VIDYASAGARUNIVERSITY

Midnapore, West Bengal

PROPOSED CURRICULUM&SYLLABUS (DRAFT) OF

BACHELOR OF SCIENCE (HONOURS)
MAJOR IN COMPUTER SCIENCE

4-YEAR UNDERGRADUATE PROGRAMME

(w.e.f. Academic Year 2023-2024)

Based on

Curriculum & Credit Framework for Undergraduate Programmes

(CCFUP), 2023& NEP, 2020

VIDYASAGAR UNIVERSITY
BACHELOR OF SCIENCE (HONOURS) MAJOR IN COMPUTER SCIENCE
(under CCFUP, 2023)

Level	YR.	SEM	Course Type	Course Code	Course Title	Credit	L-T-P	Marks		
								CA	ESE	TOTAL
B.Sc. (Hons.)	2nd	III			**SEMESTER-III**					
			Major-3	**COSHMJ03**	**T: Data Structure; P: Practical**	**4**	**3-0-1**	**15**	**60**	**75**
			Major-4	**COSHMJ04**	**T: Computer Architecture; P: Practical**	**4**	**3-0-1**	**15**	**60**	**75**
			SEC	**COSSEC03**	**P: PYTHON**	**3**	**0-0-3**	**10**	**40**	**50**
			AEC	**AEC03**	Communicative English -2 (***common for all programmes***)	**2**	**2-0-0**	**10**	**40**	**50**
			MDC	**MDC03**	Multidisciplinary Course -3 (***to be chosen from the list***)	**3**	**3-0-0**	**10**	**40**	**50**
			Minor-3 (Disc.-I)	**COSMIN03**	**T: Digital Logic**	**4**	**3-1-0**	**15**	**60**	**75**
					Semester-III Total	**20**				**375**
		IV			**SEMESTER-II**					
			Major-5	**COSHMJ05**	**T: OOPs using C++; P: Practical**	**4**	**3-0-1**	**15**	**60**	**75**
			Major-6	**COSHMJ06**	**T: Operating System; P: Practical**	**4**	**3-0-1**	**15**	**60**	**75**
			Major-7	**COSHMJ07**	**T: Computer Network; P: Practical**	**4**	**3-0-1**	**15**	**60**	**75**
			AEC	**AEC04**	MIL-2 (***common for all programmes***)	**2**	**2-0-0**	**10**	**40**	**50**
			Minor-4 (Disc.-II)	**COSMNI04**	**T: Data Structure; P: Practical**	**4**	**3-0-1**	**15**	**60**	**75**
			Summer Intern.	**INT**	**Internship/ Apprenticeship - activities to be decided by the Colleges following the guidelines to be given later**	**4**	**0-0-4**	**-**	**-**	**50**
					Semester-IV Total	**24**				**400**
					TOTAL of YEAR-2	**44**				**775**

MJ = Major, MI = Minor Course, SEC = Skill Enhancement Course, AEC = Ability Enhancement Course, MDC = Multidisciplinary Course, CA= Continuous Assessment, ESE= End Semester Examination, T = Theory, P= Practical, L-T-P = Lecture-Tutorial-Practical, MIL = Modern Indian Language

MAJOR (MJ)

MJ-3: Data Structure **Credits 04(Full Marks: 75)**

OBJECTIVE OF THE COURSE

- Introduce fundamental concepts and importance of data structures in computing.
- Teach implementation of linear data structures such as arrays, linked lists, stacks, and queues.
- Explore non-linear data structures including trees, graphs, and heaps.
- Emphasize analyzing algorithm efficiency in terms of time and space complexity.
- Develop skills in designing and implementing efficient algorithms.
- Demonstrate real-world applications of data structures in software development for context and relevance.
- Provide hands-on programming experience with languages like C or Python.
- Enhance analytical and problem-solving skills through practical assignments.
- Prepare students for advanced topics in computer science and software engineering by building a strong foundation in complex data structures.

MJ-3T: Data Structure **Credits 03**

Course contents:

Module- I Arrays **05 Hrs.**
Single and Multi-dimensional Arrays, Sparse Matrices (Array and Linked Representation)

Module- II Stacks **05 Hrs.**
Implementing single / multiple stack/s in an Array; Prefix, Infix and Postfix expressions, Utility and conversion of these expressions from one to another; Applications of stack; Limitations of Array representation of stack

Module- III Linked Lists **10 Hrs.**
Singly, Doubly and Circular Lists (Array and Linked representation); Normal and Circular representation of Stack in Lists; Self Organizing Lists; Skip Lists

Module- IV Queues **05 Hrs.**
Array and Linked representation of Queue, De-queue, Priority Queues

Module- V Recursion **05 Hrs.**
Developing Recursive Definition of Simple Problems and their implementation; Advantages and Limitations of Recursion; Understanding what goes behind Recursion (Internal Stack Implementation)

Module- VI Trees **20 Hrs.**
Introduction to Tree as a data structure; Binary Trees (Insertion, Deletion , Recursive and Iterative Traversals on Binary Search Trees); Threaded Binary Trees (Insertion, Deletion, Traversals); Height-Balanced Trees (Various operations on AVL Trees). Tree traversal techniques.

Module- VII Searching and Sorting **05 Hrs.**
Linear Search, Binary Search, Comparison of Linear and Binary Search, Selection Sort, Insertion Sort, Bubble Sort, Quick Sort, Comparison of Sorting Techniques

Module- VIII Hashing **05 Hrs.**
Introduction to Hashing, Efficiency of Rehash Methods, Resolving collision by Open Addressing, Coalesced Hashing, Separate Chaining, Dynamic and Extendible Hashing.

Suggested Readings:

1. Adam Drozdek, "Data Structures and algorithm in C++", Third Edition, Cengage Learning, 2012.
2. SartajSahni, Data Structures, "Algorithms and applications in C++", Second Edition, Universities Press, 2011.
3. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using C and C++:, Second edition, PHI, 2009.
4. Robert L. Kruse, "Data Structures and Program Design in C++", Pearson, 1999.
5. D.S Malik, Data Structure using C++, Second edition, Cengage Learning, 2010.
6. Mark Allen Weiss, ″Data Structures and Algorithms Analysis in Java", Pearson Education, 3rd edition, 2011
7. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using Java, 2003.
8. Robert Lafore, "Data Structures and Algorithms in Java, 2/E", Pearson/ Macmillan Computer Pub, 2003
9. John Hubbard, "Data Structures with JAVA", McGraw Hill Education (India) Private Limited; 2 edition, 2009
10. Goodrich, M. and Tamassia, R. "Data Structures and Algorithms Analysis in Java", 4th Edition, Wiley, 2013
11. Herbert Schildt, ″Java The Complete Reference (English) 9th Edition Paperback", Tata McGraw Hill, 2014.
12. D. S. Malik, P.S. Nair, "Data Structures Using Java", Course Technology, 2003.

MJ-3P: Data Structures La **Credits 01**

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2. WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists (include a function and also overload operator +).
4. Implement Doubly Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
5. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation. Use Templates.
8. Perform Queues operations using Circular Array implementation. Use Templates.
9. Create and perform different operations on Double-ended Queues using Linked List implementation.
10. WAP to scan a polynomial using linked list and add two polynomial.

11. WAP to calculate factorial and to compute the factors of a given no. (i) using recursion, (ii) using iteration
12. WAP to display fibonacci series (i)using recursion, (ii) using iteration
13. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion
14. WAP to create a Binary Search Tree and include following operations in tree:
    (a) Insertion (Recursive and Iterative Implementation)
    (b) Deletion by copying
    (c) Deletion by Merging
    (d) Search a no. in BST
    (e) Display its preorder, postorder and inorder traversals Recursively
    (f) Display its preorder, postorder and inorder traversals Iteratively
    (g) Display its level-by-level traversals
    (h) Count the non-leaf nodes and leaf nodes
    (i) Display height of tree
    (j) Create a mirror image of tree
    (k) Check whether two BSTs are equal or not
15. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
16. WAP to reverse the order of the elements in the stack using additional stack.
17. WAP to reverse the order of the elements in the stack using additional Queue.
18. WAP to implement Diagonal Matrix using one-dimensional array.
19. WAP to implement Lower Triangular Matrix using one-dimensional array.
20. WAP to implement Upper Triangular Matrix using one-dimensional array.
21. WAP to implement Symmetric Matrix using one-dimensional array.
22. WAP to create a Threaded Binary Tree as per in order traversal, and implement operations like finding the successor / predecessor of an element, insert an element, in order traversal.
23. WAP to implement various operations (searching, insertion, deletion) on AVL Tree.
24. Implementation of Selection Sort, Insertion Sort, Bubble Sort, Quick Sort.

MJ-4: Computer Architecture Credits 04

OBJECTIVE OF THE COURSE

- Provide a comprehensive understanding of computer system design and functionality.
- Cover fundamental concepts of computer organization, including processor architecture, memory hierarchies, and input/output systems.
- Explore instruction set architectures (ISA) and their impact on hardware performance and efficiency.
- Teach about data paths, control units, and pipelining techniques to enhance processing speed.
- Delve into parallel processing and multi-core architectures to illustrate advancements in modern computing.
- Develop practical skills through laboratory sessions and programming assignments involving assembly language and hardware simulation tools.
- Emphasize the impact of architectural decisions on software development and system performance.
- Prepare students to understand, analyze, and optimize the architecture of contemporary computer systems.
- Lay the groundwork for further study in advanced computer engineering topics.

MJ-4T: Computer Architecture **Credits 03**

Module I: Introduction **20 Hrs.**
Logic gates, Boolean algebra, combinational circuits, circuit simplification, flip-flops and sequential circuits, decoders, multiplexers, registers, counters and memory units.

Module II: Data Representation and Basic Computer Arithmetic **10 Hrs.**
Number systems, complements, fixed and floating point representation, character representation, addition, subtraction, magnitude comparison, multiplication and division algorithms for integers

Module III: Basic Computer Organization and Design **8 Hrs.**
Computer registers, bus system, instruction set, timing and control, instruction cycle, memory reference, Organization of a basic single-bus computer system.

Module IV: Central Processing Unit **10 Hrs.**
Register organization, arithmetic and logical operations, Instruction formats, addressing modes, instruction codes, machine language, assembly language, RISC, CISC architectures, Hardwired and micro programmed control unit design.

Module V: Memory Organization **6 Hrs.**
Memory interfacing and addressing, cache memory organization.

Module VI: Input-Output Organization **6 Hrs.**
Input / Output: External Devices, I/O Modules, Programmed I/O, Interrupt-Driven I/O, Direct Memory Access, I/O Channels.

Suggested Readings:

1. M. Mano, Computer System Architecture, Pearson Education 1992
2. W. Stallings, Computer Organization and Architecture Designing for Performance, 8 Edition, Prentice Hall of India,2009
3. M.M. Mano , Digital Design, Pearson Education Asia,2013
4. Carl Hamacher, Computer Organization, Fifth edition, McGrawHill, 2012.

MJ-4P: Computer Architecture Lab **Credits 01**

All laboratory assignments are based on Hardware Description Language (VHDL or Verilog) Simulation / Hardware Kit.
[Pre-requisite: The hardware based design has been done in the Analog& Digital Electronics laboratory and Computer Organization laboratory]

1. HDL introduction
2. Basic digital logic base programming with HDL
3. 8-bit Addition, Multiplication, Division
4. 8-bit Register design
5. Memory unit design and perform memory operatons.
6. 8-bit simple ALU design
7. 8-bit simple CPU design

8. Interfacing of CPU and Memory
9. Create the micro operations and associate with instructions as given in the chapter (except interrupts). Design the register set, memory and the instruction set. Use this machine for the assignments of this section.
10. Create a Fetch routine of the instruction cycle.
11. Simulate the machine to determine the contents of AC, E, PC, AR and IR registers in hexadecimal after the execution of each of following register reference instructions:

a. CLA	e. CIR	i. SNA
b. CLE	f. CIL	j. SZA
c. CMA	g. INC	k. SZE
d. CME	h. SPA	l. HLT

Initialize the contents of AC to (A937)16, that of PC to (022)16 and E to 1.
12. Simulate the machine for the following memory-reference instructions with I= 0 and address part = 082. The instruction to be stored at address 022 in RAM. Initialize the memory word at address 082 with the operand B8F2 and AC with A937. Determine the contents of AC, DR, PC, AR and IR in hexadecimal after the execution.

a. ADD	f. BSA
b. AND	g. ISZ
c. LDA	d. STA
e. BUN	

13. Simulate the machine for the memory-reference instructions referred in above question with I= 1 and address part = 082. The instruction to be stored at address 026 in RAM. Initialize the memory word at address 082 with the value 298. Initialize the memory word at address 298 with operand B8F2 and AC with A937. Determine the contents of AC, DR, PC, AR and IR in hexadecimal after the execution.
14. Modify the machine created in Practical 1 according to the following instruction format:
Instruction format

0	**2 3**	**4**	**15**
Opcode	**I**	**Address**	

a. The instruction format contains a 3-bit opcode, a 1-bit addressing mode and a 12-bit address. There are only two addressing modes, I = 0 (direct addressing) and I = 1 (indirect addressing).
b. Create a new register I of 1 bit.
c. Create two new microinstructions as follows:
i. Check the opcode of instruction to determine type of instruction (Memory Reference/Register Reference/Input-Output) and then jump accordingly.
ii. Check the I bit to determine the addressing mode and then jump accordingly.

MJ-5: OOPs using C++ Credits 04

OBJECTIVE OF THE COURSE

- Provide a deep understanding of object-oriented programming principles and their application using C++.
- Cover fundamental concepts such as classes, objects, inheritance, polymorphism, and encapsulation.
- Teach students to design and implement reusable and modular code using OOP principles.
- Emphasize the creation and manipulation of complex data structures through dynamic memory management and operator overloading.
- Develop practical programming skills through hands-on projects and assignments involving real-world applications.
- Introduce advanced C++ features such as templates and the Standard Template Library (STL) for efficient coding practices.
- Provide experience in debugging and testing C++ applications to ensure reliability and performance.
- Equip students with the skills to develop robust, efficient, and maintainable software using OOP techniques in C++.
- Prepare students for advanced programming and software development roles.

MJ-5T: OOPs using C++ Credits 03

Module-I: Introduction to OOPs and C++ Element 15 Hrs.
Structured vs. Object Oriented Programming, Object Oriented Programming Concepts, Benefits of Object oriented programming, Object Oriented Languages, Structure of a C++ program, Data Types, Operators and Control Structures, Iteration / Loop Construct, Arrays, Functions (User defined Function, Inline Function, Function Overloading), User Defined Data Types (Structure, Union and Enumeration).

Module II: Class, Object, Constructor & Destructor: 15 Hrs.
Defining Classes, Encapsulation, Instantiating Objects, Member Functions, Accessibility labels, Static Members, Friend Function, Purpose of Constructors, Default Constructor, Parameterized Constructors, Copy Constructor, Destructor.

Module III: Pointer, Polymorphism & Inheritance: 20 Hrs.
Pointer (Pointer to Object, this Pointer, Pointer to Derive Class), Introduction to Polymorphism (Compile time Polymorphism, Run time Polymorphism), Operator Overloading, Overloading Unary and Binary Operators, Virtual Function, Pure Virtual Functions, Inheritance (Single Inheritance, Multiple Inheritance, Multilevel Inheritance, Hierarchical Inheritance, Hybrid Inheritance), Virtual Base Class, Abstract Class.

Module IV: Exception Handling: 10 Hrs.
Exceptions in C++ Programs, Try and Catch Expressions, Exceptions with arguments

Suggested Readings:

1. HerbtzSchildt, "C++: The Complete Reference", Fourth Edition, McGraw Hill.2003
2. BjarneStroustrup, "The C++ Programming Language", 4th Edition, Addison-Wesley ,2013.
3. BjarneStroustrup, "Programming -- Principles and Practice using C++", 2nd Edition, Addison-Wesley 2014.
4. E Balaguruswamy, "Object Oriented Programming with C++", Tata McGraw-Hill Education, 2008.
5. Paul Deitel, Harvey Deitel, "C++ How to Program", 8th Edition, Prentice Hall, 2011.
6. John R. Hubbard, "Programming with C++", Schaum's Series, 2nd Edition, 2000.
7. Andrew Koeni, Barbara, E. Moo, "Accelerated C++", Published by Addison-Wesley ,
1. 2000. 7. Scott Meyers, "Effective C++", 3rd Edition, Published by Addison-Wesley,
2. 2005.
8. Harry, H. Chaudhary, "Head First C++ Programming: The Definitive Beginner's Guide",
3. First Create space Inc, O-D Publishing, LLC USA.2014
9. Walter Savitch, "Problem Solving with C++", Pearson Education, 2007.
10. Stanley B. Lippman, JoseeLajoie, Barbara E. Moo, "C++ Primer", Published by
4. Addison-Wesley, 5th Edition, 2012
11. E Balagurusamy , Object Oriented Programming with C++, 5 th edition, Tata McGraw, 2011.
12. Deitel and Deitel , “C++: How to Program”, 9th Edition, Pearson, 2013.

MJ-5P: OOPs using C++ (Lab) **Credits 01**

1. Write a C++ program to find the sum of individual digits of a positive integer.
2. Write a C++ program to print the given number in reverse order.
3. Write a C++ program to print first 100 non-Fibonacci numbers.
4. Write a C++ program to convert a decimal number into a hexadecimal number.
5. Write a C++ program to search an element of an array using binary search technique.
6. Write a C++ program to calculate compound interest in a bank using default arguments.
7. Write a C++ program to display the student details using classes and object as array.
8. Write a C++ program to implement stack using array.
9. Write a C++ program for matrix multiplication using dynamic memory allocation, copy construction and overloading of assignment operator.
10. Write a C++ program to read a two dimensional matrix and display its transpose.
11. Write a C++ program to implement inline function.
12. Write a C++ program to implement constructor and destructor.
13. Write a C++ program to implement the functionalities of a copy constructor.
14. Write a C++ program to display the account number and balance using constructor overloading.
15. Write a C++ program to find the volume of cube, rectangle and cylinder using function overloading. 16. Write a C++ program to overload operator ++ and operator - using friend functions.
16. Write a C++ program to add two complex numbers using binary operator overloading.
17. Write a C++ program to implement single inheritance and multilevel inheritance.
18. Write a C++ program to draw a rectangle, square and circle using multiple inheritance with virtual function.
19. Write a C++ program to implement hybrid inheritance.
20. Write a C++ program to display student details using virtual base class.

21. Write a C++ program to implement pure virtual function.

Reference Books:

1. E. Balaguruswami-Object Oriented programming with C++
2. Kris James-Success with C++
3. David Parsons-Object Oriented programming with C++
4. D. Ravichandran-Programming in C++
5. Dewhurst and Stark-Programming in C++

MJ-6: Operating System Credits 04

OBJECTIVE OF THE COURSE

- Provide a comprehensive understanding of fundamental concepts and functions of modern operating systems.
- Cover the architecture and components of operating systems, including process management, memory management, file systems, and input/output systems.
- Teach about concurrency, process synchronization, and inter-process communication for efficient task management.
- Emphasize the role of operating systems in resource allocation and system security.
- Offer hands-on experience in implementing and configuring operating system features through labs and projects.
- Explore various operating systems like Windows, Linux, and macOS to understand their differences and commonalities.
- Delve into virtualization and distributed systems to highlight current trends in operating system design.
- Equip students with the skills to analyze, design, and optimize operating systems, preparing them for advanced study and careers in systems programming and software engineering.

MJ-6T: Operating System Credits 03

Module I: Introduction **10 Hrs.**

Basic OS functions, resource abstraction, types of operating systems–multiprogramming systems, batch systems , time sharing systems; operating systems for personal computers & workstations, process control & real time systems.

Case study on Linux system **6 Hrs.**

- Cloud computing (3 lectures)
- Linux evolution and Linux distros (2 lectures)
- Linux file system (1 lecture)

Module II: Operating System Organization **6 Hrs.**

Processor and user modes, kernels, system calls and system programs.

Module III: Process Management **16 Hrs.**
System view of the process and resources, process abstraction, process hierarchy, threads, threading issues, thread libraries; Process Scheduling, non-pre-emptive and pre-emptive scheduling algorithms; concurrent processes, critical section, semaphores, methods for inter- process communication; deadlocks.

Module IV: Memory Management **10 Hrs.**
Physical and virtual address space; memory allocation strategies – fixed and variable partitions, paging, segmentation, virtual memory

Module V: File and I/O Management **8 Hrs.**
Directory structure, file operations, file allocation methods, device management.

Module VI: Protection and Security **4 Hrs.**
Policy mechanism, Authentication, Internal access Authorization.

Suggested Readings:

1. A Silberschatz, P.B. Galvin, G. Gagne, Operating Systems Concepts, 8th Edition, John Wiley Publications 2008.
2. A.S. Tanenbaum, Modern Operating Systems, 3rd Edition, Pearson Education 2007.
3. G. Nutt, Operating Systems: A Modern Perspective, 2nd Edition Pearson Education 1997.
4. W. Stallings, Operating Systems, Internals & Design Principles, 5th Edition, Prentice Hall of India. 2008.
5. M. Milenkovic, Operating Systems- Concepts and design, Tata McGraw Hill 1992.

MJ-6P: Operating System Lab **Credits 01**

1. Write a program (using *fork ()* and/or *exec ()* commands) where parent and child execute:
    a. same program, same code.
    b. same program, different code.
    c. before terminating, the parent waits for the child to finish its task.
2. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)
3. Write a program to report behaviour of Linux kernel including information on configured memory, amount of free and used memory (memory information).
4. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.
5. Write a program to copy files using system calls.
6. Write program to implement FCFS scheduling algorithm.
7. Write program to implement Round Robin scheduling algorithm.
8. Write program to implement SJF scheduling algorithm.
9. Write program to calculate sum of n numbers using *thread* library.
10. Write a program to implement first-fit, best-fit and worst-fit allocation strategies

Reference Books:

1. E. Balaguruswami-Object Oriented programming with C++
2. Kris James-Success with C++
3. David Parsons-Object Oriented programming with C++
4. D. Ravichandran-Programming in C++
5. Dewhurst and Stark-Programming in C++

MJ-7: Computer Network Credits 04

OBJECTIVE OF THE COURSE

- Provide a foundational understanding of network architectures, protocols, and technologies.
- Cover essential concepts such as the OSI and TCP/IP models, and data transmission across networks.
- Teach about network devices, including routers, switches, and hubs, and their roles in communication.
- Delve into protocols like HTTP, FTP, TCP, and UDP, illustrating their functions in data exchange.
- Emphasize network design, addressing, and subnetting for configuring and managing networks.
- Offer hands-on experience with network configuration and troubleshooting through practical labs and projects.
- Explore wireless networks, security practices, and emerging technologies like cloud computing and the Internet of Things (IoT).
- Prepare students to analyze, design, and implement network solutions, laying the groundwork for careers in network administration, cybersecurity, and IT infrastructure.

MJ-7T: Computer Network Credits 03

Module I: Introduction to Computer Networks 8 Hrs.
Network definition; network topologies; network classifications; network protocol; layered network architecture; overview of OSI reference model; overview of TCP/IP protocol suite.

Module II: Data Communication Fundamentals and Techniques 10 Hrs.
Analog and digital signal; data-rate limits; digital to digital line encoding schemes; pulse code modulation; parallel and serial transmission; digital to analog modulation-; multiplexing techniques- FDM, TDM; transmission media.

Module III: Networks Switching Techniques and Access mechanisms 10 Hrs.
Circuit switching; packets witching- connectionless datagram switching, connection-oriented virtual circuit switching; dial-up modems; digital subscriber line; cable TV for data transfer.

Module IV: Data Link Layer Functions and Protocol **10 Hrs.**
Error detection and error correction techniques; data-link control- framing and flow control; error recovery protocols- stop and wait ARQ, go-back-n ARQ; Point to Point Protocol on Internet.

Module V: Multiple Access Protocol and Networks **5 Hrs.**
CSMA/CD protocols; Ethernet LANS; connecting LAN and back-bone networks- repeaters, hubs, switches, bridges, router and gateways;

Module VI: Networks Layer Functions and Protocols **6 Hrs.**
Routing; routing algorithms; network layer protocol of Internet- IP protocol, Internet control protocols.

Module VII: Transport Layer Functions and Protocols **6 Hrs.**
Transport services- error and flow control, Connection establishment and release – three way handshake;

Module VIII: Overview of Application layer protocol 5 Hrs.
Overview of DNS protocol; overview of WWW &HTTP protocol.

Suggested Readings:

1. B. A. Forouzan: Data Communications and Networking, Fourth edition, THM, 2007.
2. A. S. Tanenbaum: Computer Networks**,** Fourth edition**,** PHI, 2002.

MJ-7P: Computer Network **Credits 01**

Use C++/ Python/ Java:

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.
3. Simulate and implement go back n sliding window protocol.
4. Simulate and implement selective repeat sliding window protocol.
5. Simulate and implement distance vector routing algorithm
6. Simulate and implement Dijkstra algorithm for shortest path routing.
7. Experiments for capturing and analyzing data packets using Wire Shark.
    a. Experiments on filtering packets
    b. Experiments on inspecting packets
8. Write a program for a HLDC frame to perform the following. i. Bit stuffing , ii. Character stuffing.
9. Write a program for distance vector algorithm to find suitable path for transmission.
10. Implement Dijkstra 's algorithm to compute the shortest routing path.
11. For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the program for the cases
    a Without error
    b. With error
12. Implementation of Stop and Wait Protocol and Sliding Window Protocol
13. Write a program for congestion control using leaky bucket algorithm.

Use NS2/NS3.

1. Write TCL Script for connecting two nodes and sending packets in wired network.

2. Write TCL Script for given STAR topology using SFQ on queue at intermediate node & use different colors for packet originated from different nodes.
3. Write TCL Script for given RING topology in wired network using For loop & making topology dynamic.
4. Write TCL Script in wired network for the given topology using TCP connection and sending data through the node.
5. Write TCL Script in wired network for the given topology using UDP connection and sending data through node.
6. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.
7. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
8. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination
9. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.
10. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
11. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
12. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
13. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
14. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment

Reference Books:

6. E. Balaguruswami-Object Oriented programming with C++
7. Kris James-Success with C++
8. David Parsons-Object Oriented programming with C++
9. D. Ravichandran-Programming in C++
10. Dewhurst and Stark-Programming in C++

MINOR (MI)

MI – 3: Digital Logic **Credits 04(Full Marks: 75)**

OBJECTIVE OF THE COURSE

- Learn the basics of binary, octal, decimal, and hexadecimal number systems and conversions between them.
- Master the principles of Boolean algebra, including logic operations, truth tables, and De Morgan's laws.
- Gain proficiency in simplifying Boolean expressions using techniques such as Karnaugh maps and the Quine-McCluskey method.
- Develop skills in designing basic combinational logic circuits, including adders, subtractors, multiplexers, and decoders.
- Understand the behaviour and design of sequential circuits, including flip-flops, latches, counters, and registers.
- Explore circuit minimization techniques to reduce complexity and cost in digital designs.

OUTCOME OF THE COURSE

- Gain a clear understanding of binary, octal, and hexadecimal number systems and their applications in digital circuits.
- Learn and apply Boolean algebra principles to simplify and analyse logical expressions.
- Develop skills in designing and implementing combinational and sequential logic circuits like multiplexers, decoders, and flip-flops.
- Use Karnaugh maps and other techniques to simplify logic functions and reduce the complexity of circuits.
- Learn the operation, behaviour, and application of basic memory elements like SR latches, JK, D, and T flip-flops.
- Develop the ability to analyse and design sequential circuits, including counters and shift registers.

MI – 3T: Digital Logic **Credits 04**

Course contents:

Number systems: **15 Hrs.**
Positional number systems; Binary, Octal, Hexadecimal, and Decimal number systems; conversion of a number in one system to the other; Representation of signed numbers-signed magnitude, one's complement, 2's complement representation techniques, Merits of 2's complement representation scheme; Various binary codes - BCD, excess -3, Gray code, ASCII, EBCDIC; Binary arithmetic- addition, subtraction, multiplication, and division of unsigned binary numbers.

Boolean algebra: **15 Hrs.**
Fundamental of Boolean Expression: Definition of Boolean Algebra, Postulates, Basic Logic gates: (OR, AND, NOT); Universal Logic Gates: (NAND & NOR); Basic logic operations: logical sum (OR), logical product (AND), complementation (NOT), anti-coincidence (EX-OR) and coincidence (EX-NOR) operations: Truth tables of Basic gates; Boolean Variables and Expressions; De-Morgan’s theorem; Boolean expressions Simplification- Algebraic technique, Karnaugh map technique, 3 variable and 4 variable Karnaugh map.

Combinational Circuits: **15 Hrs.**
Half Adder, Full Adder (3-bit), Half Subtractor, Full Subtractor (3-bit), and construction using Basic Logic Gates (OR, AND, NOT) and Universal Logic Gates (NAND & NOR), Multiplexer, Encoders, Demultiplexer, and Decoder circuits.

Sequential Circuits: **15 Hrs.**
Latch, RS, D, JK, T Flip Flops; Race condition, Master Slave JK Flip Flop; Registers: Serial Input Serial Output (SISO), Serial Input Parallel Output (SIPO), Parallel input Serial Output (PISO), Parallel Input Parallel Output (PIPO), Universal Shift Registers; Counters: Asynchronous Counter, Synchronous Counter.

Suggested Readings:

1. Morris Mano, Charles R. Kime, Logic and computer design fundamentals, Pearson Prentice Hall, 2004
2. Basavaraj,B., Digital fundamentals, New Delhi: Vikas Publishing House, 1999.
3. Kandel Langholz, Digital Logic Design, Prentice Hall, 1988.
4. Rafiquzzaman & Chandra, Modern Computer Architecture, West Pub. Comp., 1988.

MI-4: Data Structure Credits 04(Full Marks: 75)

OBJECTIVE OF THE COURSE

- Introduce students to fundamental concepts of data structures and algorithms, including the importance of data organization and management in solving computational problems efficiently.
- Teach students how to design, analyse, and implement algorithms for various data structures. This includes understanding algorithm complexity (Big O notation) and the trade-offs between time and space efficiency.
- Provide hands-on experience with implementing data structures such as arrays, linked lists, stacks, queues, trees, heaps, graphs, and hash tables using a programming language.
- Show how to apply data structures to solve real-world problems. This involves understanding which data structure is appropriate for a given situation and how to manipulate data structures to optimize performance.
- Develop problem-solving skills by practicing the design and implementation of algorithms using appropriate data structures, focusing on improving solutions' efficiency.
- Teach students how data structures interact with memory management, including dynamic memory allocation, pointers, and garbage collection.
- Lay the foundation for more advanced topics in computer science, such as algorithms, databases, artificial intelligence, and machine learning, where a strong understanding of data structures is essential.
- Encourage teamwork through group projects and assignments, fostering collaboration and communication skills crucial in software development.

OUTCOME OF THE COURSE

- Implement various data structures such as arrays, linked lists, stacks, queues, trees, graphs, heaps, and hash tables in a programming language of their choice.
- Develop the ability to design and analyse algorithms, understand their time and space complexities using Big O notation, and make informed decisions about the most efficient algorithms to use in different scenarios.
- Enhance their problem-solving skills by applying data structures and algorithms to solve computational problems effectively and efficiently.
- Learn how to choose the appropriate data structures for specific applications, optimizing data storage, retrieval, and manipulation in software systems.
- Gain a deep understanding of how data structures interact with memory, including concepts like dynamic memory allocation, pointers, and memory deallocation.
- Have a solid foundation for advanced topics in computer science, such as algorithms, machine learning, artificial intelligence, databases, and systems design, where efficient data management is crucial.
- Evaluate the trade-offs between different data structures and algorithms in terms of time, space, and complexity, and make informed choices in their design and implementation.
- Capable of applying their knowledge of data structures in real-world software development projects, contributing to more efficient, scalable, and maintainable code.
- Improve their ability to work collaboratively in teams, effectively communicating ideas and solutions related to data structures and algorithms.
- Prepare for technical interviews in the software industry, where knowledge of data structures and algorithms is often tested through coding challenges and problem-solving questions.

MI-4T: Data Structure Credits 03

Course contents:

Arrays **3 Hrs.**
Single and Multi-dimensional Arrays, Sparse Matrices (Array and Linked Representation)

Stacks **5 Hrs.**
Implementing single/multiple stacks in an Array; Prefix, Infix, and Postfix expressions, Utility and conversion of these expressions from one to another; Applications of a stack; Limitations of Array representation of a stack

Linked Lists **7 Hrs.**
Singly, Doubly, and Circular Lists (Array and Linked representation); Normal and Circular representation of Stack in Lists; Self Organizing Lists; Skip Lists

Queues **5Hrs.**
Array and Linked representation of Queue, De-queue, and Priority Queues

Recursion **5 Hrs.**
Developing Recursive Definition of Simple Problems and their implementation; Advantages and Limitations of Recursion; Understanding what goes behind Recursion (Internal Stack Implementation)

Trees **10 Hrs.**
Introduction to Tree as a data structure; Binary Trees (Insertion, Deletion, Recursive and Iterative Traversals on Binary Search Trees); Threaded Binary Trees (Insertion, Deletion, Traversals); Height-Balanced Trees (Various operations on AVL Trees). Tree traversal techniques.

Searching and Sorting **7 Hrs.**
Linear Search, Binary Search, Comparison of Linear and Binary Search, Selection Sort, Insertion Sort, Bubble Sort, Quick Sort, Comparison of Sorting Techniques

Hashing **3 Hrs.**
Introduction to Hashing, Efficiency of Rehash Methods, Resolving collision by Open Addressing, Coalesced Hashing, Separate Chaining, Dynamic and Extendible Hashing.

Suggested Readings:

1. Adam Drozdek, "Data Structures and algorithm in C++", Third Edition, Cengage Learning, 2012.
2. Sartaj Sahni, Data Structures, "Algorithms and applications in C++", Second Edition, Universities Press, 2011.
3. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using C and C++, Second edition, PHI, 2009.
4. Robert L. Kruse, "Data Structures and Program Design in C++", Pearson, 1999.
5. D.S Malik, Data Structure using C++, Second edition, Cengage Learning, 2010.
6. Mark Allen Weiss, "Data Structures and Algorithms Analysis in Java", Pearson Education, 3rd edition, 2011
7. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using Java, 2003.
8. Robert Lafore, "Data Structures and Algorithms in Java, 2/E", Pearson/ Macmillan Computer Pub, 2003
9. John Hubbard, "Data Structures with JAVA", McGraw Hill Education (India) Private Limited; 2 edition, 2009
10. Goodrich, M. and Tamassia, R. "Data Structures and Algorithms Analysis in Java", 4th Edition, Wiley, 2013
11. Herbert Schildt, "Java The Complete Reference (English) 9th Edition Paperback", Tata McGraw Hill, 2014.
12. D. S. Malik, P.S. Nair, "Data Structures Using Java", Course Technology, 2003.

MI-4P: Data Structures Lab **Credits 01**

Course Outline:

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2. WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists (include a function and also overload operator +).
4. Implement Doubly Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.

5. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation. Use Templates.
8. Perform Queues operations using Circular Array implementation. Use Templates.
9. Create and perform different operations on Double-ended Queues using Linked List implementation.
10. WAP to scan a polynomial using a linked list and add two polynomials.
11. WAP to calculate factorial and to compute the factors of a given no. (i) using recursion, (ii) using iteration
12. WAP to display Fibonacci series (i)using recursion, (ii) using iteration
13. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion
14. WAP to create a Binary Search Tree and include following operations in tree:
    a. Insertion (Recursive and Iterative Implementation)
    b. Deletion by copying
    c. Deletion by Merging
    d. Search a no. in BST
    e. Display its preorder, post-order and in-order traversals Recursively
    f. Display its preorder, post-order and in-order traversals Iteratively
    g. Display its level-by-level traversals
    h. Count the non-leaf nodes and leaf nodes
    i. Display height of tree
    j. Create a mirror image of tree
    k. Check whether two BSTs are equal or not
15. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
16. WAP to reverse the order of the elements in the stack using an additional stack.
17. WAP to reverse the order of the elements in the stack using an additional Queue.
18. WAP to implement Diagonal Matrix using the one-dimensional array.
19. WAP to implement a Lower Triangular Matrix using the one-dimensional array.
20. WAP to implement an Upper Triangular Matrix using the one-dimensional array.
21. WAP to implement a Symmetric Matrix using the one-dimensional array.
22. WAP to create a Threaded Binary Tree as per in order traversal, and implement operations like finding the successor/predecessor of an element, inserting an element, in order traversal.
23. WAP to implement various operations on AVL Tree.

SKILL ENHANCEMENT COURSE (SEC)

SEC 3: PYTHON **Credits 03 (Full Marks: 50)**

OBJECTIVE OF THE COURSE -

The objectives of this course are to make the student understand programming language, programming, concepts of Loops, reading a set of Data, stepwise refinement, Functions, Control structure, Arrays. After completion of this course the student is expected to analyze the real-life problem and write a program in 'Python' language to solve the problem. The main emphasis of the course will be on problem solving aspect i.e., developing proper algorithms.

- After completion of the course the student will be able to
- Develop efficient algorithms for solving a problem.
- Use the various constructs of a programming language viz. conditional, iteration and recursion.
- Implement the algorithms in "Python" language.
- Use simple data structures like arrays, stacks and list in solving problems.

SEC3P: PYTHON **Credits 03**

Course Outline:

Planning the Computer Program: Concept of problem solving, Problem definition, Problem design, Debugging, Types of Errors in programing, Documentation

Techniques of Problem Solving: Flowcharting, decision table, algorithms, Structured programing concepts, Programing methodologies viz. top-down and bottom-up

Overview to Python Programming: Structure of Python Program, Elements of Python

Introduction to Python: Python Interpreter, Python shell, Indentation, Atoms, Identifiers and keywords, literals, Strings, Operator (Arithmetic Operator, Relational Operator, Logical or Boolean Operator, Assignment Operator, Ternary operator, Bitwise Operator)

Creating Python Programs: Input and Output Statements, Control Statements (Branching, Looping, Conditional Statement, Exit, Function, Difference, between break, continue and pass). Defining Functions, Default arguments and Exception handling

Iterations and Recursions: Conditional execution, Alternative execution, Nested conditionals, Return statements, Recursion, Stack diagrams for recursive functions, Multiple assignment, While statement, For statement.

String and List: String as a compound data type, Length, Traversal and the for loop, String slices, String Comparison, A find function, Looping and counting, List values, Accessing elements, List length, List membership, List and for loops, List operations, List deletion, Cloning lists, Nested Lists

Object Oriented Programing: Introduction to Classes, Objects and Methods, Standard Libraries

Suggested Readings:

1. Jhon V. Guttag, "Introduction to Computation and Programming Using Python", MIT Press
2. Allen Downey, "Think Python: How to Think a Computer Scientist", O'Reilly
3. Mark Lutz, "Learning Python, 5th Edition", O'Reilly