Autonomous Program Structure
Second Year B. Tech. Third Semester
Computer Engineering
Academic Year: 2021-2022 Onwards

Course Code	Course Title	Teaching Scheme Hours /Week			Examination Scheme				Marks	Credits
		Lecture	Tutorial	Practical	In Semester	End Semester	Oral	Practical		
20HS301	Universal Human Values II	2	1	0	50	50	0	0	100	3
20CE301	Programming Paradigms	3	0	0	50	50	0	0	100	3
20CE302	Data Structures	3	1	0	50	50	0	0	100	4
20CE303	Discrete Mathematics	3	1	0	50	50	0	0	100	4
20CE304	Digital Systems and Computer Organization	3	1	0	50	50	0	0	100	4
20CE302L	Data Structures Laboratory	0	0	4	25	0	0	25	50	2
20CE304L	Digital Systems Laboratory	0	0	2	25	0	25	0	50	1
20CE305L	Programming Skills Development-I Laboratory	0	0	4	25	0	25	0	50	2
20AC301	Audit Course	0	0	1	0	0	0	0	0	No Credits
	Total	**14**	**4**	**11**	**325**	**250**	**50**	**25**	**650**	**23**
	Grand Total		**29**			**650**				**23**

[illegible]

APPROVED BY
Secretary Governing Body
MKSSS's Cummins College of Engineering
For Women, Pune-411052

[illegible]

APPROVED BY
Chairman Governing Body
MKSSS's Cummins College of Engineering
For Women, Pune-411052

M.K.S.S.S. Cummins College of Engg. for Women ★ KARVENAGAR Pune -52

20CE 301 Programming Paradigms

Teaching Scheme	**Examination Scheme**
Lectures: 3 Hours / Week	In Semester: 50 Marks
	End Semester: 50 Marks
	Credits: 3

Course Objectives:

To facilitate the learner to

1. Explore the major programming paradigm
2. Introduce frameworks for specifying and reasoning about programming languages.
3. Understand principles and techniques involved in design and implementation of modern programming languages.
4. To provide an exposure to core concepts and principles of contemporary programming languages

Course Outcomes:

After completion of the course, students will be able to

1 Describe characteristics and design principles of imperative programming language paradigms

2 Demonstrate different forms of declaration, typing, binding, visibility, scoping, and lifetime management for various programming language constructs

3 Describe characteristics and design principles of declarative programming language paradigms

4 Choose a language or paradigm suitable for solving a particular problem

Unit I: Introduction to Programming Languages (6)

Role of programming languages, towards high level programming languages, problems of Scale, programming paradigms, Language implementation, Expression notations, Abstract syntax trees, Lexical syntax, Context free grammar, variants of grammars, The need of structured programming, syntax directed control flow, design considerations, handling special cases in loops, programming with invariants, control flow in C.

Unit II: Data Representation and procedure activation (10)

The role of Types, Basic Types, assignment and local state, the environment model of evaluation, Arrays, Records, Union and variant records, sets, pointers, two string tables, types and error checking, modelling with mutable data. Introduction to procedures, Parameter passing methods, Scope rules for names, nested scope in source text, activation records, Lexical scope in C and Pascal

Unit III: Functional Programming Paradigm: (9)

Elements of functional programming, a little language of expressions, types: values and operations, function declarations, approached to expression evaluation, lexical scope, type checking. Exploring a list, function declaration by Cases, Functions as first class values. ML/Scheme/Lisp: Implicit types, Data types, Exception handling in ML/Scheme/Lisp. Functional programming features in Python.

Unit IV: Logic Programming (8)

Computing with relations, Introduction to Prolog, Data Structures in Prolog, Programming Techniques, Control in Prolog, Cuts.

Unit V: Object Oriented Programming and Recent Advances in Programming (9)

Constructs of program structuring, information hiding, program design with modules, modules and defined types, Object oriented programming in Python. Recent developments in the world of programming. Overview of emerging programming languages – Elm, Rust, Kotlin, Go, Ruby, Scala, Swift etc.

Text Books:

1. Sethi R., "Programming Languages concepts & constructs", 2nd Edition, Pearson Education, ISBN 81 - 7808 - 104 - 0
2. Harold Abelson, Gerald Jay Sussman, Julie Sussman, "Structure and Interpretation of Computer Programs", 2nd Edition, ISBN 0-07-000484-6 (McGraw-Hill hardcover)

Reference Books:

1. Roosta S., "Foundations of Programming Languages", Thomson Brookes/Cole, ISBN 981 -243-141-1
2. Sebesta R., &quot;Concepts Of Programming Languages&quot;, Pearson Education, (10th Edition)(2014)
3. Allen Tucker, Robert Noonan, "Programming Languages: Principles and Paradigms", Tata McGraw Hill, (2nd edition),(2007)
4. Carlo Ghezzi, Mehdi Jazayeri, "Programming Language Concepts",3rd Ed, Wiley Publication ISBN : 978-81-265-1861-6.

Online/Web/Other References:

1. https://nptel.ac.in/courses/106/102/106102067/

20CE 302 Data Structures

Teaching Scheme	**Examination Scheme**
Lectures: 3 Hours / Week	In Semester: 50 Marks
Tutorial: 1 Hour / Week	End Semester: 50 Marks
	Credits: 4

Course Objectives:

To facilitate the learner to

1. Learn and understand representation, implementation and applications of data Structures
2. Choose and apply linear and non linear data structures for developing solutions for solving problems in various domains.
3. Demonstrate ability to use stack and queue data structures to solve problem
4. Understand and apply the concepts of hashing.
5. Analyze algorithms using time complexity analysis

Course Outcomes:

After completion of the course, students will be able to

1. Apply appropriate data structure to construct efficient algorithms to approach the problems.
2. Distinguish between various linear data structures based on their representations and applications.
3. Apply principles of data structures- stack and queue to solve computational problems.
4. Apply non linear data structures –Trees and Graphs to solve a problem.
5. Apply the concept of Hashing techniques for solving a problem.
6. Analyze algorithms using time and space complexity

Unit I: Introduction to Algorithms; Sorting and Searching (07)

Introduction to Algorithms, Pseudo code, Abstract Data Types (ADT): e.g. Arrays as ADT, Introduction to Data Structures, Frequency Count, Analyzing Algorithm using Frequency count, Time complexity of an Algorithm, Asymptotic notations, Best, Worst and Average case analysis of an Algorithm. Sorting: Bubble sort, Insertion sort, Quick Sort.
Searching: Linear Search, Binary Search. Time complexity analysis of sorting and searching

Algorithms. Case study: Timsort

Unit II: Linked List (07)

Concept of Linked List, Comparison of Sequential and Linked Organizations, Linked List using Dynamic Memory Management, Linked List as an ADT, Singly Linked List , Doubly Linke List, Circular Linked List operations. Time complexity analysis of Linked List operations.
Case study: Garbage collection

Unit III: Stack and Queue (07)

Stack as an ADT, Representation and Implementation of Stack using Sequential and Linked Organization, Applications of Stack- Simulating Recursion using Stack, Arithmetic Expression Conversion and Evaluation. Queue as an ADT, Representation and implementation of Linear Queue, Circular Queue, Priority Queue .Time complexity analysis of Stack and Queue operations. Time complexity analysis of algorithms using stack and queue data structures.
Case study: Priority queue in bandwidth Management

Unit IV: Trees (08)

Introduction to Non Linear Data Structure, Binary Trees, Types of Binary Trees, Properties of Binary Trees, Binary Tree as Abstract Data Type, Representation using Sequential and Linked Organization, Binary Tree creation , Recursive and Non Recursive Tree Traversals, Binary earch Tree and its operations, B Tree, Heap as ADT.
Case study: expression tree, Heap as priority queue.

Unit V: Graphs (07)

Basic Terminologies, Storage Representation, Graph Traversals, Graph as Abstract Data Type, Spanning Trees, Minimum Spanning Trees, Kruskal's Algorithm, Prim's Algorithm, Dijkstra's Single Source Shortest Path Algorithm. Time complexity analysis of graph algorithms.
Case study: Google maps.

Unit VI: Hashing (06)

General idea of Hashing, Hash Table, Hash function, Rehashing, Issues in Hashing, Collision Resolution Strategies: Linear Probing, Open addressing and Chaining. Time complexity analysis of hashing techniques.
Case study: Telephone dictionary.

Text Books:

1. Sartaj Sahani, "Data Structures, Algorithms and Applications in JAVA", Universities Press (2 nd edition).

2. Robert Lafore, "Data Structures Algorithms in JAVA", Techmedia,(1 st edition).
3. E. Horowitz, S. Sahni, D. Mehta, "Fundamentals of Data Structures in C++", Galgotia Publications ,(2 nd edition).

Reference Books:

1. Yedidyah Langsam, Moshe J Augenstein, Aron M Tenenbaum, "Data Structures using C and C++" , Pearson Education, (2 nd edition).
2. A. Aho, J. Hopcroft, J. Ulman, "Data Structures and Algorithms", Pearson Education, (2 nd edition).
3. Brassard and Bratley, "Fundamentals of Algorithmics", Prentice Hall India/Pearson Education, (2 nd edition) .
4. M. Weiss, "Data Structures and Algorithm Analysis in JAVA", Pearson Education (3rd edition), (2012).
5. Goodrich, Tamassia, Goldwasser, "Data Structures and Algorithms in JAVA", Wiley publication, (6th edition).
6. R. Gillberg, B. Forouzn, "Data Structures: A Pseudocode approach with C", Cenage Learning, (2 nd edition).

Online/Web/Other References:

1. https://nptel.ac.in/courses/106/102/106102064/
2. https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
3. http://web.stanford.edu/class/cs166/

Suggestive List of the Tutorial Assignments:

Following list of tutorials can be considered as guideline for designing tutorials:
Every student should perform 12 to 14 tutorials which will cover topics of all units mentioned in the syllabus of Data Structures. Tutorial assignments will enhance the understanding of the concepts of problem solving, algorithms and data structures. Students will perform practice exercise on data representation and corresponding implementation of the data structures. Students will get opportunity to develop their logic building abilities.

Following list of tutorials can be considered as guideline for designing tutorials:

1. Demonstration of a program implementation and execution using eclipse tool.
2. Design an algorithm for simple problems like GCD calculation, power calculation etc.
3. Calculate frequency count, time complexity of sample algorithmic constructs.
4. For given algorithms of array operation, write equivalent JAVA code.
5. Practice exercise on sorting and searching algorithms for set of predefined inputs.
6. Calculate time complexity of sorting algorithms using concept of frequency count.
7. Create a linked list and write algorithms for traversal, delete a node, add a node operations on a list.
8. Create a doubly or circular linked list and write algorithms for traversal, delete a node, add a node operations on a list.
9. Solve brain teaser based on recursive code snippets.
10. Demonstration on debugging techniques.
11. Select appropriate data structures and design algorithmic solution to given application.
12. Solve puzzles based on queue data structure
13. Practice exercise on creating binary tree and perform recursive and non recursive traversal of binary tree on given data
14. Creating binary search tree for given data and perform inorder, preorder, postorder traversal.
15. Practice exercise on searching and deleting data values from given binary search tree.
16. Design a solution for "company survey" about its products in an area. Choose the appropriate algorithm to complete the survey within short period and cover all houses under that area. Give justification for your answer and also analyze your algorithm for time complexity
17. Visualize various data structures using open source tools
18. Given the input data and hash function , show the result using hashing methods .
19. Use different hashing functions to hash given values.

20. Construct a Btree of order 3 by inserting numbers of given data
21. Practice exercise on Dijkstra's algorithms.
22. Practice exercise on graph MST algorithms.
23. Practice exercise on Heap data structures.

20CE 303 Discrete Mathematics

Teaching Scheme	Examination Scheme
Lectures: 3 Hours / Week	In Semester: 50 Marks
Tutorials : 1 Hour / Week	End Semester: 50 Marks
	Credits: 4

Course Objectives:
To facilitate the learner to

1. To understand Discrete Mathematics concepts and their significance in Computer Engineering.
2. To understand set theory, logic and apply reasoning to solve problems.
3. To solve problems based on algebraic systems, permutation and combination.
4. To solve problems on functions and relations and learn the basic properties of graphs and trees.

Course Outcomes:
After completion of the course, students will be able to

1. Solve problems on set using Venn diagram, theorems like the principle of inclusion-exclusion and mathematical induction and solve problems on relations and functions.
2. Apply concepts of propositional calculus for solving problems, formal proofs, reasoning and represent problems using first-order logic.
3. Apply the concepts of groups, permutations and combinations to solve problems.
4. Apply basic terminologies of graphs and trees to solve problems on paper.

Unit I: Sets and Mathematical Induction (07)
Significance of Discrete Mathematics in Computer Engineering, Sets, Subset, Universal Set, Empty Set, Algebra of Sets and Duality, Operations on Sets, Finite and Infinite Sets, Un Countably Infinite Sets, Multi-Sets, Power Set, Venn Diagram, Principle of Inclusion and Exclusion, Principle of Mathematical Induction, Applications of Set.

Unit II: Logic and Propositional Calculus (06)
Propositions, Logical connectives, Conditionals and Bi-Conditionals, Tautology, Contradiction, Contingency, Logical Equivalences, Algebra of Propositions, Logical Implications, Conjunctive and disjunctive Normal Forms, Rules of Inference, Predicates and Quantifiers, Nested Quantifiers, Applications of Logic.

Unit III: Groups, Rings and Permutations and Combinations (08)
Algebraic Systems, Groups, Semi Groups, Monoids, Subgroups, Introduction to Isomorphism, Homomorphism and Automorphism of groups, Cosets and Normal Subgroups, Introduction to Rings, Integral Domain and Field, Applications of Algebraic System, Introduction to Permutations and Combinations.

Unit IV: Relations and Functions (08)
Introduction to Relations, Product Sets, Pictorial Representation of Relations, Composition of Relations, Closure of Relations, Warshall's Algorithm, Properties of Binary Relations,

Equivalence Relations and Partitions, Partial Ordering Relations, Hasse Diagram, Lattices, Chains and Anti-Chains.
Functions: Composition of Functions, Injective function, Surjective function, Bijective function, Invertible Functions, Hash function: Division method, Midsquare method and Folding method, Pigeonhole Principle.

Unit V: Graph Theory (07)
Basic Terminology, Multi-Graphs and Weighted Graphs, Sub-Graphs, Isomorphic Graphs, Complete, Regular and Bipartite Graphs, Operations on Graph, Factors of a Graph, Paths and Circuits, Connectivity, Hamiltonian and Euler Paths and Circuits, Planar Graph and Theorem, Shortest Path in Weighted Graphs (Dijkstra's Algorithm), Applications of Graph-Graph Coloring Problem, Travelling Salesman Problem.

Unit VI: Trees (06)
Basic Terminologies in Trees and Properties of Trees, Binary Search Trees, Tree Traversal, Spanning Trees, Fundamental Circuits and Cut Sets, Minimal Spanning Trees, Kruskal's and Prim's Algorithms for Minimal Spanning Trees, Transport Network.

Text Books:
1. C. L. Liu and D. P. Mohapatra, **"Elements of Discrete Mathematics"**, 4th Edition, *Tata McGraw-Hill*, 2017, ISBN 978-1-25-900639-5.
2. Kenneth H. Rosen, **"Discrete Mathematics and its Applications",** 7th Edition, 2012, *Tata McGraw-Hill*, ISBN 978-0-07-338309-5.

Reference Books:
1. B. Kolman, R. Busby and S. Ross, **"Discrete Mathematical Structures"**, 6th Edition, *Pearson Education*, 2009, ISBN 81-7808-556-9.
2. Seymour Lipsehutz and Marc Lars Lipson **"Discrete Mathematics"**, 3rd Special Indian Edition, ISBN-13: 978-0-07-060174-1.
3. J. P. Tremblay and R. Manohar, "**Discrete Mathematical Structures with Applications to Computer Science"**, 1997, *Tata McGraw-Hill*, ISBN 0-07- 463113-6.
4. E. Goodaire and M. Parmenter, **"Discrete Mathematics with Graph Theory"**, third edition, *Pearson Education*, 2008, ISBN 81 – 7808 – 827 – 4.
5. N. Deo, **"Graph Theory with application to Engineering and Computer Science"**, Eastern Economy Edition, *Prentice Hall of India*, 1990, 0 – 87692 – 145 – 4.

Online/Web/Other References:
1. 12 Week NPTEL course, https://nptel.ac.in/courses/106/106/106106183/#

List of the Tutorial Assignments

Every student should perform 12-14 tutorials which will cover topics of all units mentioned in the Syllabus of Discrete Mathematics.
Following list of tutorials can be considered as a guideline for designing tutorials in such a way that all topics should be distributed and covered amongst all batches.

1. Problems on set, multi-set operations, Venn diagram and algebra of sets.
2. Problems on Principle of Inclusion-Exclusion.
3. Illustrative example solving using Mathematical Induction.
4. Translating English statement into propositional logic and predicate logic.
5. Problems on groups.
6. Problems on permutation and combination.
7. Representation of relations and functions, closure of relations and equivalence relation.
8. Problems on partitions, POSET's, Hasse diagram and Lattices.
9. Problems on Warshall's Algorithm.
10. Problems on composition of functions, invertible functions, recurrence relation.
11. Problems on multi-graphs and weighted graphs, sub-graphs, isomorphic graphs.
12. Solve problems for shortest path in weighted graphs (Dijkstra's algorithm)
13. Solve problems on Kruskal's and Prim's algorithms for minimal spanning trees.
14. Solve problems on Cut sets and Transport network

20CE 304 Digital Systems And Computer Organization

Teaching Scheme	**Examination Scheme**
Lectures: 3 Hours / Week	In Semester: 50 Marks
Tutorial: 1 Hour / Week	End Semester: 50 Marks
	Credits: 4

Course Objectives:
To facilitate the learner to
1. Understand the basic digital circuits and logic design.
2. Apply techniques for designing combinational and sequential circuits.
3. Understand the functional components of a computer and its organization.
4. Understand design issues of instructions and instruction pipelining.
5. Understand and classify memory and input/output organizations.

Course Outcomes:
After completion of the course, students will be able to
1. Apply the knowledge of basic digital circuits and logic design.
2. Apply the knowledge of combinational and sequential digital circuits.
3. Relate the basic building blocks, their coordination and instruction pipelining in computer organization.
4. Utilize the concept of I/O and memory organization to computer system.

Unit I: Combinational Circuits (08)
Minimization of Product of Sum(POS) and Sum of Product(SOP) functions and realization using logic gates, Introduction to Numbers and Codes, BCD, Gray, Excess-3 and their applications, Code conversion, Integer and floating point number representation, Signed and unsigned numbers, arithmetic operations.

Unit II: Combinational Logic Design (06)
Realization of basic combinational functions like comparison, decoding, multiplexing, demultiplexing, Design of Half Adder and Full Adder, Design of Half Subtractor and Full Subtractor, BCD Adder, Look ahead and carry generator, Introduction to Carry Propagation Adder.

Unit III: Sequential Circuit's Design (07)
Flip flops (FF) and their excitation tables, FF conversions, Shift registers, Applications of FFs, Asynchronous and Synchronous counters, Sequence detectors using Moore and Mealy, Introduction to Algorithmic State Machines (ASM) charts, notations, design of a simple controller using ASM.

Unit IV: Introduction to Computer Organization (07)
Introduction to Computer Organization, Function and structure of a computer, Functional components and their Interconnection, Register organization, Case study of 8086, Number and size of registers, General purpose registers, Design and Organizational issues of registers, Control Unit organization, Hardwired vs. microprogrammed organization.

Unit V: Characteristics, Functions and Pipelining of Instructions (07)
Instruction cycle, type of instructions, types of operands, Instruction set design, machine instructions characteristics, design issues of instructions, addressing modes, Case study of 8086, Instruction pipelining, performance and hazards of pipelining, RISC, CISC.

Unit VI: Memory and Input/output Organization (07)
Memory devices and organization, ROM, RAM, EPROM, SDRAM, DDR4 RAM, Flash memory. Cache memory organization, principles, cache design elements, performance characteristics, External memory devices and organization, Introduction to buses, types of buses, bus organization, DMA organization, need, working principle.

Text Books:

1. R. P. Jain, 'Modern Digital Electronics', Tata McGraw-Hill, (4th Edition), (2009)
2. A. AnandKumar, 'Fundamentals of Digital Circuits', PHI Learning, (4th Edition), (2016)
3. C. Hamacher, Z. Vranesic and S. Zaky, 'Computer Organization and Embedded Systems', McGrawHill, (5th Edition), (2017)
4. W. Stallings, 'Computer Organization and Architecture - Designing for Performance', Prentice Hall of India, (10th Edition), (2016)

Reference Books:

1. Anil Maini, 'Digital Electronics: Principles and Integrated Circuits', Wiley India Ltd, (2019)
2. Malvino, D. Leach, 'Digital Principles and Applications', Tata Mc-Graw Hill, (8th edition), (2014)
3. John P Hays, 'Computer Architecture and Organization', McGraw-Hill Publication, (3rd Edition), (2017)
4. A. Tanenbaum, 'Structured Computer Organization', Pearson, (6th Edition), (2016)

Online/Web/Other References:

1. NPTEL series – nptel.ac.in/courses/117105080/ (Digital System Design by Prof. D. Roychoudhary, Dept. of Computer Science and Engineering, IIT Kh.)
2. Online Chapters – WilliamStallings.com/COA/COA8e.html

Suggestive List of Tutorials

Sr. No	Topics
1	SOP and POS examples to implement and verify Boolean laws.
2	K- Map Examples based on 2, 3 and 4 variables.
3	The output of a shaft encoder need to be given to a computer system. Design a suitable code converter circuit.
4	The output of a decimal counter need to be apply to a computer system which understands Bin/Oct/Hex Number system. Convert the decimal number in respective other number system.
5	A. Design a digital circuit to work as a TV remote having 8/16 inputs. B. Design a digital circuit to broadcast a message to 4/8 screens after a fixed time interval.
6	Design a digital circuit to convert JK flip-flop to the available D/T/SR Flip Flop.
7	Design a sequence detector to find the given sequence in a large bit stream using Moore/Mealy method.
8	Identify the Addressing modes for the given code/set of instructions.
9	Find speed up and throughput for a given system for a non-pipelined method and compare it with pipelined system having 5 stages.
10	Cache Memory Examples – Direct mapping, Fully Associative mapping, Set Associative mapping

20CE 302L Data Structures Laboratory

Teaching Scheme	**Examination Scheme**
Practical : 4 Hrs/Week	In Semester : 25 Marks
	Practical : 25 Marks
	Credits: 2

Prerequisite:

1. 20ES05 Fundamentals of Programming Language
2. 20ES05L Fundamentals of Programming Language Laboratory

Course Objectives:

To facilitate the learner to

1. Develop algorithmic foundations to solve problems.
2. Select and use appropriate data structure for a given problem statement.
3. Analyze algorithms using time complexity.
4. Implement small application using data structures.

Course Outcomes:

After completion of the course, students will be able to

1. Select appropriate data structure for given problem.
2. Develop the solution for the given problem using programming language.
3. Analyze solutions using time complexity.
4. Design a small application using data structure.

Preamble:

The laboratory assignments are designed in a set of group A, B and C such that students will be able to design and implement solution for a given problem using various data structures. Motivation here is that students should be able to code the basic algorithm and select appropriate data structure to implement the solution of given problem. Faculty members are encouraged to expand problem statements with variations. Assignments can be framed and expanded in such a way
that it explores concepts, logic of solution and simple application. Students will be encouraged to solve open problems in different domains. Faculty will appropriately adopt assignments on similar lines as the examples shown here.

Group B assignments are designed in such a way that students will choose appropriate data structures to implement solution of a given problem. Some assignments of group A are designed to make students able to implement Abstract Data Type of a data structure and use it for a given application. Faculty members should choose the assignments from group A such a way that all the units of the syllabus of Data Structures are covered. In group C assignments students will design an algorithmic solution for selected problem using concepts covered in the subject Data Structures.

The laboratory assignments of group A and B are to be submitted by student individually using C++/JAVA object oriented programming language. Group C assignments may be performed in a group of 2 to 4 students from the same batch. For each assignment program code with sample output is to be submitted as a soft copy.

Suggestive List of Assignments

Group A : (Any Six)

1. In a group of M persons, some people can speak English and some people can speak French. Implement program to find and display-
   1. People who speak either English or French or both.
   2. People who speak both English and French.
   3. People who speak only English not French.
   4. Remove the person from the group.
2. Consider students marks of specific subject are to be stored using Array as ADT. Implement operations to summarize/ analyze the marks of the subjects.
3. Consider a mobile phone stores name and contact number in ascending order. Write program to search a contact details of specified name.
4. Consider students roll numbers and percentages of SY class are stored. Implement operations to arrange students records in ascending/ descending order based on their marks using various sorting methods.
5. Implement Doubly Linked List as ADT .Use same ADT to simulate Browser URL application.
6. Implement Singly Linked List as ADT. Use same ADT to simulate deck of cards application.
7. A 'concordance List' is an alphabetical list of words that appear in the book. Implement concordance list using ordered Linked List with insertion function that restrict duplicate value to be inserted in the list.
8. Implement Singly Linked List as ADT. Use it to simulate banking operations.
9. Student's information along with their percentage is stored in linked list for every division. Generate a combine list of students which is sorted in descending order based on their percentage.
10. Implement Stack as ADT using linked list or array. Use same ADT to check given expression is well formed parenthesized.
11. Implement Stack as ADT using linked list or array. Use same ADT to evaluate given postfix expression.
12. Implement Priority Queue as ADT using linked list or array. Use ADT to simulate pizza parlor order management.
13. Operating system stores N jobs and processing time require to complete each job in data structure. Design a program to simulate the job execution sequence

14. Implement Queue as ADT . Use Queue ADT to simulate 'waiting list' operations of railway reservation system.
15. Company wants to lease phone lines to connect its offices of different cities, with each other. Company charges different amounts of money to connect different pairs of offices. Solve the problem using graph data structures to connect all offices of a company with a minimum cost.
16. Implement graph as ADT to represent current flow in electrical circuit board.
17. An airport is developing a computer simulation of air traffic control that handles events such as landings and takeoffs. Each event has a time stamp that denotes the time when the event will occur . Develop a code for inserting an event and exacting most recent event and display all events. Use heap as ADT to implement priority queue .
18. Consider players score obtained in game are stored. Find out maximum and minimum score obtained in that game using heap data structure.
19 . Implement binary tree as ADT and use it for simulating operations on employee data.
20. Implement open hashing technique and use it to quickly look up employee's information. Provide facility to insert, display, search record .
21. Consider telephone book database of N clients. Make use of a hash table implementation to quickly look up client's telephone number.
22. Implement dictionary as ADT using hashing technique.

Group B: (Mandatory)

1. Department of Computer Engineering has 'CSI student branch'. Students of second, third and final year can subscribe to membership. Design a system to maintain CSI student branch membership information to add, delete, and modify details of records with ease. Use appropriate data structure.
2 College Library maintains records of books. Book records contain basic information of book. Book records are to be listed in the specific order. List of books of specific author are to be searched. Use appropriate data structure to perform sorting, searching operations of book data effectively.
3. A dictionary t stores keywords and its meanings as a key value pair. Use appropriate data structure that will provide minimum comparisons to find any keyword. Provide facility to adding new keywords, deleting keywords and modifying meaning of keywords.
4. A news paper delivery boy every day drops news paper in a society having many lanes and houses. Design a program to provide different paths that he could follow. Solve the problem by suggesting appropriate data structures. Design necessary classes.

Group C:

1. Design a game like snake and ladder, tic-tac-toe, generating magic square.
2. Design a small application using appropriate data structures to manage library data medical shop data/ College admission data / P.M.P.M.L. bus scheduling data etc.

20CE 305L Programming Skills Development - I Laboratory

Teaching Scheme	**Examination Scheme**
Practical: 4 Hrs/week	In semester: 25 Marks
	Oral: 25 Marks
	Credits: 2

Prerequisites:
1. Fundamentals of Programming Language Lab-I (20ES02L)
2. Fundamentals of Programming Language Lab-II (20ES05L)

Laboratory Objectives:
To facilitate the learners to -
1. Use basics of Python, including working with functions, numbers, lists, and strings.
2. Work with file handling concepts.
3. Use numpy, matplotlib libraries for python application.
4. Apply object-oriented features to Python code.
5. Create a simple GUI using Tkinter

Laboratory Outcomes:
By taking this course, the learner will be able to -
1. Make use of basics of Python, including working with functions, numbers, lists, and strings.
2. Implement a python program to work with files.
3. Write basic, object-oriented Python code.
4. Create Python programs using numpy, matplotlib libraries.
5. Build a simple GUI using Tkinter

A large part of the lab would be for understanding the basic concepts of Python programming and implementation of some real world simple applications. Assignment statements are in brief and should be implemented in Python programming language. Motivation here is that students should be able to code the basic algorithm and also should be able to make use of built in functions available in different libraries of Python. Faculty members are encouraged to expand problem statements with variations. Assignments can be framed and expanded in such a way that it explores concepts, logic of solution and simple application. Students will be encouraged to solve open problems in different domains. Faculty will appropriately adopt assignments on similar lines as the examples shown here. Group A assignments are based on basics of Python and file handing. Group B assignments are based on object oriented programming, use of Matplotlib and numpy libraries and GUI using Tkinter. Group C assignment is implementation on mini project.

Suggestive List of Assignments:

Group A: (Mandatory)

1. Assignments to explore Lists, Dictionary and tuples like Create a menu drive Python program with a dictionary for words and their meanings. Write functions to add a new entry (word: meaning), search for a particular word and retrieve meaning, given meaning find words with the same meaning, remove an entry, display all words sorted alphabetically.
2. Assignments to explore String. For Example: Write a function word_lengths that takes a sentence (string), computes the length of each word in that sentence, and returns the length of each word in a list. You can assume that words are always separated by a space character " ".
3. Assignment to display a particular pattern or sequence. For example: Generate a Pascal triangle for n rows
4. Assignment to perform file operations. For example: Write a program that opens a file dialog that allows you to select a text file. The program then displays the contents of the file in a textbox.

Group B: (Any four)

1. Assignment based on object oriented principles. For example: Design a student data base in Python using classes and objects to perform the following operations:
    a. add b) delete c) display d) update e) search
2. Implement a Python program to perform operations on arrays and matrices. For example, Matrix multiplication
3. Read data from CSV file and plot it using matplotlib library
4. A picture or image can be represented as a NumPy array of "pixels", with dimensions H × W × C, where H is the height of the image, W is the width of the image, and C is the number of colour channels. Typically, we will use an image with channels that give the Red, Green, and Blue "level" of each pixel, which is referred to with the short form RGB. You will write Python code to load an image, and perform several array manipulations to the image and visualize their effects.
5. Use Tkinter to build a simple graphical user interface. For example: GUI to maintain a simple phone list.

Group C: Mini Project (Any one: For example:)

1. Devise a Python program to implement the Rock-Paper-Scissor game.
2. Devise a Python program to implement the Hangman Game.
3. Creating a Calculator with Tkinter

References:

Text Books

1. Kenneth. A. Lambert, "**Fundamentals of Python First Programs**", Cengage, 2nd Edition, 2019

2. Vamsi Kurama, "**Python Programming: A Modern Approach** ", Pearson, 1st Edition, 2017.

Reference Books:

1. Gowrishankar.S, Veena A, "**Introduction to Python Programming**", CRC Press, Paperback Edition, 2019.

2. Y. Daniel Liang, "**Introduction to Programming Using Python**", Pearson, Paperback Edition, 2017.

e-Resources:

1. https://www.tutorialspoint.com/python3/python_tutorial.pdf

20CE 304L Digital Systems Laboratory

Teaching Scheme	**Examination Scheme**
Practical : 2 Hours / Week	In Semester : 25 Marks
	Oral: 25 Marks
	Credits: 1

Prerequisite: Basic Electrical and Electronics Engineering (20ES01)

Course Objectives:

To facilitate the learner to

1. Understand the basic digital circuits and logic design.
2. Apply techniques for designing combinational and sequential circuits.
3. Apply the knowledge to select different digital IC packages as per design specifications.
4. Develop minimum digital systems for simple real time applications.

Course Outcomes:

After completion of the course, students will be able to

1. Apply the knowledge of basic gates to build digital circuits.
2. Make use of available circuit packages to develop combinational circuits.
3. Apply the knowledge of sequential circuits design to model digital systems.
4. Build a small digital system using an emulator tool.

The laboratory work of Digital Electronics Lab is designed to enhance problem solving in digital electronics with the help of Boolean algebra, logic gates, computer number systems, data encoding, combinational and sequential elements. The circuit optimization is introduced using K-Maps. The solution building to real world problems is aimed with the help of circuit packages. Faculty members are encouraged to expand problem assignments with variations for Group B and Group C assignments. Assignments can be framed and expanded to understand the basic concepts, design steps, logic of solution and simple digital application. The students will be also encouraged to experiment open problems with the designs using appropriate emulator tools. Faculty will ratify the assignments on similar lines as examples shown here. Majority of Group A assignments are based on combinational circuits such as code converter, multiplexer, decoder etc. and partially sequential circuits such as Asynchronous and Synchronous counters. Group B assignments are based on sequential circuits such as sequence generator, sequence detector, ASM using flip-flops as well as based on real world application level assignments. Group C assignments are based on implementation of different real time applications based on combinational and sequential circuits.

Suggestive List of Assignments

Group A : (Perform minimum 6)

1. Design and implement different logic circuits by using Basic gates and Universal gates.
2. Design and implement code converter circuits e.g. Binary to Gray, BCD to Ex-3 etc.
3. Design and implement circuits using Multiplexer and Decoder.
4. Design and implement a circuit to detect the error in the digital data communication system.
5. Design and implement Binary subtractor using 1's and 2's complement method. Use binary adder IC 7483.
6. Design and implement Asynchronous counters using a given flip flop.
7. Design and implement Synchronous counters using a given flip flop.

Group B: (Perform any one assignment each from 1 to 4 and 5 to 8)

1. Design and implement Sequence generator circuit. Check for the lockout condition.
2. Design and implement Sequence detector circuit using Moore and Mealy.
3. Design and implement flip flop conversion circuit.
4. Design and implement a simple ASM chart using a digital circuit.
5. Design and implement a car parking system using Entry and Exit gate, synchronized with each other. Entry gate counter will increment the count for each car entry. When all parking slots are occupied, it should indicate that parking is Full.
6. Design and implement a car parking system using Entry and Exit gate, synchronized with each other. Exit gate counter will decrease the count for each car leaving the gate. When no vehicle is present in the parking slot system, it should indicate Parking is Empty.
7. Design and implement an Ice Cream cup distribution counter based on dozens system. Counter should decrement for each sale of the cup. When a dozen of cups in the box are sold, the counter should indicate the one box is done.
8. Design and implement a packaging counter for 16 items. Counter should increment for each entry of the item in the box. When the box is full, the counter should indicate the box is full and reset itself.

Group C (Perform any one)

Select any open source / freeware tool and design a digital system of your choice.

1. Design a trigger circuit which will activate next circuit after 45 clocks.
2. Design a square wave generator circuit.
3. Build a random sequence generator.
4. Design a decimal adder circuit.
5. Design an octal adder circuit.
6. Design a decimal subtractor circuit.
7. Design a traffic signal controller which can show red signal for 70 sec, yellow signal for 5 sec and green signal for 40 sec.
8. Design a 2 digit traffic signal controller.
9. Design a 2 digit Bank token system.
10. Design a 3 digit Vaccine token system.
11. Design a BCD to 7-Segment display.
12. Design a character to 7-Segment display.