<table>
<tr><td colspan="6">Devi Ahilya Vishwavidhyalaya, Indore, India<br>Institute of Engineering & Technology</td><td colspan="2">II Year B.Tech.<br>(Computer Science & Engineering)</td></tr>
<tr><td>Course Code & Name</td><td colspan="7">Instructions Hours per Semester and Credits</td></tr>
<tr><td rowspan="3">3RCPC1<br><br>DATA STRUCTURES</td><td colspan="2">Classroom Instruction (CI)</td><td>Lab Instruction (LI)</td><td>Term Work (TW) and Self Learning (SL)</td><td>Total no. of Hours Per semester</td><td>Total Credits (Total Hours/ 30)</td></tr>
<tr><td>L</td><td>T</td><td>P</td><td>TW+SL</td><td rowspan="2">120</td><td rowspan="2">4</td></tr>
<tr><td>30</td><td>10</td><td>20</td><td>60</td></tr>
</table>

**Course Learning Objectives:**

- To provide knowledge of basic data structures and their implementations.
- To understand the importance of data structures in the context of writing efficient programs.
- To develop skills to apply appropriate data structures in problem-solving.

**Prerequisites:** Basic Programming Course on C++

## Course Contents

**Unit-I**

**Introduction to Data Structures, Performance Analysis, and Arrays:** Data structures classification: primitive and non-primitive, Abstract Data Types (ADT), Time and space complexity, Asymptotic notations, Recursion fundamentals, Array operations: insertion, deletion, traversal, Dynamic memory allocation, Multidimensional arrays.

**Unit-II**

**Linked Lists, Stacks & Queues:** Singly, doubly, and circular linked lists; Node operations: create, insert, delete, search; Applications: Polynomial Addition, Sparse Matrices, Stack: implementation and Applications: infix to postfix, recursion handling, Queue: linear and circular, priority queue, deque; Applications.

**Unit-III**

**Trees:** Binary tree: Definition, Basic terminology, Binary tree, Complete Binary Tree, representations: Static and dynamic, Traversal techniques in binary tree, BST and AVL trees; M-way search trees, Binary Heap, Applications: expression trees.

**Unit-IV**

**Graphs:** Definition, Basic terminology, Graph Types, Representations: Adjacency Matrix, Adjacency List; Implementations, Graph traversal: BFS and DFS, Shortest path and MST: Dijkstra, Kruskal, Prim.

**Unit-V**

**Hash Tables, Searching and Sorting Algorithms:** Hash functions and collision resolution; Chaining, open addressing, Linear & binary search, Sorting: bubble, selection, insertion, merge, quick, heap, Overview of Standard Template Library (STL) & Templates.

**Course Outcomes(CO):**

| CO. No. | CO |
|---|---|
| CO1 | Understand and analyse the complexity of algorithms and recursive techniques, as well as Introduction to Data Structures. |
| CO2 | Implement linear data structures (arrays, linked lists, stacks, and queues) in C++. |
| CO3 | Apply tree data structure to solve problems using C++. |
| CO4 | Understand the Graph data structures, their implementations and real-world applications |
| CO5 | Implement searching, sorting, and hashing algorithms and analyse their efficiency. Apply STL containers and use them for real-world applications. |

**Books Recommended**

[1] Mark Allen Weiss, Data Structures and Algorithm Analysis In C++, Pearson, 4th Edition, 2014.
[2] E. Horowitz & Sahni, Fundamentals of Data Structures In C++, Universities Press, 2nd Edition, 2008.
[3] Adam Drozdek, Data Structures and Algorithms in C++, 2nd Edition, Cengage Publication, 2017.
[4] Goodrich & Tamassia, Data Structures and Algorithms In C++, 2nd Edition, John Wiley & Sons, 2011.
[5] Narasimha Karumanchi, Data Structures and Algorithms Made Easy, Careermonk Publications, 5th Edition, 2016.

**CO-PO-PSO Relationship**

| CO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CO1** | 3 | 3 | - | 2 | 1 | - | - | - | - | - | 2 | 1 | 3 | 1 |
| **CO2** | 3 | 2 | 2 | - | 3 | - | - | - | - | - | 1 | 2 | 3 | 1 |
| **CO3** | 3 | 3 | 3 | 2 | 2 | - | - | - | - | - | 1 | 2 | 3 | 1 |
| **CO4** | 2 | 3 | 2 | 2 | 3 | 1 | - | - | - | - | 1 | 2 | 3 | 2 |
| **CO5** | 3 | 3 | 2 | 2 | 3 | - | - | - | - | - | 2 | 3 | 3 | 2 |

**List of Laboratory Experiments**

1. Introduction to C++ and STL
   - Basics of C++ programming
   - Use of STL containers (vector, list, stack, queue, map)

2. Time and Space Complexity Analysis
    - Analysis of iterative and recursive algorithms
    - Program to demonstrate best, average, and worst-case complexities
3. Implementation of Arrays
    - Insertion, deletion, traversal, and searching operations
4. Implementation of Singly and Doubly Linked Lists
    - Creation, insertion, deletion, and traversal operations
5. Implementation of Stack using Array and Linked List
    - Applications: Expression evaluation, infix-to-postfix conversion
6. Implementation of Queue and Circular Queue
    - Applications: CPU scheduling, producer–consumer problem
7. Implementation of Recursion Techniques
    - Programs for factorial, Fibonacci, and Tower of Hanoi
8. Implementation of Searching Algorithms
    - Linear search
    - Binary search and performance comparison
9. Implementation of Sorting Algorithms
    - Bubble sort, Selection sort, Insertion sort
    - Comparison of time complexity
10. Implementation of Hashing Techniques
    - Hash tables using chaining and linear probing
11. Implementation of Trees
    - Binary Tree creation and traversal
    - Binary Search Tree (BST) operations
12. Implementation of Tree Applications
    - Height of tree
    - Tree traversal using recursion and non-recursion
13. Implementation of Graphs using Adjacency Matrix and List
    - Graph traversal using BFS and DFS
14. Minimum Spanning Tree and Shortest Path Algorithms
    - Prim’s / Kruskal’s algorithm
    - Dijkstra’s algorithm