COEP Technological University

Department of Computer Science and Engineering

Curriculum Structure & Detailed Syllabus (UG Program)

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

(Effective from: A.Y. 2025-26)

Sr. No.	Item	Page No
1	Programme Education Objectives (PEOs), Programme Outcomes (POs) and Programme Specific Outcomes (PSOs)	2
2	Correlation between PEOs, PSOs and POs	3
3	List of Abbreviations	4
4	Curriculum Structure, List of: Electives, Minor, Honor courses	6
5	Detailed Syllabus	8

B.Tech. Computer Science and Engineering

Programme Educational Objectives (PEOs):

I. To create graduates with sufficient capabilities in computer engineering who can become researchers, entrepreneurs and software professionals to satisfy the needs of the core industry, research, academia and society at large.

II. To build the ability to continuously learn the latest trends in computer engineering and engage in a lifelong learning process.

III. To build engineers aware of professional ethics of the software Industry, and equipped with basic soft skills essential for working in community and professional teams.

Programme Outcomes (POs):

At the end of the program, the graduates will

1. **Computer engineering knowledge:** Apply the knowledge of mathematics, science, computer engineering fundamentals, and emerging fields of computer engineering to the solution of complex real-life problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex computer engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and computer engineering sciences.
3. **Design/development of solutions:** Design solutions for complex computer engineering problems and design system components or processes that meet the specified needs considering public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern computer engineering and IT tools including FOSS tools.
6. **Social responsibility:** Apply reasoning informed by the contextual knowledge to assess social, health, safety, legal and cultural issues and the consequent responsibilities.
7. **Environment and sustainability:** Understand the impact of the professional computer engineering solutions in socio-environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Demonstrate knowledge and practice of engineering ethics.
9. **Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary, multi-cultural settings.
10. **Communication:** Communicate effectively with the engineering community and with society at large, demonstrating ability to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the computer engineering, finance and management principles.
12. **Life-long learning:** Recognize the need for, and ability to engage in independent and life- long learning

Programme Specific Outcomes (PSOs)

Students will be able to

1. Demonstrate competence in Programming Technologies.
2. Design, implement, test software solutions in core Computer Engineering areas including Computer Networks, Databases, Systems Software, Computer Architecture, Artificial Intelligence, Software Engineering
3. Acquire and demonstrate skills in emerging areas like Information Security, Data Science, Natural Language Processing, Cloud Computing, etc.

Correlation between the PEOs and the POs

PO→ PEO↓	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
I	▪	▪	▪	▪	▪					▪	▪	▪
II	▪	▪	▪	▪	▪	▪	▪	▪	▪	▪	▪	▪
III						▪	▪	▪	▪	▪	▪	

Correlation between the PEOs and the PSOs

PSO→ PEO↓	I	II	III
I		▪	▪
II	▪		▪
III			

List of Abbreviations

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

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	50	50
02	PCC	CTS-24007	Principles of Programming Languages	3	0	0	1	3	30	20	50	--	--
03	PCC	CTS-24006	Data Structures and Algorithms	3	0	2	1	4	30	20	50	50	50
04	OE	As per course	Open Elective-1	2	0	0	1	2	30	20	50	--	--
05	HSMC	As per course	Indian language	2	0	0	0	2	CIE: 100			--	--
06	VEC	AS-24002	Environment studies	1	0	0	2	1	CIE: 100			--	--
07	HSMC	HS-24001	Entrepreneurship	2	0	0	0	2	CIE: 100			--	--
08	CEA	AS-24004	Community Engagement Activity (CEA)/Field Project	0	0	0	0	2	--			100	--
			Total	**16**	**00**	**04**	**06**	**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	50	50
02	PCC	CTS-24009	Computer Organization	3	0	0	1	3	30	20	50	--	--
03	PCC	CTS-24011	Theory of Computation	2	1	0	1	3	30	20	50	--	--
04	VEC	CTS-24012	Constitution of India and Universal Human Values	1	0	0	2	1	CIE: 100			--	--
05	HSMC	HS-25005	Economics	2	0	0	0	2	CIE: 100			--	--
06	OE	As per course	Open Elective-2	2	0	0	1	2	30	20	50	--	--
07	VSEC	CTS-25006	Development Tools Laboratory	0	0	2	0	1	--			100	--
08	MDM	As per course	Multidisciplinary Minor-1	3	0	0	1	3	30	20	50	--	--
09	HSMC	<TBD>	Design Thinking	1	0	0	1	1	CIE: 100			--	--
			Total	**17**	**01**	**06**	**08**	**20**					

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	50	50
02	PCC	CTS-24007	Principles of Programming Languages	3	0	0	1	3	30	20	50	--	--
03	PCC	CTS-24006	Data Structures and Algorithms	3	0	2	1	4	30	20	50	50	50
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	0	2	CIE: 100			--	--
06	VEC	AS-24002	Environment Studies	1	0	0	2	1	CIE: 100			--	--
07	HSMC	HS-24001	Entrepreneurship	2	0	0	0	2	CIE: 100			--	--
09	PCC	MA-24001	Matrices Differential Calculus and Probability	3	0	0	1	3	30	20	50	--	--
			Total	**19**	**00**	**04**	**08**	**21**					

[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	50	50
02	PCC	CTS-24009	Computer Organization	3	0	0	1	3	30	20	50	--	--
03	PCC	CTS-24011	Theory of Computation	2	1	0	1	3	30	20	50	--	--
04	VEC	CTS-24012	Constitution of India and Universal Human Values	1	0	0	2	1	CIE: 100			--	--
05	HSMC	HS-25005	Economics	2	0	0	0	2	CIE: 100			--	--
06	OE	As per course	Open Elective-2	2	0	0	1	2	30	20	50	--	--
07	VSEC	CTS-25006	Development Tools Laboratory	0	0	2	0	1	--			100	--
08	MDM	As per course	Multidisciplinary Minor-1	3	0	0	1	3	30	20	50	--	--
09	HSMC	<TBD>	Design Thinking	1	0	0	1	1	CIE: 100			--	--
10	HSMC	HS-24007	Communication Skills	2	0	0	0	2	CIE: 100			--	--
11	PCC	<TBD>	Discrete Structures	2	0	0	1	2	30	20	50	--	--
			Total	**21**	**01**	**06**	**09**	**24**					

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

Semester	Course Code	Course Title	L	T	P	Cr
IV	MDM-01	Data Structures, Files and Algorithms	3	0	0	3

OPEN ELECTIVES LIST
(offered to other departments)

Semester	Course Code	Course Title	L	T	P	Cr
III	OE-01	Data Analytics	2	0	0	2
IV	OE-02	Fundamentals of Operating Systems/ Fundamentals of Algorithms	2	0	0	2

(PCC-02) Microprocessors

Teaching Scheme
Lectures: 3 Hr/ week
Labs: 2 Hr/ week
Self-Study: 1 Hr/Week

Evaluation Scheme
Theory: MSE:30 Marks, TA:20 Marks ESE:50 Marks
Laboratory: CIE:100 Marks

Course Outcomes

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 Contents

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

Programming with 8086: 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.

[8 Hrs]

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

[4 Hrs]

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

Serial Communication, DMA mode: Serial I/O, Asynchronous and synchronous mode, design of serial communication using 8251 USART and RS232C drivers, Direct Memory Access and Interfacing of 8237 **[4 Hrs]**

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

Self Study:

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

[12 Hrs]

Note:

- All the Course outcomes 1 to 3 will be judged by 75% of the questions and outcomes 4, 5 and 6 will be judged by 25 % of questions.
- To measure CO3 and CO4, some questions may be based on self-study topics

Text Book

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

Reference Books

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

Suggested List of Assignments:

Experiment

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.

E) Find opcodes for the instructions in C above using instructions templates of 8086.

Experiment

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

Experiment

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.

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.

Experiment

Demonstrate 8255 for strobed input and strobed output in interrupt driven mode.

Experiment

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.

Experiment 7:

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.

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.

Principles of Programming Languages

Teaching Scheme
Lectures: 3 Hr/ 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. 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 Contents

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

[5 Hrs]

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

[5 Hrs]

Syntax and Semantics Lexical and Syntax Analysis Names, Bindings, and Scopes, Type checking, Strong Typing. Type Conversions. Short-Circuit Evaluation.

[8 Hrs]

Statement-Level Control Structures Subprograms: Introduction. Fundamentals of Subprograms. Design Issues for Subprograms. Local Referencing Environments. Parameter-Passing Methods.

[6 Hrs]

Exception Handling: 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

[6 Hrs]

Programming Paradigms: 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 **[10 Hrs]**

Self Study:
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.

[12 Hrs]

Text Books

- Schesta R., "Concepts Of Programming Languages", 4th Edition, Pearson Education, ISBN- 81-7808-161-X

- T. W. Pratt , "Programming Languages", 4th Edition ,Prentice-Hall Of India, ISBN 9780130287199.

Reference Books

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

(PCC-04) Data Structures and Algorithms

Teaching Scheme
Lectures: 3Hr/Week
Labs: 2 Hr/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 Contents

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 [8 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 [6 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 [6 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]**

Text Books

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

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-01) Data Analytics

Teaching Scheme
Lectures: 2 Hr/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 Contents

Introduction to Data Analytics: 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.

[6 Hrs]

Introduction to Python Libraries-NumPy: 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

[6 Hrs]

Pandas Library: 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.

[7 Hrs]

Data Visualization with Matplotlib and Seaborn libraries: 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.

[7 Hrs]

Self Study

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. **[12 Hrs]**

Text Books

- 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
- Bharti Motwani, "Data Analytics using Python ", Publisher: Wiley, ISBN: 8126502959

Reference Books

- Vincent Granville, "Developing Analytic Talent -Becoming a Data Scientist", Publisher: Wiley, 2014, ISBN: 9781118810095.

- Glenn J. Myatt, Wayne Johnson, “Making Sense of Data I”, Second Edition, Publisher: Wiley, ISBN 978-1-118-40741-7
- Glenn J. Myatt, Wayne Johnson, “Making Sense of Data II”, Publisher: Wiley, ISBN 978-0- 470-22280-5

(PCC-05) Object Oriented Programming & Design

Teaching Scheme:
Lectures: 3 Hr/Week
Practical: 2 Hr/Week
Self-Study: 1 Hr/Week

Examination Scheme:
Theory: MSE:3 0Marks, 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 Contents

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

[8 Hrs]

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

[8 Hrs]

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

[6 Hrs]

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

[6Hrs]

Exception Handling & Multi-Threaded Programming: Exception handling, Exception types, Concurrent Programming, Basic Concepts of Concurrent Programming, Threads.

[6 Hrs]

Introduction to OOAD: 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

[6 Hrs]

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

[12 Hrs]

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 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.
2. Write a program to implement the following. Create a base class called person consisting 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.
3. 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.
4. Write a menu driven to read, add, subtract, multiply, divide and transpose two matrices.
5. 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).
6. 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.
7. 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.
8. You may then freely raise or lower the extension. The latch produces a clacking sound as it

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.

9. 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.

Text Books

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

Reference Books

- Herbert Schilt, “JAVA Complete Reference”, 7th Edition, Tata McGraw Hill, ISBN: 9780070636774
- Scott W. Ambler ,The Elements of UML (TM) 2.0 Style, ISBN- 978-0521616782
- 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
- Eckel B., "Thinking in Java", 3rd Edition, Pearson Education, 2012
- E. Balagurusamy, Object Oriented Programming with C++, 6th Edition, McGraw Hill, ISBN-10: 125902993X

(PCC-06) Computer Organization

Teaching Scheme
Lectures: 3 Hr/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. 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 Contents

CPU Architecture: 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

[6Hrs]

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

[6Hrs]

Memory: 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

[8Hrs]

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

[7Hrs]

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

[7Hrs]

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

[5Hrs]

Self Study
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. **[12Hrs]**

Text Books

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

Reference Books

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

(PCC-06) Theory of Computation

Teaching Scheme
Lectures: 2 Hr/Week
Tutorial: 1 Hr /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. 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 Contents

Finite Automata: 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.
[10 Hrs]

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

Context-Free Languages: Context-free Grammars: Formal definition, Designing context-free grammars, Parse Trees, Ambiguity, Chomsky Normal Form. Pushdown Automata: Formal definition, Examples.
[6 Hrs]

Turing Machines and Undecidability: 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.
[6 Hrs]

Self study: 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.
[12 Hrs]

Text Books:
- Michael Sipser, "Introduction to the Theory of Computation", 3rd Edition, 2013, Cengage Learning Publications, ISBN-13: 978-1133187790.
- 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:
- E. V. Krishnamurthy, "Theory of computer science", 2004, Affiliated East Press Publications, ISBN-10: 038791255X / ISBN-13: 978-0387912554.
- Dexter C. Kozen, "Automata and Computability", 1997, Springer Verlag Publications, ISBN 0-387-94907-0.
- Harry Lewis, Christos H. Papadimitriou, "Elements of the Theory of Computation", 2nd Edition, 1997, Prentice-Hall Publications, ISSN 0891-4516.
- 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.

(OE-02) Fundamentals of Operating Systems

Teaching Scheme:	**Evaluation Scheme:**
Lectures: 2 Hr/Week	MSE: 30 marks
Self Study: 1 Hr/Week	TA: 20 marks
	ESE: 50 marks

Course Outcomes:

Upon completion of this course, participants 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 Contents

Introduction to Operating System (OS): 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.

[6 Hrs]

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

[6 Hrs]

Process Synchronization: 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.

[6 Hrs]

Memory Management: 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

[6 Hrs]

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

[6 Hrs]

Self Study: Case Studies of Linux OS: Design Principles, Kernel Modules, Process Management, Scheduling, Memory Management, File Systems.

[12 Hrs]

Textbooks

- "Operating System Concept" by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, 10th Edition, 2021, Wiley, ISBN-978-1-119-32091-3.
- "Modern Operating System", by Andrew S. Tanenbaum, 5th Edition, 2022, Pearson Education, ISBN 13: 978-0137618873.

(OE-02) Fundamentals of Algorithms

Teaching Scheme	**Evaluation Scheme**
Lectures: 2 Hr/Week	**Theory:** CIE: 100 Marks
Self-Study: 1 Hr/Week	

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 Contents

Basics of Algorithm: 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.

[6 Hrs]

Design Techniques-I: 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.

[6 Hrs]

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

[5 Hrs]

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

[5 Hrs]

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

[4 Hrs]

Selected Algorithms: String Matching: The naïve string-matching algorithm, The Robin-Karp algorithm, The Knuth- Morris-Pratt algorithm. **[4 Hrs]**

Self Study:

Algorithmic Problem Solving Platforms:Activity: Solve algorithmic problems on platforms like LeetCode or HackerRank. **[8 Hrs]**

Text Books

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

(VSEC-01) Development Tools Laboratory

Teaching Scheme	**Suggested Evaluation Scheme**
Labs: 2 Hr/Week	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 Contents

LaTEX: 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: 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: 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

Suggested List of 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.

References:

LaTeX

- Leslie Lamport, "LaTeX: A document preparation system", User's guide and reference manual, 2nd Edition, 1994, by Addison-Wesley Professional. ISBN 0201529831, 9780201529838
- 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/
- Introduction to LaTEX, MIT
http: //web.mit.edu/rsi/www/pdfs/new-latex.pdf
- A simple guide to LaTeX - Step by Step
https: //www.latex-
tutorial.com/tutorials/

Shell

- Bash Guide for Beginners: https://www.tldp.org/LDP/Bash-Beginners-Guide/Bash- Beginners-Guide.pdf
- Bash Reference Manual https://www.gnu.org/software/bash/manual/bash.pdf
- Tutorials on Shell Programming https://www.shellscript.sh/
https://www.tutorialspoint.com/unix/shell_scripting.htm

GIT

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

(MDM-02) Data Structures, Files and Algorithms

Teaching Scheme:
Lectures: 3 Hr/week

Examination Scheme:
Theory: MSE:30 Marks, TA:20, 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 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 Contents

Introduction: 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.

[4 Hrs]

Arrays, Searching and Sorting: 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.

[8 Hrs]

Files and File Handling: 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()

[6Hrs]

Stacks and Queues: 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.

[6 Hrs]

Linked Lists: 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.

[8 Hrs]

Trees: 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.

[8 Hrs]

Self Study:
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

[8 Hrs]

Text Books

- E. Horowitz, S. Sahni, S.Anderson-freed, "Fundamentals of Data Structures in C", Second Edition, University Press, ISBN 978-81-7371-605-8
- B. Kernighan, D. Ritchie, "The C Programming Language", Prentice Hall of India, Second Edition, ISBN 81-203-0596-5
- 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

- Ellis Horowitz, S. Sahni, D. Mehta "Fundamentals of Data Structures in C++", Galgotia Book Source, New Delhi 1995 ISBN 16782928
- 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

(HSMC) Design Thinking

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

Evaluation Scheme
CIE: 100 Marks

Course outcomes

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: Testing, Feedback, RE-Design & Re-Create **[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/Journals

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-14) Discrete Structures

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. 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 **[6 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/Journals

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.