GUJARAT TECHNOLOGICAL UNIVERSITY

Syllabus for Master of Computer Applications, 2nd Semester
Subject Name: Data Structures
Subject Code: 629401

With effective from academic year 2020-21

1. Learning Objectives:

- To develop proficiency in the specification, representation, and implementation of Data Types and Data Structures.
- To introduce the concepts of algorithmic paradigms and basic data structures and their applications.
- To analyze various algorithms for space and time complexity.
- To implement and compare various searching and sorting techniques.
- To apply appropriate data structures to solve different problems.
- To develop a base for advanced computer science study.

2. Prerequisites:

Proficiency in a programming language

3. Course Contents:

Unit No.	Course Content	Weightage Percentage
I	**Introduction to Data Structure and Algorithm Analysis:** Data Structure Definition and classification, Algorithm Analysis, Storage Representation of Strings, Text Handling and KWIC Indexing.	**10%**
II	**Linear Data Structures:** Arrays, Storage Structure for Arrays, Stack: List Implementation, Applications of Stacks: Function Call, Recursion, Balancing Symbols Queue: List Implementation, Circular Queue, Priority Queue, double ended queue. Linked List: Cursor Implementation, Multi List Applications of Linked List : Addition and Multiplication of Polynomial in one and two variables	**25%**
III	**Nonlinear Data Structures:** Tree - Basic Tree Concepts, Operations on Binary Trees, Storage Representation & Manipulation of Binary Trees, Conversion of General Tree to Binary Trees, Sequential & Other Representation of Trees, Application of Trees – The Manipulation of Arithmetic Expression, Multi-linked Structures - Sparse Matrices.	**20%**
IV	**Graphs and Their Representation**: Matrix Representation of Graphs, List Structures, Other Representation of Graphs, Breadth First Search and Depth First Search.	**20%**
V	**Sorting and Searching Techniques:** Sorting – Notation and Concepts, Selection Sort, Bubble Sort, Merge Sort, Heap Sort, Quick Sort, Searching - Sequential Searching, Binary Searching, Search Trees – Height Balanced, 2-3 Trees, Weight Balanced Tree, Tree Structures, Hash Table Search Methods, Introduction, Hashing Functions.	**25%**

4. Text Books:

1. Jean-Paul Tremblay, Paul G. Sorenson, "An Introduction to Data Structures with Applications", Tata McGraw-Hill, 2nd Edition, (2007)
2. Mark Allen Weiss, “Data Structures and Algorithm Analysis in C”, Pearson, Second Edition

1. Reference Books:

- Ashok N. Kamthane, “Introduction to Data Structures in C”, Pearson Education (2004).
- Cormen, Leiserson, Rivest, Stein,"Introduction to Algorithm", PHI. 2nd Edition(2003).
- Parag H Dave, Himanshu B Dave, “Design and Analysis of Algorithms”, Pearson (2014)
- Samir Kumar Bandyopadhyay, Kashi Nath Dey, “Data Structures Using C”, Pearson Education, Year: 2004.
- Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, “Data Structures and Algorithms”, Pearson Education (2002).
- Horowitz, Sahni, Anderson-Freed, “Fundamentals of Data Structures in C”, University Press (2nd edition-2007)
- G. A.V.PAI, "Data Structures and Algorithms, Concepts, Techniques and Applications”, TMH, 1st Edition (2008).

2. Chapter Wise Coverage from Text Book:

Unit No.	Text Books	Topics/Subtopics
I	Book-1	0-3.0 to 0-3.5, 2.4, 2.5.3
II	Book-1 Book-2	3.2, 3.5,3.6 to 3.8,4.3.1 3.3.3,3.2.7,3.2.8
III	Book-1	5.1.1 to 5.1.5, 5.2.1, 5.3.1
IV	Book-1	5.4.1 to 5.4.5
V	Book-1	6.1.1, 6.1.2, 6.1.3, 6.1.4, 6.1.5, 6.1.6, 6.2.1, 6.2.2, 6.2.3,6.2.3.1, 6.2.3.2, 6.2.3.3, 6.2.3.4, 6.2.4, 6.2.4.1, 6.2.4.2

3. Accomplishments

- Apply sorting and searching algorithms to the small and large data sets.
- Ability to design and implement abstract data types such as linked list, stack, queue, graphs and trees using static or dynamic implementations.
- Analyze the complexity of different algorithms.

Practical List

Use C programming language to perform followings Lab work:

1.	Create a Structure with following Data Members: 1. Integer Array 2. Size of the Array Sort the Array using various Sorting algorithms such as (i) Selection Sort (ii) Bubble Sort (iii) Two-way Merge Sort (iv) Quick Sort (v) Heap Sort. And store the sorted Array in a text file.
2.	Create a Structure with following Data Members: 1. Integer Array 2. Size of the Array Search an element in Array using Linear (Sequential) Search and Binary Search, and Display result in file. For Sequential Search, assume that input array is Unordered and for Binary Search assume that input array is Ordered and develop programs accordingly.
3.	Create a "Stack" data structure with following Data members: 1. Integer Array 2. Stack Pointer (Top of Stack: Is it same as the Size of the Array) Perform the following operations on stack using user-defined functions: 1. Push 2. Pop 3. Isempty 4. Isfull 5. Peep Create a file which stores all values of Array through Stack. Has it reversed the order of the elements of the Array? Why?
4.	Create a "Linked List" structure with the following data members: 1. A Data 2. A link to the next node Perform the following operations on stack using user-defined functions: 1. Insert a value X at the first place 2. Insert a value X at the end of the list 3. Insert a value X at the place so that it preserves the ordering of the terms in the increasing order. i. Delete an element whose address is given by X i. Copy a linked linear list Create a file which stores all values of list.
5.	Write a program to convert an infix arithmetic expression (parenthesize / unparenthesized) into postfix notation.
6.	Write a program to evaluate a postfix expression.
7.	Create a structure with the following Data members: 1. Integer Array 2. Size of the Array Search an element in a given list using Binary Search by recursion. And Display result in a file.

8.	Create a "Queue" structure with following Data members: 1. Integer Array 2. Size of the Array Perform the following operations on Simple queue using user-defined functions: 1. Insert an element 2. Remove an element 3. Display 4. Isfull 5. Isempty Create a file which stores all values of Array.
9.	Create a "Queue" user-defined structure with the following data members: 1. A Data 2. A link to the next node Perform the following operations on Simple queue using user-defined functions: 1. Insert an element 2. Remove an element 3. Display 4. Isfull 5. Isempty Create a file which stores all values of list.
10.	Create a "Circular Queue" structure with following Data members: 1. Integer Array 2. Size of the Array Perform the following operations on Circular queue using user-defined functions: 1. Insert an element 2. Remove an element 3. Display 4. Isfull 5. Isempty Create a file which stores all values of Array.
11.	Create a "Circular Queue" user-defined structure with the following data members: 1. A Data 2. A link to the next node Perform the following operations on Circular queue using user-defined functions: 1. Insert an element 2. Remove an element 3. Display 4. Isfull 5. Isempty Create a file which stores all values of list.
12.	Create a user-defined "Linked List" structure with the following data members: 1. A Co-efficient 2. An Exponent 3. A link to the next node Perform the following operations on Singly list using user-defined functions: 1. Create 2. Display 3. Addition 4. Multiplication Create a file which stores all values of list.

13.	Create a user-defined structure with the following data members: 1. A Data 2. A link to the next node Perform the following operations on list using user-defined functions: 1. Create a list 2. Traverse the whole list 3. Delete first node 4. Delete last node 5. Delete a node before specified data 6. Insert at first position 7. Insert at last position 8. Insert a node before specified data 9. Insert a node at specified position 10. Count 11. Copy 12. Merge two list 13. Reverse 14. Search 15. Sort Create a file which stores all values of list.
14.	Create a user-defined structure with the following data members: 1. A Data 2. A link to the next node Perform the following operations on Circular list using user-defined functions: 1. Create a list 2. Traverse the whole list\ 3. Delete first node 4. Delete last node 5. Delete a node before specified data 6. Insert at first position 7. Insert at last position 8. Insert a node before specified data 9. Insert a node at specified position 10. Count 11. Copy 12. Merge two list 13. Reverse 14. Search 15. Sort Create a file which stores all values of list.

15.	Create a user-defined structure with the following data members: 1. A Data 2. A link to the next node 3. A link to the previous node Perform the following operations on the doubly-linked list using user-defined functions: 1. Create a list 2. Traverse the whole list\ 3. Delete first node 4. Delete last node 5. Delete a node before specified data 6. Insert at first position 7. Insert at last position 8. Insert a node before specified data 9. Insert a node at specified position 10. Count 11. Copy 12. Merge two list 13. Reverse 14. Search 15. Sort Create a file which stores all values of list.
16.	Create a user-defined structure with the following data members: 1. A Data 2. A link to the next node 3. A link to the previous node Perform the following operations on doubly-linked Circular list using user defined functions: 1. Create a list 2. Traverse the whole list\ 3. Delete first node 4. Delete last node 5. Delete a node before specified data 6. Insert at first position 7. Insert at last position 8. Insert a node before specified data 9. Insert a node at specified position 10. Count 11. Copy 12. Merge two list 13. Reverse 14. Search 15. Sort Create a file which stores all values of list.

17.	Write a program to represent an undirected graph using the adjacency matrix to implement the graph and perform following operations, with menu driven options for following tasks: 1. Create graph 2. Insert an edge 3. Print Adjacency Matrix 4. List all vertices that are adjacent to a specified vertex. 5. Print out vertices using depth first search 6. Print out vertices using breadth first search 7. Exit program
18.	Create a user-defined structure with the following data members: 1. A Data 2. A link to the Left child 3. A link to the Right child Perform the following operations on Binary Search Tree using recursion: 1. Create 2. Traverse (Inorder, Preorder, Postorder) 3. Insert 4. Delete 5. Search 6. Create a file which stores all values of traversal.