COEP Technological University

Department of Computer Science and Engineering

Curriculum Structure

S. Y. B. Tech. Computer Science and Engineering

(Effective from: A.Y. 2026-27)

List of Abbreviations

Abbreviation	Title
BSC	Basic Science Course
ESC	Engineering Science Course
PCC	Programme Core Course
PEC	Programme Elective Course
OE	Open Elective - other than particular program
MDM	Multidisciplinary Minor
VSEC	Vocational and Skill Enhancement Course
HSMC	Humanities Social Science and Management
AEC	Ability Enhancement Course
IKS	Indian Knowledge System
VEC	Value Education Course
RM	Research Methodology
OJT	Internship
CEA	Community Engagement Activity / Field Project
CCA	Co-curricular & Extracurricular Activities

S.Y. B. Tech.

Computer Science and Engineering

[Level 5, UG Diploma] Semester -III

Sr. No.	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme (Weightages in %)				
									Theory			Laboratory	
									MSE	TA	ESE	CIE	ESE
01	PCC	CTS-24008	Microprocessors	3	0	2	1	4	30	20	50	100	--
02	PCC	CTS-24007	Principles of Programming Languages	2	0	2	1	3	30	20	50	100	--
03	PCC	CTS-24006	Data Structures and Algorithms	3	0	2	1	4	30	20	50	100	--
04	OE	As per course	Open Elective-1	2	0	0	1	2	30	20	50	--	--
05	AEC	As per course	Indian language	2	0	0	1	2	CIE: 100			--	--
06	HSMC	HS-24001	Entrepreneurship	1	0	0	1	1	CIE: 100			--	--
07	VEC	CTS-24012	Constitution of India and Universal Human Values	1	0	0	2	1	CIE: 100			--	--
08	CEA	AS-24004	Community Engagement Activity (CEA)/Field Project	0	0	0	0	2	--			100	--
			Total	**14**	**00**	**06**	**08**	**19**					

[Level 5, UG Diploma] Semester -IV

Sr. No.	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme (Weightages in %)				
									Theory			Laboratory	
									MSE	TA	ESE	CIE	ESE
01	PCC	CTS-24010	Object Oriented Programming and Design	3	0	2	1	4	30	20	50	100	--
02	PCC	CTS-24009	Computer Organization	3	0	0	1	3	30	20	50	--	--
03	PCC	CTS-24011	Theory of Computation	3	1	0	1	4	30	20	50	--	--
04	VSEC	CTS-25006	Development Tools Laboratory	0	0	2	0	1	--			100	--
05	OE	As per course	Open Elective-2	2	0	0	1	2	30	20	50	--	--
06	MDM	As per course	Multidisciplinary Minor-1	3	0	0	1	3	30	20	50	--	--
07	VEC	AS-24002	Environmental studies	1	0	0	2	1	CIE: 100			--	--
08	HSMC	CTS-24013	Design Thinking	1	0	0	1	1	CIE: 100			--	--
09	HSMC	HS-25005	Economics	2	0	0	1	2	CIE: 100			--	--
			Total	**18**	**01**	**04**	**09**	**21**					

Legends: **L**-Lecture, **T**-Tutorial, **P**-Practical, **S**-Self Study, **Cr**-Credits, **ISE**-In-Semester-Evaluation, **ESE**-End-Semester-Evaluation, **MSE**-Mid-Semester-Evaluation, **TA**-Teachers' Assessment, **CIE**-Continuous-Internal Evaluation

**** Exit option to qualify for UG Diploma:** Two courses of 3 Credits each

S.Y. B. Tech. (Lateral Entry)
Computer Science and Engineering
[Level 5, UG Diploma] Semester -III

Sr. No.	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme (Weightages in %)				
									Theory			Laboratory	
									MSE	TA	ESE	CIE	ESE
01	PCC	CTS-24008	Microprocessors	3	0	2	1	4	30	20	50	100	--
02	PCC	CTS-24007	Principles of Programming Languages	2	0	2	1	3	30	20	50	100	--
03	PCC	CTS-24006	Data Structures and Algorithms	3	0	2	1	4	30	20	50	100	--
04	OE	As per course	Open Elective-1	2	0	0	1	2	30	20	50	--	--
05	AEC	As per course	Indian language	2	0	0	1	2	CIE: 100			--	--
06	HSMC	HS-24001	Entrepreneurship	1	0	0	1	1	CIE: 100			--	--
07	VEC	CTS-24012	Constitution of India and Universal Human Values	1	0	0	2	1	CIE: 100			--	--
08	PCC	MA-24001	Matrices, Differential Calculus and Probability	3	0	0	1	3	30	20	50	--	--
			Total	**17**	**00**	**06**	**09**	**20**					

[Level 5, UG Diploma] Semester -IV

Sr. No.	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme (Weightages in %)				
									Theory			Laboratory	
									MSE	TA	ESE	CIE	ESE
01	PCC	CTS-24010	Object Oriented Programming and Design	3	0	2	1	4	30	20	50	100	--
02	PCC	CTS-24009	Computer Organization	3	0	0	1	3	30	20	50	--	--
03	PCC	CTS-24011	Theory of Computation	3	1	0	1	4	30	20	50	--	--
07	VSEC	CTS-25006	Development Tools Laboratory	0	0	2	0	1	--			100	--
05	OE	As per course	Open Elective-2	2	0	0	1	2	30	20	50	--	--
06	MDM	As per course	Multidisciplinary Minor-1	3	0	0	1	3	30	20	50	--	--
07	VEC	AS-24002	Environmental studies	1	0	0	2	1	CIE: 100			--	--
08	HSMC	CTS-24013	Design Thinking	1	0	0	1	1	CIE: 100			--	--
09	HSMC	HS-25005	Economics	2	0	0	1	2	CIE: 100			--	--
10	IKS	HS-24007	Communication Skills	2	0	0	0	2	CIE: 100			--	--
11	PCC	CTS-24004	Discrete Structures	2	0	0	1	2	CIE: 100			--	--
			Total	**22**	**01**	**04**	**10**	**25**					

Legends: **L**-Lecture, **T**-Tutorial, **P**-Practical, **S**-Self Study, **Cr**-Credits, **ISE**-In-Semester-Evaluation, **ESE**-End-Semester-Evaluation, **MSE**-Mid-Semester-Evaluation, **TA**-Teachers' Assessment, **CIE**-Continuous-Internal Evaluation

** **Exit option to qualify for UG Diploma:** Two courses of 3 Credits each

MULTIDISCIPLINARY MINORS

SEM	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme (Weightages in %)				
									Theory			Laboratory	
									MSE	TA	ESE	CIE	ESE
IV	MDM	MDM-01	Data Structures, Files and Algorithms	3	0	0	1	3	30	20	50	--	--

OPEN ELECTIVES

SEM	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme (Weightages in %)				
									Theory			Laboratory	
									MSE	TA	ESE	CIE	ESE
III	OE	OE-01	Data Analytics	2	0	0	1	2	30	20	50	--	--
IV	OE	OE-02	Fundamentals of Operating Systems/ Fundamentals of Algorithms	2	0	0	1	2	30	20	50	--	--

HONORS

Data Science

Sem	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme				
									Theory			Laboratory	
									MSE	TA	ESE	ISE	ESE
III	HON-01		Making Sense of Data	3	0	0	1	3	30	20	50	--	--
IV	HON-02		Big Data Analytics	3	0	2	1	4	30	20	50	50	50
V	HON-03		Data Mining	3	0	2	1	4	30	20	50	50	50
VI	HON-04		Reinforcement Learning	3	0	2	1	4	30	20	50	50	50
VII	HON-05		Explainable AI	3	0	0	1	3	30	20	50	--	--
Total				**15**	**00**	**06**	**05**	**18**					

Cyber Security

Sem	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme				
									Theory			Laboratory	
									MSE	TA	ESE	ISE	ESE
III	HON-01		Fundamentals of Information and Coding Theory	3	0	0	1	3	30	20	50	--	--
IV	HON-02		Ethical Hacking	3	0	2	1	4	30	20	50	50	50
V	HON-03		Malware Analysis	3	0	2	1	4	30	20	50	50	50
VI	HON-04		IOT Security	3	0	2	1	4	30	20	50	50	50
VII	HON-05		Cyber Laws	3	0	0	1	3	30	20	50	--	--
Total				**15**	**00**	**06**	**05**	**18**					

INTERDISCIPLINARY MINOR

Computer Engineering

Sem	Course Type	Course Code	Course Name	L	T	P	S	Cr	Evaluation Scheme				
									Theory			Laboratory	
									MSE	TA	ESE	ISE	ESE
III	IDP-01		Fundamentals of Software Engineering	3	0	0	1	3	30	20	50	--	--
IV	IDP-02		Fundamentals of Programming and Algorithms	3	0	2	1	4	30	20	50	50	50
V	IDP-03		Fundamentals of Computer Networks	3	0	2	1	4	30	20	50	50	50
VI	IDP-04		Fundamentals of System Programming	3	0	2	1	4	30	20	50	50	50
VII	IDP-05		Fundamentals of Cyber Security	3	0	0	1	3	30	20	50	--	--
			Total	**15**	**00**	**06**	**05**	**18**					

(PCC) Microprocessors

Teaching Scheme

Lectures: 3 Hrs/ Week

Laboratory: 2 Hrs/ Week

Self-Study: 1 Hr/Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Laboratory: CIE: 100 Marks

Course outcomes

Students will be able to:

Upon completion of this course students will be able to:

1. Design a microcomputer around 8086b CPU in minimum mode
2. Explain importance of power on reset circuit in the microcomputer
3. Develop assembly language programs for 8086 CPU
4. Explain and write programs for dedicated interrupts (Break Point, Overflow) of 8086
5. Design and Test interfacing of 8255, 8279, 8259, 8251,DMA and interfacing of 8237 to 8086
6. Design maximum mode CPU module for coprocessor configuration and multiprocessor system for local bus and system bus

Course content

Design of Microcomputer: **[8 Hrs]**

Von Pneumann's Principle, 8086 Architecture, Advantages of pipelining and segmentation, Simultaneity and concurrency, Demultiplexing of address and data bus, ALE, Even and Odd Bank, BHE, Static memory organisation, Design of memory map, Design of Interfacing of static memory to 8086.Reset in, Power on Reset Circuit, Bus cycle, Instruction cycle, Organising program for Basic Input Output System or Boot Strap loader after reset, clock input, 8284 clock driver, Design of minimum mode CPU module.

Programming with 8086: **[8 Hrs]**

Programming model of 8086,Physical Address, Segment Address and Logical address/offset, Addressing modes, Instruction codes, default segment assignment, Instruction Groups, Flags, Important instructions of the group, flags, prefixes. Programming in Assembly Language of 8086. I/O interfacing. I/O mapped I/O versus Memory mapped I/O. In and OUT. Interfacing of 8 bit input port and outport in I/O mapped I/O mode. Disadvantage of memory mapped I/O in design of multitasking Microcomputer.

Interrupts: **[4 Hrs]**

Interrupt structure of 8086, Dedicated Interrupts, flags associated with interrupts, hardware interrupts, non vectored interrupt, INTR and INTA, Software Interrupts

I/O Interfacing: **[6 Hrs]**

Program driven data transfer versus Interrupt driven data transfer, Interfacing design and demonstrating modes of 8255, simple I/o, Strobed input and output, bidirectional strobed I/O, BSR, 8279, encoded and decoded scan, 2 key lockout and N key rollover, sensor matrix, strobed keyboard, 8259, EOI, Non specific and specific EOI, Priority structure, Initialisation and Operational Command words, cascaded mode, buffered and non buffered.

Serial Communication, DMA mode: **[4 Hrs]**

Serial I/O, Asynchronous and synchronous mode, design of serial communication using 8251 USART and

RS232C drivers, Direct Memory Access and Interfacing of 8237

Multiprocessing in 8086: **[8 Hrs]**

Design of maximum mode CPU module, 8288 bus controller and 8289 bus arbiter, coprocessor configuration, Introduction to NDP8087, Resident Bus and System bus, loosely coupled and closely coupled configuration. Protected mode of 80286 for improved memory management and task switching for multiprogramming or multitasking, PVAM, Register set, segment descriptors, selector, virtual address space, single level and multiple level tasks, cache memory and its management

Self-study **[12 Hrs]**

Emu86 or suitable assembler, Study of assembler directives, Instruction templates, No. of Bus cycles and clock cycles for execution of instructions, All instructions of 8086, Prefixes, Programming practice in assembly language of 8086, Function calls of DOS and BIOS Interrupts for display and keyboard of IBM PC Interfacing and programming for 4x4 key matrix and 4 seven segment multiplexed displays to 8255

Textbooks

1. John P Uffenbeck, The 8086/8088 Family Design, Programming and Interfacing, PHI

Reference Books

1. Yu-Cheng Liu, Glenn A. Gibson, Microcomputer Systems: The 8086/8088 family Architecture, Programming and Design, II Edition

Web references

1. https://nptel.ac.in/courses/108103157?utm
2. https://nptel.ac.in/courses/108105102?utm
3. https://archive.nptel.ac.in/content/storage2/courses/106108100/pdf/Lecture_Notes/LNm4.pdf

Laboratory assignments

Suggested List of Assignments:

1. **Experiment 1:** Study of 8086 based microcomputer trainer kit:
    a. Locate CPU8086, 74373 devices, 24 MHz crystal, 8284 clock generator, RAM and ROM devices on the microcomputer board. Find total memory capacity: Give table of memory map & usable space from RAM
    b. Use and describe following commands: D, E, R, T, G and U and A command.
    c. Study programming model and different addressing modes using relevant MOV instructions.
    d. Find opcodes for the instructions in C above using instructions templates of 8086.
2. **Experiment 2:**
    a. Explain briefly various directives of assembler
    b. Study instructions of 808All groups
    c. Write a program in assembly language of 8086 to add two 32 bit numbers. First number is stored from 0000H: 2000H; second number is stored from location 0000H: 3000H. Store the result from 0000H: 4000H onwards. Write your code in format: Address, Opcodes, Mnemonics, Comments
3. **Experiment 3 :** Write an assembly language program in 8086:
    a. As a subroutine to add two 32 bit numbers when operand pointers are initialized in main program. First number is stored from 1000H: 1000H and Second number is stored from 1000H: 2000H. Result can be stored from 1000H: 3000H.
    b. Use this subroutine in A above in the main program to add two series of 32 bit numbers. Store

the result from 1000H: 3000H onwards. If carry is generated in double word addition stores it at third location in sequence of the result.

4. **Experiment 4:**
    a. Design a microcomputer around 8086 CPU. Interface 64 KB of ROM and 64KB of RAM. The starting address of ROM is F0000H and starting address for RAM is 00000H. Create control signals for even and odd banks using BHE# and A0 Line. Interface 8255 programmable peripheral interface to 8086 with addresses 80H.
    b. Write a program to interface 8255 in simple IO mode. Generate a square wave of 1 Hz frequency having 50% duty cycle at port A and B. Write program for initialization of 8255 and square wave generation. Write subroutine for 0.5 second delay.
5. **Experiment 5:**
    a. Demonstrate 8255 for strobed input and strobed output in interrupt driven mode.
6. **Experiment 6:**
    a. Demonstrate 8255 for strobed bidirectional I/O in interrupt driven mode
    b. Interface 8279 to 8086 with starting address of 30H. Interface 4 seven segment LED displays and 4x4 key Matrix to 8279 in encoded scan and decoded scan mode.
7. **Experiment 7:**
    a. Demonstrate 8279 in Encoded scan right and left entry seven segment display mode with n key roll over and 2 key lockout for keys, Decoded scan keyboard with n key roll over and 2 key lockout.
8. **Experiment 8:** Divide by zero and overflow interrupt.
    a. Write ISS for divide by zero interrupt to display DIVIDE BY ZERO ERROR. Test the Interrupt Service Subroutine by creating divide by zero error in the main program.
    b. Repeat the exercise above to test Interrupt on overflow

This list is a guideline. The instructor is expected to improve it continuously.

(PCC) Principles of Programming Languages

Teaching Scheme
Lectures: 2 Hrs/ Week

Laboratory: 2 Hrs**/** Week

Self-Study: 1 Hr/Week

Evaluation Scheme

Theory: MSE:30 Marks, TA:20 Marks ESE:50 Marks

Laboratory: CIE: 100 Marks

Course outcomes

Students will be able to:

1. Explain, compare and discuss different language translation mechanisms.
2. Explain fundamental concepts of different programming paradigms.
3. Discuss and analyze factors that impact implementation of different programming language concepts and tradeoffs involved.
4. Demonstrate ability to write simple programs using exception handling and event driven programming concepts.
5. Suggest a suitable programming paradigm for a given problem.
6. implement the solution for a given problem using different programming paradigms.

Course content

Preliminaries: **[5 Hrs]**

Reasons for Studying Concepts of Programming Languages. Programming Domains. Language Evaluation Criteria. Influences on Language Design: computer architecture. Language Categories. Language Design Trade-Offs.

Implementation Methods: **[5 Hrs]**

Compilation, Interpretation, Hybrid Implementation, Preprocessors. Programming Environments. Evolution of the Major Programming Languages (Pseudocodes, Assembly, C, C++, FORTRAN, Java). Introduction to assembly language instructions.

Syntax and Semantics Lexical and Syntax Analysis **[8 Hrs]**

Names, Bindings, and Scopes, Type checking, Strong Typing. Type Conversions. Short-Circuit Evaluation.

Statement-Level Control Structures Subprograms: **[6 Hrs]**

Introduction. Fundamentals of Subprograms. Design Issues for Subprograms. Local Referencing Environments. Parameter- Passing Methods.

Exception Handling: **[6 Hrs]**

Basic Concepts. Design Issues. Exception Handlers. Binding Exceptions to Handlers. Event Handling: Basic Concepts. Event handling with Programming Language. Comparison of Exception handling & Event Handling. GUI development

Programming Paradigms: **[10 Hrs]**

Introduction to Logic Programming. Introduction. Applications of Logic Programming. Introduction to Functional Programming. Introduction. Introduction to LISP. Garbage Collection Algorithms. A Comparison of Functional and Imperative Languages. Introduction to Procedural Programming. Introduction to Object Oriented Programming

Self-study **[12 Hrs]**

Python as a multi-paradigm language for functional and logic programming. Case study of Ada, PERL, Go and

Ruby for their features. Comparison of efficiency of compiled and interpreted languages in the form of assignments.

Textbooks

1. Schesta R., "Concepts Of Programming Languages", 4th Edition, Pearson Education, ISBN-81-7808-161-X
2. T. W. Pratt , "Programming Languages", 4th Edition ,Prentice-Hall Of India, ISBN 9780130287199.

Reference Books

1. Ghezzi C, Milano P., Jazayeri M., "Programming Languages Concepts", 3rd Edition, John Wiley and Sons Pvt. Ltd (WSE), ISBN - 0195113063
2. Michael L. Scott “Programming Language Pragmatics”, ELSEVIER Publication, ISBN: 81-8147-370-1
3. Roosta S., "Foundations of Programming Languages", Thomson, Brooke/Cole, ISBN 981-243-141-1
4. Sethi R., "Programming Languages concepts & constructs", 2nd Edition, Pearson Education, ISBN 81 - 7808 - 104 – 0

Web references

1. https://nptel.ac.in/courses/106102067

Laboratory assignments

1. **Task 1:** To study various language implementation methods.
    a. Draw and explain the working of:
        i. Compiler
        ii. Interpreter
        iii. Hybrid Implementation System
        iv. Preprocessor
    b. Write a simple C program and observe:
        i. Source Code
        ii. Object Code Generation
        iii. Executable File Generation
    c. Compare execution time of:
        i. C Program
        ii. Python Program for the same logic (e.g., factorial or array sum).
2. **Task 2:** To understand low-level programming concepts.
    a. Study basic assembly instructions:
        i. MOV
        ii. ADD
        iii. SUB
        iv. MUL
        v. DIV
        vi. JMP
    b. Write assembly-style algorithms for:
        i. Addition of two numbers
        ii. Largest among three numbers

iii. Sum of first N numbers

c. Explain how computer architecture influences language design.

d. Draw CPU-memory interaction diagram.

3. **Task 3:** To understand the front-end phases of language processing.
   a. Given a C program, identify:
      i. Keywords
      ii. Identifiers
      iii. Constants
      iv. Operators
      v. Special Symbols
   b. Construct a token table.
   c. Draw parse trees for:
      i. a+b*c
      ii. (a+b)*(c-d)
   d. Write a Python program to count:
      i. Keywords
      ii. Operators
      iii. Identifiers from a source file.
4. **Task 4:** To study variable visibility and type systems.
   a. Write a C++ program demonstrating:
      i. Local Scope
      ii. Global Scope
      iii. Block Scope
   b. Demonstrate static binding and dynamic binding using C++ classes.
   c. Explain strong typing and weak typing with examples.
   d. Perform explicit and implicit type conversion examples.
5. **Task 5:** To understand expression evaluation mechanisms.
   a. Write programs demonstrating:
      i. Logical AND (&&)
      ii. Logical OR (||)
   b. Verify short-circuit evaluation using function calls.
   c. Compare normal evaluation and short-circuit evaluation.
   d. Measure the number of function calls executed in each case.
6. **Task 6:** To study function implementation and parameter passing techniques.
   a. Implement programs demonstrating:
      i. Call by Value
      ii. Call by Reference
   b. Explain Call by Result and Call by Value-Result using examples.
   c. Write a menu-driven calculator using functions.
7. **Task 7:** To understand runtime exception management and event-driven programming.
   a. Write programs for handling:
      i. Divide-by-zero exception
      ii. Invalid input exception
   b. Develop a simple GUI application using Python Tkinter or Java Swing containing:
      i. Button
      ii. Text Box
      iii. Label
   c. Implement button click event handling.
8. **Task 8:** To understand functional programming features.

a. Implement programs using:
    i. lambda()
    ii. map()
    iii. filter()
    iv. reduce()
b. Compare procedural and functional solutions for:
    i. Finding square of numbers
    ii. Filtering even numbers
c. Explain advantages and limitations of functional programming.

9. **Task 9:** To explore different programming paradigms and modern languages.
    a. Demonstrate Python as:
        i. Procedural Language
        ii. Object-Oriented Language
        iii. Functional Language
    b. Perform a case study of any two languages among:
        i. Ada
        ii. Perl
        iii. Go
        iv. Ruby
    c. Compare:
        i. Features
        ii. Advantages
        iii. Limitations
        iv. Application Areas
    d. Prepare a presentation (10–12 slides) summarizing findings.

(PCC) Data Structures and Algorithms

Teaching Scheme

Lectures: 3 Hrs/ Week

Laboratory: 2 Hrs/ Week

Self-Study: 1 Hr/Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Laboratory: CIE: 100 Marks

Course outcomes

Students will be able to:

1. Understand the fundamental concepts of various data structures (e.g., arrays, linked lists, stacks, queues, trees, graphs, hash tables).
2. Understand the concepts of linear and non-linear data structure and their representation
3. Demonstrate the ability to write abstract data types and the use of basic data structures in solving problems.
4. Discuss, compare and implement algorithms for various operations in different implementations of stacks and queues, tree, graph, matrices, heap, etc.
5. Analyze the time complexity of different searching, sorting, and traversal algorithms.

Course content

Fundamental Concepts **[6 Hrs]**

Introduction to Data Structures: Data, Data Object, Data types, Abstract Data Types (ADT), Data structures, Concept of primitive and non-primitive, linear and non-linear data structures. Introduction to Algorithms: Definition and Characteristics of an algorithm, Algorithm Specification, Performance Analysis- Time and space complexity, Asymptotic notations, Best, Average and worst cases.Introduction to Data representation and files: Text and binary files, use of various libraries for handling files.

Linear Data Structures and their storage representation **[6 Hrs]**

Features of Linear Data Structures, Array as ADT, List as ADT, Concept of linked linear list, Operations on linked linear list, Singly linked list, doubly linked list, circular linked list. Application of linked linear list- Polynomial manipulations, linked dictionary.

Linear Data Structures: Stacks and Queues **[8 Hrs]**

Stack and queue as ADT, Operations on stack and queue. Implementations using arrays and linked lists. Application of stack for expression conversion and evaluation, Recursion. Problems like maze and knight's tours

Non-Linear Data Structures: Trees **[6 Hrs]**

Concept of Non-Linear Data Structures, Tree: Basic terminology, representation of trees, Binary Tree: ADT Binary trees, Properties of a Binary Tree and its representations, Binary tree traversals – Inorder, preorder, postorder (recursive and non-recursive) and various operations. Binary Search Tree (BST): Definition, searching a BST, Operations of Insertion and deletion on BST. Heaps: Priority queues, Max and Min Heap, Operations on heap

Non-Linear Data Structures: Graphs **[8 Hrs]**

Graph ADT, Representation of graphs using adjacency matrix, adjacency list, Elementary graph operations- Breadth First Search (BFS), Depth First Search (DFS), Analysis of BFS and DFS Spanning Trees- Introduction, obtaining minimum cost spanning tree using greedy design strategy of Prim's and Kruskal's algorithms, Single source shortest paths using Dijkstra's algorithm.

Searching and Sorting **[8 Hrs]**

Linear and binary search algorithms and their analysis. Bubble Sort, Selection sort, Insertion Sort, Quick Sort, Heapsort with time complexity analysis. Hashing: hashing functions, chaining, overflow handling with and without chaining, open addressing: linear and quadratic probing.

Self-study **[12 Hrs]**

Content

Textbooks

1. E. Horowitz, S. Sahni, S.Anderson-freed, “Fundamentals of Data Structures in C”, Second Edition, University Press, ISBN 978-81-7371-605-8
2. B. Kernighan, D. Ritchie, “The C Programming Language”, Prentice Hall of India, Second Edition, ISBN 81-203-0596-5
3. Y. Langsam, M. Augenstein and A. Tannenbaum, “Data Structures using C”, Pearson Education Asia, First Edition, 2002, ISBN 978-81-317-0229-1

Reference Books

1. Ellis Horowitz, S. Sahni, D. Mehta “Fundamentals of Data Structures in C++”, Galgotia Book Source, New Delhi 1995 ISBN 16782928
2. Jean-Paul Tremblay, Paul. G. Sorensan, “An introduction to data structures with Applications”, Tata Mc-Graw Hill International Editions, 2nd edition 1984, ISBN-0-07- 462471-7

Web references

1. https://nptel.ac.in/courses/106102064
2. https://nptel.ac.in/courses/106106127
3. https://nptel.ac.in/courses/106105171

Laboratory assignments

Suggested List of Assignments:

1. Write a program to remove duplicate doubles from an array of doubles. In the program, write a function which accepts an array of doubles and removes the duplicates from the array and has return type void.
2. Compare the time complexity of two sorting algorithms, by following the given steps. Create a set of data files with count of integers varying into thousands and millions. Sort the files using both the algorithms. Plot graph of the time taken by both the programs using tools like gnuplot. Compare the graphs and comment on the time complexity theoretically predicted and practically observed.
3. Write a function which evaluates an infix expression, without converting it to postfix. The input string can have spaces, (,) and precedence of operators should be handled.
4. Implement a queue (that is write queue.c and queue.h only) of characters, such that on an enqueue, the char is added at the end of queue, and on a dequeue the first element is taken out, but the queue uses only a 'head' pointer and not a 'tail pointer.
5. Write a data type called "Integer". The data type should represent integers of unlimited length.
6. Write a sorting program with the following features: Reads data from a text file and sorts it

alphabetically by default. If the file has data in rows and columns (separated by space or tab) then allows sorting on a particular column. Allows any sort using numeric or alphabetical ordering.

7. Write the following functions for a binary search tree implementation: Searches the maximum value in the tree, preorder traversal without using recursion, Search the string in the tree and returns a pointer to the node, print the binary tree so that it looks like a tree.
8. Write code to list leaf nodes, non-leaf nodes and level of all nodes in a given binary tree.
9. Write a code for level order traversal of a binary tree with and without stack.
10. Start with an empty AVL tree and perform a series of insertions like : December, January, April, March, July, August, October, February, November, May, June. Display the tree.
11. Implement a sparse matrix with operations like initialize an empty sparse matrix, insert an element, sort a sparse matrix on a row-column, add two matrices and return the result as a matrix, transpose a matrix, etc.
12. Develop C functions to insert and delete into/from a max heap under the assumption that a dynamically allocated array is used, the initial capacity of this array is 1, and array doubling is done whenever we are to insert into a max heap that is full.
13. Implement Heap sort algorithm on a set of records, with a specified key.
14. Write a graph implementation, using adjacency lists and demonstrate Dijkstra's algorithm on it.
15. Write a program to read a map stored in a text file with (city1, city2, distance) comma separated values. Build a graph using this data. Print all pairs shortest paths information between all pairs of cities.
16. Implement DFS and BFS on a Graph.
17. Write a program to find all connected components of a Graph, on a map specified in a text file as Source, Destination, distance comma separated values.
18. Develop a hash table implementation in which overflows are resolved using chaining. Read a set of records from a file, insert them into a hash table, then perform a set of searches using supplied data and show the search results.
19. Implement a dictionary using a sparse matrix data structure.

eMini-project: Write an application of your own demonstrating your skills in defining a problem, writing down the requirements carefully, designing a modular solution with clear separation of abstract data types and their use, design of proper function prototypes and division of work among functions. The application can be a unix command re-implemented (e.g. cut, find, tar, fdupes, bc, etc.), reimplementation of C library functions, memory allocator, a simple game using libraries like n-curses or SDL, games like sudoku or chess, or an application to manage institutions like hospitals, colleges, shops, etc

(OE) Data Analytics

Teaching Scheme

Lectures: 2 Hrs/ Week

Self-Study: 1 Hr/Week

Evaluation Scheme

Theory: CIE:100 Marks

Course outcomes

Students will be able to:

1. Know basics of Data Analytics, people associated with it, roles and responsibilities to be an data Analyst.
2. Learn and apply how to interpret data in Python using multi-dimensional arrays in NumPy.
3. Learn and apply to manipulate DataFrames in Pandas.
4. Apply visualization libraries in Python like Matplotlib and Seaborn to gain insight of the dataset.
5. Learn the latest data analytics tools excel, Plotly, Power BI, Tableau.

Course content

Introduction to Data Analytics: **[6 Hrs]**

Introduction to Data Analytics, Need of data Analytics, types of data analytics, Challenges, different types of data, data sources, data collection, data integration, data wrangling, role of Data Engineers, Data Analysts, Data Scientists, Business Analysts, and Business Intelligence Analysts, roles, responsibilities and skill sets required to be a Data Analyst.

Introduction to Python Libraries-NumPy: **[6 Hrs]**

Installation, Basic operation: Arithmetic, Matrix Product, Increment, Decrement, Aggregate function, Indexing, Slicing and Iterating an array, Conditions and Boolean Arrays, Shape manipulations, Array manipulation-multidimensional array

Library: **[7 Hrs]**

Installation of Pandas, Testing pandas installation, Introduction to pandas data structure- the series, data frame, Index objects, other functionalities on Indexes-Reindexing, Dropping, Operations on Data Frame and Series- merging, combining, pivoting, removing, data transformation, Data aggregation, Reading and writing data to CSV file, random sampling, group iteration.

Data Visualization with Matplotlib and Seaborn libraries: **[7 Hrs]**

Pyplot, plotting window, adding elements to chart- text, grid, legend, saving charts, Handling data values, Line chart, Histogram, Bar chart, pie chart, scatter plot, box plot, heat map.

Self-study **[12 Hrs]**

Tools for Data Analytics: Introduction to data analytics tools, collection of maps, diagrams, charts designed to gather, interpret, and visualize data across diverse applications, how to Choose the right data analysis tool, data visualization with different free open source tools.

Textbooks

1. Fabio Nelli, “Python Data Analytics with Pandas, Numpy and Matplotlib”, ISBN: 978-1- 4842-3913-1, DOI: 10.1007/978-1-4842-3913-1, Publisher: Apress,Year: 2018
2. Bharti Motwani, “Data Analytics using Python “, Publisher: Wiley, ISBN: 8126502959

Reference Books

1. Vincent Granville, “Developing Analytic Talent -Becoming a Data Scientist”, Publisher: Wiley, 2014, ISBN: 9781118810095.
2. Glenn J. Myatt, Wayne Johnson, “Making Sense of Data I”, Second Edition, Publisher: Wiley, ISBN 978-1-118-40741-7
3. Glenn J. Myatt, Wayne Johnson, “Making Sense of Data II”, Publisher: Wiley, ISBN 978- 0-470-22280-5

Web references

1. https://nptel.ac.in/courses/106106212

(PCC) Object Oriented Programming & Design

Teaching Scheme

Lectures: 3 Hrs/ Week

Laboratory: 2 Hrs/ Week

Self-Study: 1 Hr/Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Laboratory: CIE: 100 Marks

Course outcomes

Students will be able to:

1. Design a class hierarchy using object oriented thinking for a given problem.
2. Create object oriented application code for a given problem.
3. Write small pieces of code demonstrating various object oriented programming concepts.
4. Compare, annotate, and comment on various object oriented programming concepts.
5. Demonstrate ability to write concurrent programs for given problems.
6. Apply different UML diagrams for given problems.

Course content

Introduction: **[8 Hrs]**

Various programming paradigms: Procedural, object-oriented, logic and functional, concurrent programming. Classes, Objects, Methods; Data types provided by OO languages; Abstract Data Types.

Abstraction Mechanisms: **[8 Hrs]**

Encapsulation; Constructors, Destructors; Polymorphism; Access specification: Private, public, protected members.

Inheritance: **[6 Hrs]**

Types of inheritance: single, multilevel, multiple, hierarchical and hybrid; Class hierarchies; Virtual Functions; Virtual Base class.

Standard Libraries: **[6 Hrs]**

Templates; Generic Programming using generic function and class; Packages; Interfaces. Iterators; Containers.

Exception Handling & Multi-Threaded Programming: **[6 Hrs]**

Exception handling, Exception types, Concurrent Programming, Basic Concepts of Concurrent Programming, Threads.

Introduction to OOAD: **[6 Hrs]**

Unified Process – UML diagram, use case, class diagrams, interaction diagrams, state diagrams – activity diagrams – package, component and deployment diagrams. Design: Unified Modelling language; use case diagrams; Class Diagrams

Self-study **[12 Hrs]**

Introduction to Design Patterns, Types of Design Patterns, Application on various design patterns.

Textbooks

1. Cay S Horstmann and Gary Cornell, Core Java Vol-1 and Vol-2, 9th Edition, Pearson Education India, ISBN-10: 9332518904 and 9332518890
2. Kenneth Barclay and John Savage, Object-Oriented Design with UML and Java, Elsevier Science. ISBN: 9780080497556
3. Ronald Leach, Object-Oriented Design and Programming with C++: Your Hands-On Guide to C++ , Academic Press, ISBN: 9781483214122
4. M.Ben Ari, Principles of Concurrent Programming, Prentice-Hall International, ISBN:0137010788

Reference Books

1. Herbert Schilt, "JAVA Complete Reference", 7th Edition, Tata McGraw Hill, ISBN:9780070636774
2. Scott W. Ambler ,The Elements of UML (TM) 2.0 Style, ISBN- 978-0521616782
3. Sharon Zakhour, Scott Hommel, Jacob Royal, Isaac Rabinovitch, Tom Risser, Mark Hoeber, "The Java Tutorial, "Addison Wesley Professional, 2006, Print ISBN-10: 0-321-33420-5
4. Eckel B., "Thinking in Java", 3rd Edition, Pearson Education, 2012
5. E. Balagurusamy, Object Oriented Programming with C++, 6th Edition, McGraw Hill, ISBN- 10: 125902993X

Web references

1. https://nptel.ac.in/courses/106105191

Laboratory assignments

Suggested List of Assignments

This is only a suggestive list. The instructor is encouraged to update the list of assignments and projects. The purpose of the assignments should be to lead to a meaningful project. The students should be encouraged to build different projects.

1. Define a class to represent a bank account. Include the following details like name of the depositor, account number, type of account, balance amount in the account. Write
2. methods to assign initial values, to deposit an amount, withdraw an amount after checking the balance, to display name, account number, account type and balance.
3. Write a program to implement the following. Create a base class called person consisting
4. of name and code. Create 2 child classes. Account with member pay and b. Admin with experience and inherit the base class. Create a class Employee with name, code, experience and pay by inheriting the above class.
    a. Write Python script to display
    b. Current date and time,
    c. Current year,
    d. Month of year,
    e. week number of the year,
    f. weekend of the week,
    g. Day of year,
    h. Day of the month and Day of week.
5. Write a program that has an abstract class Polygon. Derive two classes Rectangle and Triangle from Polygon and write methods to get details of their dimensions hence calculate

the area.

6. Write a menu driven to read, add, subtract, multiply, divide and transpose two matrices.
7. Write a program that has a class TIME. Enter the time when a user started an online test and completed the test. Subtract the two-time values and display the duration in which the test was completed. Throw exceptions whenever need arises (like invalid data, or if start time is greater than completion time).
8. Construct a class diagram for a geometrical document. Add at least 10 relationships (associations and generalizations). Use associations and association end names wherever required. Also use qualified associations and show multiplicity. You do not need to show attributes or operations. As you prepare the diagrams, you may add classes. Be sure to explain your diagrams.
9. Scenario: An extension ladder has a rope, pulley, and latch for raising, lowering, and locking the extension. When the latch is locked, the extension is mechanically supported and you may safely climb the ladder. To release the latch, you raise the extension slightly with the rope.
10. You may then freely raise or lower the extension. The latch produces a clacking sound as it
11. passes over rungs of the ladder. The latch may be reengaged while raising the extension by reversing direction just as the latch is passing a rung. Construct a state diagram of an extension ladder.
12. Draw a Scenario: A hockey league is made up of at least four hockey teams. Each hockey team is composed of six to twelve players, and one player captains the team. A team has a name and a record. Players have a number and a position. Hockey teams play games against each other. Each game has a score and a location. Teams are sometimes lead by a coach. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses. Construct a UML Class Diagram representing the above problem domain for a hockey league.

(PCC) Computer Organization

Teaching Scheme

Lectures: 3 Hrs/ Week

Self-Study: 1 Hrs/ Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Analyze Performance aspects, Instruction set architecture, and Functional Units.
2. Apply different computer arithmetic principles for implementation of functional units.
3. Interpret the working of memory hierarchy of computer system.
4. Justify the need of paging in virtual memory and secondary storage with its advantages.
5. Measure and analyze features of multiprocessor systems like bus arbitration, Instruction pipelining, and RISC & CISC.
6. Examine Input/Output including interfaces, buses, interrupt handling mechanisms, and I/O controllers

Course content

CPU Architecture: **[6 Hrs]**

Performance understanding, Amdahl's Law, Benchmarking, Flynn's Classification, Instruction format, control signals in CPU, micro program control unit and hardwired control unit, ALU & sequencer, look ahead carry generator, MIPS ISA

Arithmetic: **[6 Hrs]**

Integer Arithmetic-multiplication, Booth's Algorithm, division algorithm; Floating point number representation, and floating-point arithmetic

Memory: **[8 Hrs]**

Dynamic RAM organization, Cache memory &basic cache optimizations, cache coherence & MESI protocol, virtual memory, secondary storage, MBR and GPT hard disks, RAID, File system FAT

System and memory map: **[7 Hrs]**

Closely coupled and loosely coupled multiprocessor systems, bus arbitration, co-processor, lower 1MB memory map

Instruction Pipelining: **[7 Hrs]**

Basic concepts and issues, Introduction to the basic features & architecture of RISC & CISC processors, super scalar processor, MIPS pipeline

Multiprocessing: **[5 Hrs]**

Symmetric multiprocessing (SMP), Asymmetric multiprocessing (AMP), Hardwired based multithreading approaches and examples, Different examples of computer organization as per Flynn's Classification

Self-study **[12 Hrs]**

Closely coupled and loosely coupled multiprocessor systems, Differences in RISC & CISC approaches, Differences in FAT and NTFS, Programs exhibiting Locality of Reference, Dependence analysis and hazard

detection using different resources.

Textbooks

1. William Stallings, Computer Organization and Architecture, 11/E ISBN- 9781292420103/ 9781292420080, Pearson Education, Global Edition, 2021-22.
2. Carl Hamacher, Zvonko Vranesic, Safwat Zaky, and Naraig Manjikian, Computer Organisation, 6th Edition, ISBN- 978-0-07-338065-0/ 0-07-338065-2, McGrawHill, 2011-12.

Reference Books

1. D. Patterson, J. Hennessy, Computer Organization and Design: The Hardware Software Interface, RISC V Edition, ISBN- 978-0-12-812275-4, Morgan Kauffman, 2018.
2. Liu & Gibson, Microcomputer Systems, Second Edition, ISBN: 978-81-203-0409-3, PHI, 1985.
3. John Uffenbeck, THE 8086/ 8088 FAMILY: Design, Programming, and Interfacing, EEE, ISBN- 0132467526/ 9780132467520, PHI, 1987.

Web references

1. https://nptel.ac.in/courses/106105163
2. https://nptel.ac.in/courses/106102114

(PCC) Theory of Computation

Teaching Scheme

Lectures: 3 Hrs/ Week

Tutorial: 1 Hrs / Week

Self-Study: 1 Hrs/ Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Design appropriate automata for modeling the solution for various computational problems.
2. Apply transformation between multiple representations of automata/machines.
3. Make use of the pumping lemma to show that a language is not regular/context-free.
4. Distinguish different formal computing languages and classify their respective types.
5. Describe the limitations of a computing machine in terms of language recognition.

Course content

Finite Automata: **[10 Hrs]**

Computability, and Complexity, Strings and Languages: symbol, alphabet, string, formal languages, Formal definition of a finite automaton, Designing finite automata, Non- determinism: Formal definition, Equivalence of NFAs and DFAs, Minimization of DFA.

Regular Expressions: **[4 Hrs]**

Regular Expressions: Formal definition of a regular expression, Equivalence with finite automata, Closure properties of regular languages, the pumping lemma for regular languages.

Context-Free Languages: **[6 Hrs]**

Context-free Grammars: Formal definition, Designing context-free grammars, Parse Trees, Ambiguity, Chomsky Normal Form. Pushdown Automata: Formal definition, Examples.

Turing Machines and Undecidability: **[6 Hrs]**

Turing Machines Formal definition, Examples, Decidability and Undecidability:Decidable Languages: Decidable problems concerning regular languages, Decidable problems concerning context-free languages, Undecidable problems, The Halting Problem.

Self-study **[12 Hrs]**

Basic Mathematical notations and terminologies for set theory. Closure Properties of context-free languages, the pumping lemma for context-free languages. Variants of Turing Machine: Multi-tape Turing machines, Nondeterministic Turing machines, Enumerators.

Textbooks

1. Michael Sipser, “Introduction to the Theory of Computation”, 3rd Edition, 2013, Cengage Learning Publications, ISBN-13: 978-1133187790.
2. John. E. Hopcroft, Rajeev Motwani, J. D. Ullman, “Introduction to Automata Theory, Languages, and Computations”, 3rd Edition, 2009, Pearson Education Publisher, ISBN-10: 0321455363 /

ISBN-13: 978-0321455369

Reference Books

1. E. V. Krishnamurthy, “Theory of computer science”, 2004, Affiliated East Press Publications, ISBN-10: 038791255X / ISBN-13: 978-0387912554.
2. Dexter C. Kozen, "Automata and Computability", 1997, Springer Verlag Publications, ISBN 0-387-94907-0.
3. Harry Lewis, Christos H. Papadimitriou, “Elements of the Theory of Computation”, 2nd Edition, 1997, Prentice-Hall Publications, ISSN 0891-4516.
4. John Martin, “Introduction to Languages and Theory of Computations”, 4th edition, 2010, McGraw-Hill Publications, ISBN 978–0–07–319146–1 / MHID 0–07–319146–9.

Web references

1. https://nptel.ac.in/courses/106106049

(OE) Fundamentals of Operating Systems

Teaching Scheme

Lectures: 2 Hrs/ Week

Self Study: 1 Hr/Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Understand the structure of OS and basic architectural components involved in OS design.
2. Describe the mechanisms of OS to handle processes, threads, and their communication.
3. Analyze the memory, file, and resource management techniques.
4. Illustrate the role of paging, segmentation, and virtual memory in operating systems.

Course content

Introduction to Operating System (OS): **[6 Hrs]**
Computer System, OS Operations, Virtualization, Distributed Systems, Kernel Data Structures, Computing Environments, OS Services, User and OS Interface, System Calls, System Services, OS Design, OS Structure.

Process Management: **[6 Hrs]**
Scheduling, Operations, Interprocess Communication, Models of IPC, Thread, Multicore Programming, Multithreading Models, Implicit Threading, CPU scheduling, Scheduling Algorithms, Thread Scheduling, Multi-Processor Scheduling.

Process Synchronization: **[6 Hrs]**
Critical-section, Peterson's Solution, Hardware Support, Mutex Locks, Semaphores, Classic Problems of Synchronization, Synchronization within the Kernel, Deadlocks: necessary conditions, Prevention, Avoidance, and Detection.

Memory Management: **[6 Hrs]**
Logical and Physical Memory, Contiguous Memory Allocation, Paging, Swapping, Virtual Memory, Demand Paging, Page Replacement, Allocation of Frames, Thrashing, Secondary Storage: HDD Scheduling, Swap-Space Management

File Management: **[6 Hrs]**
File Concept, Access Methods, Directory Structure, Protection, Filesystem Structure, Directory Implementation, Allocation Methods, Free-Space Management, Recovery, Filesystem Mounting, Partitions and Mounting, File Sharing

Self-study **[12 Hrs]**
Case Studies of Linux OS: Design Principles, Kernel Modules, Process Management, Scheduling, Memory Management, File Systems.

Textbooks

1. "Operating System Concept" by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, 10th Edition, 2021, Wiley, ISBN-978-1-119-32091-3.

2. “Modern Operating System”, by Andrew S. Tanenbaum, 5th Edition, 2022, Pearson Education, ISBN 13: 978-0137618873.

Reference Books

1. Milan Milenkovic; Operating Systems; Tata McGraw Hill; Second Edition. ISBN: 0- 07-044700-4
2. Andrew S. Tanenbaum; Modern Operating Systems; Prentice Hall of India Publication; 3rd Edition. ISBN: 978-81-203-3904-0
3. Maurice J. Bach; The Design of the Unix Opearating System; Prentice Hall of India; ISBN: 978-81-203-0516-8
4. Uresh Vahalia; Unix Internals, The New Frontiers; Prentice Hall; ISBN: 0-13-101908- 2

Web references

1. https://nptel.ac.in/courses/106106144
2. https://nptel.ac.in/courses/106108101

(OE) Fundamentals of Algorithms

Teaching Scheme

Lectures: 2 Hrs/ Week

Self-Study: 1 Hr/Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Recall and comprehend time and space complexity of algorithms and evaluate algorithmic efficiency in terms of runtime and memory usage using asymptotic analysis.
2. Describe the performance of recursive algorithms and analyze the correctness and efficiency of their solutions.
3. Evaluate and use appropriate algorithmic techniques, and design efficient solutions to various computational challenges.
4. Apply design technique that would be most suitable to solve a given problem, justify the technique used and analyze the resultant algorithm.

Course content

Basics of Algorithm: **[6 Hrs]**

Introduction to algorithm, Classification of algorithms- Deterministic, non- deterministic, performance analysis of an algorithm, time and space complexity, asymptotic notations (O, Ω, θ notations)**,** complexity analysis with examples.

Design Techniques-I: **[6 Hrs]**

Greedy Methods: Introduction, Knapsack problem, Job sequencing with deadlines, Dijikstra's Single source shortest paths, Minimum cost spanning tree: Prim's algorithm, Kruskal's algorithm, Optimal merge patterns.

Design Techniques-II: **[5 Hrs]**

Divide and Conquer: Introduction, Mergesort, Binary Search, Quicksort, Multiplication of two n-bit numbers.

Design Techniques-III: **[5 Hrs]**

Dynamic Programming: Introduction, All pairs shortest paths, Traveling salesperson problem, Longest common subsequence, Bellman-Ford algorithm

Graph Algorithms: **[4 Hrs]**

Depth-first search (DFS), breadth-first search (BFS), Topological sort.

Selected Algorithms: **[4 Hrs]**

String Matching: The naïve string-matching algorithm, The Robin-Karp algorithm, The Knuth- Morris-Pratt algorithm.

Self-study **[12 Hrs]**

Algorithmic Problem Solving Platforms:Activity: Solve algorithmic problems on platforms like LeetCode or HackerRank.

Textbooks

1. Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, “Fundamentals of Computer Algorithms”, Universities Press, 2nd edition (2008) , ISBN-13: 978-8173716126
2. Thomas Cormen, Charles Leiserson, Ronald Rivest and Cliford Stein, “Introduction to Algorithms”, PHI, 3rd edition, ISBN-13: 978-8120340077

Reference Books

1. Gilles Brassard and Paul Bratley, “Fundamentals of Algorithmics”, PHI, ISBN-13: 978 8120311312
2. Jon Kleinberg and Éva Tardos, “Algorithm Design”, Pearson Education India, ISBN-13: 978-9332518643

Web references

1. https://nptel.ac.in/courses/106102064
2. https://onlinecourses.nptel.ac.in/e-learning/preview/noc26_cs67

(VSEC) Development Tools Laboratory

Teaching Scheme

Laboratory: 2 Hrs/ Week

Evaluation Scheme

Laboratory: CIE: 100 Marks

Course outcomes

Students will be able to:

1. Develop an application in a group using GIT, demonstrating ability to work remotely, push, and pull.
2. Write a report in a specified format using LaTEX.
3. Demonstrate programming ability using Unix Shell.

Course content

LaTEX: **[6 Hrs]**

Basic syntax, compiling and creating **documents**; Document structure, sections, paragraphs; packages, Math, Adding Images, Drawing images (using tools like Inkscape) Table of contents; Source code, graphs (using tools like Graphviz), Adding references, different templates, IEEE format, Bibliography

Shell Programming: **[6 Hrs]**

Introduction to Linux commands, concept of shell, shell variables, getcwd() and pwd; Introduction to shell programming features: Variables declaration & scope, test, return value of a program, if-else and useful examples, for and while loop, switch case; Shell functions, pipe and redirection, wildcards, escape characters; Awk script: Environment and workflow, syntax, variables, operators, regular expressions, arrays, control flows, loops, functions, output redirections

GIT: **[8 Hrs]**

Creating a project using git locally, add, commit, status, diff; branch and merge, GIT: cloning a remote repo, working with a remote repo – git push, pull, fetch; creating issues and pull requests; working on a project in a distributed fashion

Web references

1. Leslie Lamport, “LaTeX: A document preparation system”, User’s guide and reference manual, 2nd Edition, 1994, by Addison-Wesley Professional. ISBN 0201529831, 9780201529838
2. Stefan Kottwitz, “LaTeX Beginner's Guide: Create High-quality and Professional-looking Texts, Articles, and Books for Business and Science Using LaTeX, Packt Publishing, 2011. ISBN: 1847199860, 9781847199867 ”https: //www.latex-project.org/
3. Introduction to LaTEX, MIT http: //web.mit.edu/rsi/www/pdfs/new-latex.pdf
4. A simple guide to LaTeX - Step by Step https: //www.latex-tutorial.com/tutorials/
5. Bash Guide for Beginners: https://www.tldp.org/LDP/Bash-Beginners- Guide/Bash- Beginners-Guide.pdf
6. Bash Reference Manual https://www.gnu.org/software/bash/manual/bash.pdf
7. Tutorials on Shell Programming https://www.shellscript.sh/ https://www.tutorialspoint.com/unix/shell_scripting.htm

8. Pro GIT Book https://github.com/progit/progit2/releases/download/2.1.204/progit.pdf

Laboratory assignments

The lab incharge will impart instructions during a part of the lab session and assign the assignments based on it for an appropriate duration of days.

1. Format a given essay using sections, paragraphs, headings in LaTEX.
2. Format a given report in IEEE format using LaTEX.
3. Write a shell program which reads a set of unspecified count of numbers and prints their sum and average.
4. Write a shell program to extract a compressed file in any format (zip, tar.gz, tar.gz2, .tar, .bz2, .gz, .rar, .Z, .7z, etc)
5. Write a shell program to convert a CSV file of contacts, into a VCF file.
6. Write a shell program to sort all files stored in a given folder hierarchy, on their size.
7. Write a shell script to manage a todo list from the command line. The script should be able to add, remove, list, sort, prepend, append, deduplicate todo-items
8. Write a program that scans a file line by line, splits each input line into fields later, compares input line/fields to pattern and performs action(s) on matched lines
9. Develop a program using git locally. E.g. add the exponent operator to the calculator program that you wrote. Demonstrate the ability to do git add, commit, status, diff.
10. Create a branch in your calculator program to do hexadecimal calculation. Write code and develop two branches. Merge the two to have a decimal/hex calculator. Demonstrate git branch, merge capability.
11. In a group of 3, create a github/gitlab repo. Raise issues, send pull requests, do the local and remote merges and finally get a synced local repo. For example, in the calculator project one student will become the developer, the other two will create issues and send pull requests for features like adding an operator, developing and pulling those requests. Rotate the roles and repeat.

(MDM) Data Structures, Files and Algorithms

Teaching Scheme

Lectures: 3 Hrs/ Week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Write neat code by selecting appropriate data structure and demonstrate a working solution for a given problem.
2. Demonstrate the ability to write reusable code and abstract data types.
3. Demonstrate the ability to implement different data structures with a variety of implementations.
4. Analyze and compare given algorithms for time and space complexity.
5. Handle all possible cases in designing an algorithm.
6. Demonstrate the ability to write modular and re-usable code.

Course content

Introduction: **[4 Hrs]**

Concept of Data types and Abstract Data types; Characteristics of an algorithm; Analyzing programs; Frequency count; Time and space complexity; Big 'O' and Ω" notation; Best, average and worst cases; Programming language provided data types, operations on various data types; Dangling pointers and garbage memory.

Arrays, Searching and Sorting: **[8 Hrs]**

Searching: linear and binary search algorithm; Hashing: hashing functions, chaining, overflow handling with and without chaining, open addressing: linear, quadratic probing; Sorting: bubble sort, selection sort, quick sort, merge sort, insertion sort. Time complexity analysis of searching and sorting techniques.

Files and File Handling: **[6 Hrs]**

Files handling: various library functions for handling files; system call interface and library interface; Different file formats like csv, pdf, odt, etc; Text and binary files; Programs to copy, concatenate, rename files; Programs to handle existing file types; arguments to main()

Stacks and Queues: **[6 Hrs]**

Stack and queue as ADT; Operations on stack and queue; Implementations using arrays and dynamic memory allocation; Application of stack for expression evaluation, expression conversion; Recursion and stacks; Problems like maze and knight's tour.

Linked Lists: **[8 Hrs]**

Dynamic memory management; List as ADT; Concept of linked organization of data against linked list; Singly linked list, doubly linked list, circular linked list; Representation & manipulations of polynomials/sets using linked lists.

Trees: **[8 Hrs]**

Basic terminology; Binary trees and its representation; Types of Binary tree; Binary tree traversals and various operations; Insertion and deletion of nodes in binary search tree; Applications of trees.

Self-study **[12 Hrs]**

Introduction to graph data structure; Types of graphs: directed, undirected, weighted, unweighted Graph representation: adjacency matrix, adjacency list; Graph traversal algorithms: depth-first search (DFS), breadth-first search (BFS);Minimum spanning tree algorithms: Prim's algorithm, Kruskal's algorithm

Textbooks

1. E. Horowitz, S. Sahni, S.Anderson-freed, “Fundamentals of Data Structures in C”, Second Edition, University Press, ISBN 978-81-7371-605-8
2. B. Kernighan, D. Ritchie, “The C Programming Language”, Prentice Hall of India, Second Edition, ISBN 81-203-0596-5
3. Y. Langsam, M. Augenstin and A. Tannenbaum, “Data Structures using C”, Pearson Education Asia, First Edition, 2002, ISBN 978-81-317-0229-1

Reference Books

1. Ellis Horowitz, S. Sahni, D. Mehta “Fundamentals of Data Structures in C++”, Galgotia Book Source, New Delhi 1995 ISBN 16782928
2. Jean-Paul Tremblay, Paul. G. Soresan, “An introduction to data structures with Applications”, Tata Mc-Graw Hill International Editions, 2nd edition 1984, ISBN-0-07-462471-7

Web references

1. GeeksforGeeks (Data Structures)
   https://www.geeksforgeeks.org/data-structures/
2. Programiz (Data Structures & Algorithms)
   https://www.programiz.com/dsa
3. TutorialsPoint (Data Structures and Algorithms)
   https://www.tutorialspoint.com/data_structures_algorithms/index.htm
4. Programiz (C File Handling)
   https://www.programiz.com/c-programming/c-file-input-output
5. MIT OpenCourseWare – Introduction to Algorithms
   https://ocw.mit.edu/courses/6-006-introduction-to-algorithms-fall-2011/

(HSMC) Design Thinking

Teaching Scheme

Lectures: 1 Hrs/ Week

Self-study: 1 Hrs/ Week

Evaluation Scheme

Theory : CIE:100 Marks

Course outcomes

Students will be able to:

1. Outline various learning styles and psychological principles and Infer Design Thinking principles & methodology.
2. Explain the levels of designs and Experiment with the process using human centric tools.
3. Propose real-time innovative engineering product designs and choose appropriate frameworks, strategies, techniques for prototype development.
4. Appraise user feedback and propose corrective innovative solutions to meet project requirements using critical thinking skills.

Course content

An Insight to Learning **[3 Hrs]**

Experiential Learning Styles, Self-assessment Psychological Principles in Design Thinking
Perception & Observation, Imagination & Creative Confidence (lateral thinking & 6 thinking hats)

Design Thinking Framework **[2 Hrs]**

Introduction to different frameworks of DT, Stanford d. school framework Empathize, Define, Ideate, Prototype, Test
Case Study: IDEO Shopping Cart, etc.

User-Design Relationship **[2 Hrs]**

Levels of Designs
Understanding users through interviews, personas, empathy maps/affinity diagrams/journey maps and need identification

Introduction to Human Centric Tools in Design Thinking Process **[2 Hrs]**

Brainstorming & Mind mapping, POV and HMW

Process of Product Design **[2 Hrs]**

Process of simple Product Design using real life problem statements from our daily routine activities, Design Thinking Approach, Stages of Product Design, Examples of best product designs and functions. Hands on Lab Assignment –Simple routinely used Product Design. Hands-on demonstration of how to translate the ideas into physical objects. Better visualization of the ideas and concepts using, IDEA LAB/FAB LAB facilities such as Wood router, Laser cutting of thin plastic sheets, clay modelling, Expandable Polystyrene etc

Prototyping **[2 Hrs]**

What is Prototype? Understanding necessity of making prototypes by building the prototypes for pre-selected Engineering problem, using one or combinations of the digital fabrication techniques & electronics fabrication systems.

Self-study **[12 Hrs]**

Testing, Sample Example, Test Group Marketing Feedback, Re-Design & Re-Create
Final Presentation – “Solving Practical Engineering Problem through Innovative Product Design & Creative Solution”

Textbooks

1. Norman, D. (2013). The Design of Everyday Things. Basic Books, NY.
2. Norman, D. (2004). Emotional Design. Basic Books, NY.
3. Brown, T. (2019). Change by Design. HarperCollins Publishers, NY.
4. Lal, D. M. (2021). Design Thinking- Beyond the Sticky Notes. Sage Publications India Pvt. Ltd.

Reference Books

1. Malik, A. D. M. (2019). Design Thinking for Educators. Notion Press, Chennai, India.
2. E. F. Crawley, "Creating the CDIO Syllabus, a universal template for engineering education," 32nd Annual Frontiers in Education, Boston, MA, USA, 2002, pp. F3F-F3F, doi: 10.1109/FIE.2002.1158202.
3. Dym, C. L., Agogino, A. M., Eris, O., Frey, D. D., & Leifer, L. J. (2005). Engineering design thinking, teaching, and learning. Journal of engineering education, 94(1), 103-120.
4. Panke, S. (2019). Design thinking in education: Perspectives, opportunities and challenges. Open Education Studies, 1(1), 281-306.
5. Parmar, A. J. (2014, October). Bridging gaps in engineering education: Design thinking a critical factor for project-based learning. In 2014 IEEE frontiers in education conference (FIE) proceedings (pp. 1-8). IEEE.

Web references

1. Thompson, L., & Schonthal, D. (2020). The Social Psychology of Design Thinking. California Management Review, 62(2), 84–99. https://doi.org/10.1177/0008125619897636

(PCC) Discrete Structures

Teaching Scheme

Lectures: 2 Hrs/ Week

Self-study: 1 Hrs/ Week

Evaluation Scheme

Theory: CIE: 100 Marks

Course outcomes

Students will be able to:

1. State real life situations using predicate/propositional calculus and do logical arguments leading to conclusions.
2. Solve problems involving sets, functions, sequences and summation
3. Apply principles of counting and induction to solve real life problems.
4. Solve problems involving algebraic systems.
5. Apply principles of relations to solving real life problems.

Course content

The Foundation Logic and Proofs **[6 Hrs]**

Propositions, Conditional Propositions, Logical Connectivity, Propositional equivalences, predicates and Quantifiers, First order logic, Proofs: Proof Techniques, Rule of inference.

Sets, Functions Sequences and Sums **[6 Hrs]**

Set, set operation, Functions, Types of Functions, Functions: Definition, Domain, Range, Image, etc. Types of functions: Surjection, Injection, Bisection, Inverse, Identity, Composition of Functions,

Counting **[8 Hrs]**

Basic Counting Techniques (sum, product, subtraction, division, exponent), Pigeonhole Principle, Permutations and Combinations and numerical problems, Binomial Coefficients, Pascal's Identity and Triangle.

Relations **[6 Hrs]**

Definitions, Properties of Binary Relations, Representing relation, closures of relations, Equivalence Relations and partitions, Partial ordering relations.

Self-study **[12 Hrs]**

Algebraic Systems, Groups, Semi Groups, Monoids, Subgroups, Permutation Groups, Codes and Group codes, Isomorphism and Automorphisms, Homomorphism and Normal Subgroups, Ring, Field.

Textbooks

1. “Discrete Mathematics and Its Applications”, Kenneth H. Rosen, 7th Edition, Tata McGraw-Hill, 2017, ISBN: 9780073383095.
2. “Elements of Discrete Mathematics”, C. L. Liu, 4th Edition, Tata McGraw-Hill, 2017, ISBN-10: 1259006395, ISBN-13: 9781259006395.

Reference Books

1. “Discrete Mathematical Structures”, G. Shanker Rao, 2nd Edition, 2009, New Age International, ISBN-10: 8122426697, ISBN-13: 9788122426694.
2. “Discrete Mathematics”, Lipschutz, Lipson, 2nd Edition, 1999, Tata McGraw-Hill, ISBN: 007463710X.
3. “Graph Theory”, K. Balakrishnan, 1st Edition, 2004, Tata McGraw-Hill, ISBN-10: 0-07-058718-3, ISBN-13: 9780070587182.

4. “Discrete Mathematical Structures”, B. Kolman, R. Busby and S. Ross, 4th Edition, Pearson Education, 2002, ISBN: 8178085569.
5. “Discrete Mathematical Structures with Application to Computer Science”, J. Tremblay, R. Manohar, Tata McGraw-Hill, 2002, ISBN: 0070651426.

Web references

1. https://onlinecourses.nptel.ac.in/e-learning/preview/noc22_cs49
2. https://onlinecourses.nptel.ac.in/e-learning/preview/noc22_cs04
3. https://ocw.mit.edu/courses/6-042j-mathematics-for-computer-science-spring-2015/

(HON) Making Sense of Data

Teaching Scheme
Lectures: 3 Hrs/ Week
Self-study: 1 Hr/Week

Evaluation Scheme
Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

1. Understand and identify different types of data, data distribution.
2. Analyze dataset using estimation methods, hypothesis testing, and analysis of variance to support statistical decision-making.
3. Demonstrate the use of data preprocessing methods and data analytic tools for effective data preparation and analysis.
4. Apply data visualization techniques to represent and interpret univariate, bivariate, and multivariate data effectively.
5. Demonstrate clustering and association rule mining techniques to discover patterns and relationships in data.

Course content

Introduction: **[6 Hrs]**

Overview, Sources of Data, Process for Making Sense of Data. Describing Data: Observations and Variables, Types of Variables, types of data, Data Distributions: Discrete Distributions such as Binomial, Poisson, Geometric etc.; Continuous Distributions such as Exponential, Normal etc.; Expectation: Moments; Central Limit theorem and its significance; Some sampling distributions like chi-square, t, F.

Statistical Inference: **[6 Hrs]**

Estimation - Introduction, classical methods of estimation, single sample: estimating the mean and variance, two samples: estimating the difference between two means and ratio of two variances; Tests of hypotheses - Introduction, testing a statistical hypothesis, tests on single sample and two samples concerning means and variances; ANOVA - One–way, Two–way with/without interactions.

Data Analytics Tools **[6 Hrs]**

Data analytics using Python: Statistical Procedures, NumPy, Pandas, SciPy, Matplotlib.

Data Pre-Processing **[6 Hrs]**

Understanding the Data, Dealing with Missing Values, Data Formatting, Data Normalization, Data Binning, Importing and Exporting data in Python, Turning categorical variables into quantitative variables in Python, Accessing Databases with Python.

Data Visualization **[6 Hrs]**

Graphic representation of data, Characteristics and charts for effective graphical displays, Chart types- Single var: Dot plot, Jitter plot, Error bar plot, Box-and-whisker plot, Histogram, Two variable: Bar chart, Scatter plot, Line plot, Log-log plot, More than two variables: Stacked plots, Parallel coordinate plot.

Identifying and Understanding Groups **[6 Hrs]**

Clustering: Agglomerative Hierarchical Clustering, k-Means Clustering, Association rules: Grouping by Combinations of Values, Extracting and Assessing Rules.

Self-study **[12 Hrs]**

Objectives of EDA, Sign Test, Wilcoxon Signed Rank Test, Seaborn Library, Feature extraction, Feature construction, Feature selection, Handling outliers, Violin Plot, Density Plot, Heat Map, Radar Chart,Silhouette Score, Elbow Method.

Textbooks

1. Glenn J. Myatt, Wayne P. Johnson, Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining, Wiley 2009.
2. Glenn J. Myatt, Wayne P. Johnson, Making Sense Of Data II A Practical Guide To Data Visualization, Advanced Data Mining Methods, And Applications,Wiley 2009.

Reference Books

1. H. Davenport, Jeanne G. Harris and Robert Morison, “Analytics at Work: Smarter Decisions, Better Results”, Harvard Business Press, 2010
2. Anil Maheshwari, “Data Analytics made accessible,” Amazon Digital Publication, 2014.

Web references

1. https://numpy.org
2. https://pandas.pydata.org
3. https://scipy.org

(HON) Big Data Analytics

TeachingScheme
Lectures:3 Hrs/Week
Laboratory: 2 Hrs/ Week
Self-Study: 1 Hrs/ Week

EvaluationScheme
Theory: MSE: 30Marks,TA: 20Marks,ESE: 50 Marks
Laboratory: CIE: 50 Marks, ESE: 50 Marks

Course outcomes

1. Identify , evaluate the need for Big Data and analyze its challenges,
2. Apply the Hadoop framework to process datasets and demonstrate use of ecosystem tools in distributed computing
3. Design and implement Big Data solutions using Spark, and assess performance in analyzing large datasets.
4. Develop and optimize data analysis using Apache Pig , Hive for large-scale datasets.

Course content

Introduction to Big Data [5Hrs]

Types of Digital Data, Overview of Big Data Analytics, Evolution of Big Data, Characteristics of Big Data - Volume, Variety, Velocity, Veracity, Valence, Value, Applications of Big Data, Challenges with Big data, Introduction to Enabling Technologies for Big Data.

Big Data Technology Stack [7Hrs]

Introduction to Big Data Technology Stack, Overview of Big Data distribution packages, Introduction to Big Data Platforms, Big Data Storage Platforms for Large Scale Data Storage, CAP Theorem, Eventual Consistency, Consistency Trade-Offs, ACID and BASE, Overview of Zookeeper and Paxos, Cassandra

Apache Hadoop [8Hrs]

History of Hadoop, Overview Apache Hadoop and Apache Spark, Hadoop Storage Framework – Hadoop Distributed File System (HDFS), HDFS Architecture, HDFS Concept, Hadoop Data Processing Framework – Map Reduce, Anatomy of a Map Reduce Job Run, Failures, Job Scheduling, Shuffle and Sort, Task Execution, Map Reduce Types and Formats, Map Reduce Features, MapReduce Programming Model with Spark.

Introduction to Big Data Streaming [7Hrs]

Big Data Pipelines for Real-Time computing, Introduction to Apache Spark Streaming, Kafka, Streaming Ecosystem, Spark Components, Resilient Distributed Dataset and data frames.

Apache PIG [7Hrs]

What is ETL, Introduction to Apache PIG, Execution Modes of PIG, Comparison of PIG with SQL and No-SQL Databases, PIG Data Types, Data Models in PIG, Grunt, PIG Latin, Overview of PIG User Defined Functions
Content

Apache HIVE [6Hrs]

Self-study **[12Hrs]**

Big Data Insights Using R and ML Algorithms: Regression Model, Clustering, Collaborative Filtering, Associate Rule Making, Decision Tree

Textbooks

1. Big Data Analytics, Seema Acharya, SubhasiniChellappan, Second Edition, 2019, Wiley India Pvt.Ltd, ISBN 978-81-2657-951-8.
2. Hadoop: The Definitive Guide, Tom White, Fourth Edition, 2015, O'Reilly, ISBN 978-93-5213-067-2.

Reference Books

1. Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics, Bill Franks, First Edition,2012, John Wiley & sons, ISBN: 978-1-118-20878-6
2. Mining of Massive Datasets, Jure Leskovec, Anand Rajaraman and Jeffrey David Ulman, Second Edition 2016, Dreamtech Press, ISBN - 978-13-1663-849-1
3. Analytics in a Big Data World: The Essential Guide to Data Science and its Applications, Bart Baesens, First Edition 2014, Wiley, ISBN : 1118892704
4. Big Data Glossary, Pete Warden, First Edition 2011, O'Reily, ISBN
5. Harness the Power of Big Data The IBM Big Data Platform ",Paul Zikopoulos ,Dirk DeRoos , Krishnan Parasuraman , Thomas Deutsch , James Giles , David Corigan , Annotated Edition 2012, Tata McGraw Hill Publications, ISBN 978-0071808170
6. Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses, Michael Mineli, Michele Chambers, Ambiga Dhiraj, First Edition 2013, Wiley Publications, ISBN 978-1118147603
7. Big Data Analytics: Disruptive Technologies for Changing the Game, ArvindSathi, MC Press, 2012, ISBN 1583473807

Web references

1. IBM Skills Network – Big Data, Data Engineering and Data Science MicroMasters Programs
2. HKUST – Big Data Technology Course Resources
3. Introduction to Big Data: Aligning Strategy, Analytics and Operations (Coursera)
4. Mastering BigQuery (Starweaver)
5. Big Data Engineering Bootcamp with Hadoop, Spark, Kafka, GCP & Azure (Udemy)

(HON) Fundamentals of Information and Coding Theory

Teaching Scheme

Lectures: 3 Hrs/ Week

Self-Study: 1 Hrs/week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Gainsubstantialknowledgeofinformationandentropy,andtheiruseininformation theory
2. Learn principles data compression
3. Understand techniques of design and performance evaluation of error correcting codes
4. Design and develop solutions for technical issues related to information coding
5. Get exposure to emerging topics in information theory, coding and compression.

Course content

Introduction to Information Theory **[8 Hrs]**

Introduction to Information Theory and Coding, Definition of Information Measure and Entropy, Information rate, Adjoint of an Information Source, Joint and Conditional Information Measure, Properties of Joint and Conditional Information Measures and A Morkov Source, Asymptotic Properties of Entropy and Problem Solving in Entropy.

Introduction to Coding **[8 Hrs]**

Classification of codes, Kraft-McMillan inequality, Source coding theorem, Shannon- Fano coding, Huffman coding, Mutual Information - Discrete memory less channels – BSC, BEC – Channel capacity, Shannon limit.

Data Compression **[7 Hrs]**

Adaptive Huffman Coding, Arithmetic Coding, LZW algorithm, Perceptual coding, Masking techniques, Channel Vocoder, Linear Predictive Coding, Video Compression and H.261.

Network Coding **[7 Hrs]**

The Buttery Network, Wireless and Satellite Communications, Source Separation, the Max-Flow Bound, Single-Source Linear Network Coding..

Error Control Coding: Block Codes **[6 Hrs]**

Definitions and Principles: Hamming weight, Hamming distance, Minimum distance decoding- Single parity codes, Hamming codes, Linear block codes, Cyclic codes – Syndrome calculation, Encoder and decoder – CRC

Error Control Coding: Convolutional Codes **[6 Hrs]**

Convolutional codes–code tree, trellis, state diagram-Encoding–Decoding: Sequential search and Viterbi algorithm.

Self-study **[12 Hrs]**

Extension of An Information Source and Markov Source, Extended Huffman coding, Psychoacoustic model, Acyclic Networks, Repetition codes, Principle of Turbo coding

Textbooks

1. T.M.CoverandJ.A.Thomas,“ElementsofInformationTheory”,John Wiley &Sons,second edition
2. RanjanBose,“InformationTheory,CodingandCryptography”,2E,Tata- McGraw Hill,second edition
3. MuralidharKulkarniandK.S.Shivaprakasha,“InformationTheoryand Coding”, WileyIndia Pvt Ltd

Reference Books

1. RaymondW.Yeung,"InformationTheoryandNetworkCoding",Springer,2008, ISBN: 978-0-387-79234-7, 978-0-387-79233-0, 978-1-4419-4630-0.

Web references

1. https://nptel.ac.in/courses/117101053
2. https://ocw.mit.edu/courses/6-441-information-theory-spring-2016/
3. https://onlinecourses.nptel.ac.in/e-learning/preview/noc20_ee94

(HON) Ethical Hacking

Teaching Scheme

Lectures: 3 Hrs/ Week

Laboratory: 2 Hrs/ Week

Self-Study: 1 Hrs/week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Laboratory: CIE: 100 Marks

Course outcomes

Students will be able to:

1. Identify legal and ethical issues related to vulnerability and penetration testing.
2. Report on the strengths and vulnerabilities of the tested network.
3. Exploit the vulnerabilities related to computer system and networks using state of the art tools and technologies
4. Execute a penetration test using standard hacking tools in an ethical manner
5. Evaluate best practices in security concepts to maintain confidentiality, integrity and availability of computer systems

Course content

Introduction to Hacking **[10 Hrs]**

Ethical hacking process, Hackers behaviour & mindset, Maintaining Anonymity, Hacking Methodology, Information Gathering, Active and Passive Sniffing, Physical security vulnerabilities and countermeasures. Internal and External testing. Preparation of Ethical Hacking and Penetration Test Reports and Documents.

Social Engineering and Attacks **[10 Hrs]**

Social Engineering attacks and countermeasures. Password attacks, Privilege Escalation and Executing Applications, Network Infrastructure Vulnerabilities, IP spoofing, DNS spoofing, Wireless Hacking: Wireless footprint, Wireless scanning and enumeration, Gaining access (hacking 802.11), WEP, WPA, WPA2.

Application Vulnerabilities and Attacks **[10 Hrs]**

DoS attacks. Web server and application vulnerabilities, SQL injection attacks, Vulnerability Analysis and Reverse Engineering, Buffer overflow attacks. Client-side browser exploits, Exploiting Windows Access Control Model for Local Elevation Privilege. Exploiting vulnerabilities in Mobile Application

Introduction to Metasploit **[10 Hrs]**

Metasploit framework, Metasploit Console, Payloads, Metrpreter, Introduction to Armitage, Installing and using Kali Linux Distribution, Introduction to penetration testing tools in Kali Linux. Case Studies of recent vulnerabilities and attacks

Self-study **[12 Hrs]**

Laws, Ethics, and Compliance in Ethical Hacking, Penetration Testing Methodologies Vulnerability Assessment vs. Penetration Testing, information gathering using publicly available sources, OSINT frameworks and methodologies

Textbooks

1. Ethical Hacking and Penetration Testing Guide", Baloch R, CRC Press, 2015.
2. "Hacking for Dummies " Beaver, K., Third edition John Wiley & Sons, 2013

Reference Books

1. "Network Forensics Tracking Hackers through Cyberspace ", Davidoff, S. and Ham, J., Prentice Hall, 2012.
2. "Computer Forensics JumpStart", Michael G. Solomon, K Rudolph, Ed Tittel, Broom N., and Barrett, D, Willey Publishing Inc, 2011

Web references

1. https://onlinecourses.nptel.ac.in/e-learning/preview/noc22_cs13
2. https://www.offsec.com/metasploit-unleashed/#metasploit-unleashed--free-ethical-hacking-course
3. https://tryhackme.com/

Laboratory assignments

Suggested List of Assignments:

1. Perform passive reconnaissance on a target organization using publicly available information.
2. Discover active hosts and services in a controlled network.
3. Identify vulnerabilities in a test system.
4. Evaluate password strength and security policies.
5. Identify common web application vulnerabilities in a deliberately vulnerable application.
6. Understand how improper input validation can affect database security.
7. Examine client-side input validation weaknesses.

(IDP) Fundamentals of Software Engineering

Teaching Scheme

Lectures: 3 Hrs/ Week

Self-Study: 1 Hrs/week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Course outcomes

Students will be able to:

1. Understand fundamental concepts of the System Development Lifecycle (SDLC) and illustrate their application in software engineering contexts
2. Construct user interface prototypes for real-world scenarios by applying appropriate analysis and design methods.
3. Formulate and assess procedures to assure product quality and maintainability before and after deployment
4. Demonstrate acquired knowledge to develop transferable skills across diverse industrial and commercial domains

Course content

Software Development process **[6 Hrs]**

Software Engineering basics, Software Crisis and Myths, Software Process and development, Software life cycle and Models, Analysis and comparison of various models, agile process

Requirement Engineering **[6 Hrs]**

Requirements Engineering, requirement engineering process, Introduction to Analysis model.

System Architecture and Design Overview **[8 Hrs]**

Architecture 4+1 view, architecture styles, Design process, quality concepts, Analysis model to Design Model transformation, Standardisation using UML.

Software Metrics **[8 Hrs]**

Introduction to Software Metrics, Size-oriented metrics and function point metrics. Effort and cost estimation techniques -LOC-based and Function-point based measures - The COCOMO model.

Testing and Maintenance **[6 Hrs]**

Validation and Verification activities, Testing Principles and strategies, Testing levels & types- White Box & Black Box Testing, Maintenance, Types of Maintenance.

Quality Assurance **[6 Hrs]**

Quality Standards Models Overview - ISO, Process improvement Models: CMM & CMMI. Quality Control and Quality Assurance.

Self-study **[12 Hrs]**

Laws, Ethics, and Compliance in Ethical Hacking, Penetration Testing Methodologies Vulnerability Assessment vs. Penetration Testing, information gathering using publicly available sources, OSINT frameworks and methodologies

Textbooks

1. “Pressman R., "Software Engineering, A Practitioners Approach", 6th Edition, Tata McGraw Hill Publication, 2004, ISBN 007-124083-124083-7.
2. “G. Booch, J. Rumbaugh and I. Jacobson. The Unified Modeling Language User Guide, Addison Wesley, 1999

Reference Books

1. Shari Pfleeger, “Software Engineering”, 2nd Edition. Pearson‟s Education, 2001.
2. Ian Sommerville, “Software Engineering”, 6th Edition, Addison-Wesley, 2000
3. Pankaj Jalote, “An Integrated Approach to Software Engineering”, Narosa publication house
4. Fred Brooks, “Mythical Manmonths”, www.cs.drexel.edu/~yfcai/CS4517/.

Web references

1. nptel.ac.in/courses/106105182
2. CS 682 Software Engineering, IITB

(IDP) Fundamentals of Programming and Algorithms

Teaching Scheme

Lectures: 3 Hrs/ Week

Laboratory: 2 Hrs/ Week

Self-Study: 1 Hrs/week

Evaluation Scheme

Theory: MSE: 30 Marks, TA: 20 Marks, ESE: 50 Marks

Laboratory: CIE: 100 marks

Course outcomes

0. Understand fundamental data structures and their applications.
1. Develop problem-solving and computational thinking skills.
2. Apply programming concepts to solve engineering problems.
3. Analyze algorithm efficiency using complexity measures.
4. Design solutions using appropriate algorithmic paradigms.

Course content

Programming Fundamentals and Problem Solving: [6 Hrs]

Problem-solving techniques and computational thinking, Algorithms and flowcharts, Program development cycle, Data types, variables, constants, Operators and expressions, Input/output operations, Conditional statements, Iterative constructs (for, while, do-while), Debugging and testing basics, Functions and modular programming Parameter passing techniques, Scope and lifetime of variables, Recursive functions, String handling and manipulation, File handling basics, Error and exception handling concepts

Fundamental Data Structures: [6 Hrs]

Abstract Data Types (ADT), Arrays and multidimensional arrays, Structures and pointers, Dynamic memory allocation, Singly linked lists, Doubly linked lists, Circular linked lists, Stacks and queues, Applications of stacks and queues

Algorithm Analysis and Design Fundamentals:

Characteristics of algorithms, Time and space complexity, Asymptotic notations: Big-O, Omega, Theta, Best, average and worst-case analysis, Recurrence relations, Brute force approach, Divide and conquer strategy

Searching and Sorting Algorithms: [6 Hrs]

Linear search, Binary search, Bubble sort, Selection sort, Insertion sort, Merge sort, Quick sort, Heap sort, Performance comparison of algorithms

Trees and Graphs [6 Hrs]

Binary trees and traversals, Binary Search Trees, Graph representations, Breadth First Search (BFS), Depth First Search (DFS), topological sort, Dijikstra's Single source shortest paths

Advanced Algorithmic Techniques: [8 Hrs]

Dynamic Programming: Introduction, All pairs shortest paths, 0-1 Knapsack, Traveling salesperson problem, Chained matrix multiplication, Longest common subsequence, Bellman-Ford algorithm. Back tracking: subset sum problem, Hamiltonian cycle, Graph coloring

Self-study [2 Hrs]

Stack applications in expression evaluation and function calls, Growth of functions and algorithm efficiency, Algorithm visualization tools, Applications of trees in file systems and databases, Analysis of algorithms used in Google Maps navigation.

Textbooks

1. T. W. Pratt , "Programming Languages", 4th Edition ,Prentice-Hall Of India, ISBN 9780130287199.
2. Thomas Cormen, Charles Leiserson, Ronald Rivest and Cliford Stein, “Introduction to Algorithms”, PHI, 3rd edition, ISBN-13: 978-8120340077
3. E. Horowitz, S. Sahni, S.Anderson-freed, “Fundamentals of Data Structures in C”, Second Edition, University Press, ISBN 978-81-7371-605-8
4. B. Kernighan, D. Ritchie, “The C Programming Language”, Prentice Hall of India, Second Edition, ISBN 81-203-0596-5

Reference Books

- Ellis Horowitz, S. Sahni, D. Mehta “Fundamentals of Data Structures in C++”, Galgotia Book Source, New Delhi 1995 ISBN 16782928
- Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, “Fundamentals of Computer Algorithms”, Universities Press, 2nd edition (2008) , ISBN-13: 978-8173716126

Web references

1. https://nptel.ac.in/courses/106105164
2. https://nptel.ac.in/courses/106104128
3. https://nptel.ac.in/courses/106102064

Laboratory assignments

1. Recursive implementation of factorial, Fibonacci, and Tower of Hanoi.
2. String manipulation and pattern matching programs.
3. File handling operations: create, read, update, and append records.
4. Implementation of one-dimensional and two-dimensional array operations.
5. Dynamic memory allocation using pointers.
6. Implementation of singly linked list operations.
7. Implementation of doubly linked list and circular linked list operations.
8. Stack implementation-using arrays and linked lists.
9. Implement a queue (that is write queue.c and queue.h only) of characters, such that on an enqueue, the char is added at the end of queue, and on a dequeue the first element is taken out, but the queue uses only a 'head' pointer and not a 'tail pointer.
10. Write the following functions for a binary search tree implementation: Searches the maximum value in the tree, preorder traversal without using recursion, Search the string in the tree and returns a pointer to the node, print the binary tree so that it looks like a tree.
11. Implementation and analysis of linear, binary search algorithms and comparison of Bubble, Selection, and Insertion Sort.
12. Implementation and comparison of Merge Sort, Quick Sort, and Heap Sort.
13. Greedy algorithm implementation: Activity Selection and Fractional Knapsack.
14. Dynamic Programming: Fibonacci and Longest Common Subsequence, Traveling salesperson problem

Laboratory instructions or disclaimers if any

Guidelines for Laboratory /TW Assessment:

1. Continuous assessment of laboratory work is done based on overall performance & Laboratory performance of students.
2. Each Laboratory assignment assessment should assign grade/marks based on parameters with appropriate weightage.
3. Suggested parameters for overall assessment as well as each Laboratory assignment assessment include- timely completion, performance, innovation, efficiency, punctuality, and neatness.

Guidelines for Laboratory Conduction

1. Operating System recommended: - 64-bit Open-source Linux or its derivative

2. Programming tools recommended: - C, C++, Python