiently.
	• Analyze the principles behind Faster Calculation Methods for improved computation speed.
	• Evaluate the effectiveness of Ancient Vedic Mathematics techniques and Faster Calculation Methods in various problem-solving scenarios.
	• Design strategies for learning and practicing Ancient Vedic Mathematics techniques and Faster Calculation Methods to enhance mathematical proficiency.
Module1 (Credit1)	
Learning Outcomes	**After learning the Module, learners will be able to**
	• Apply the understanding of the differences between general mathematics and Vedic mathematics in problem-solving.
	• Analyze techniques for rapidly adding single, double, and triple digits using Vedic mathematics methods.
	• Evaluate the efficiency of Vedic mathematics in performing rapid addition tasks compared to traditional methods.
	• Design practice exercises to learn and master rapid addition techniques for single, double, and triple digits using Vedic mathematics principles.
Content Outline	• Introduction of Vedic Maths • Benefits of Vedic Maths • Difference between general Maths and Vedic Maths • Mental Maths Addition • Rapid Addition-Single to Double-Digit • Rapid Addition-Double to Double-Digit • Rapid Addition-Triple to Triple-Digit • Left to Right Addition
Module2 (Credit1)	
Learning Outcomes	**After learning the Module, learners will be able to**
	• Apply multiplication techniques to solve mathematical problems efficiently.
	• Analyze various multiplication tricks to understand their underlying principles.
	• Evaluate the effectiveness of different multiplication tricks in solving multiplication problems quickly.
	• Design practice exercises to master different multiplication tricks and enhance multiplication skills.
Content Outline	• Multiplication with Double Digit to Single Digit numbers • Multiplication by Multiples of 10 • Traditional Multiplication • Multiplication with Tricks

Assignments/Activities to wards Comprehensive Continuous Evaluation (CCE)

Module 1 :

Create a series of practice exercises and a summary report on the benefits and differences between Vedic mathematics and traditional mathematics in rapid addition.

1. Develop a set of practice exercises for rapid addition using Vedic mathematics principles, covering single-digit, double-digit, and triple-digit addition problems.
2. Provide step-by-step explanations for each rapid addition technique, including mental maths addition and left-to-right addition.
3. Analyze and compare the efficiency of Vedic mathematics rapid addition techniques with traditional methods, highlighting the differences and benefits of Vedic maths.
4. Include examples and case studies to illustrate the application of Vedic mathematics in real-world scenarios.
5. Design a summary report summarizing the key concepts, benefits, and differences between general mathematics and Vedic mathematics in problem-solving.

Module 2:

Develop a tutorial and practice exercises focusing on various multiplication techniques derived from Vedic mathematics principles.

1. Create a tutorial explaining multiplication techniques for multiplying double-digit numbers with single-digit numbers, multiplying by multiples of 10, and traditional multiplication methods.
2. Include step-by-step instructions for each multiplication technique, demonstrating how to apply Vedic mathematics principles to solve multiplication problems efficiently.
3. Analyze the effectiveness of different multiplication tricks in terms of speed and accuracy, comparing them to traditional multiplication methods.
4. Develop a series of practice exercises covering different multiplication techniques, allowing learners to practice and master each method.
5. Provide solutions and explanations for the practice exercises, ensuring learners understand the underlying principles behind each multiplication trick.

References:

1. Tirtha, B. K. (2012). Mental Calculation. Sheth Publishing House.
2. Tirtha, B. K. (2013). Vedic Mathematics. Motilal Banarsidass Publishers.

Semester: II

2.7 AEC

Semester: II

2.8 VEC

Semester: II
2.9 CC (Co-Curricular Course-II)