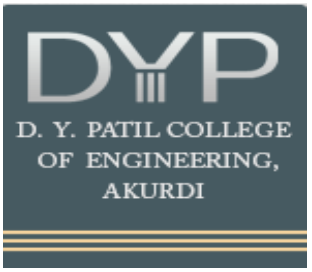
D. Y. Patil College of Engineering
Akurdi, Pune -44



Fundamentals of Data Structure Lab

LAB MANUAL

Year: II

Semester: III

Hardware and Software Requirement

Hardware Requirement

- Processor : Dual Core
- RAM : 1GB
- Hard Disk Drive : > 80 GB

Software Requirement

Server Configuration

- IBM Server
- RAM – 32 GB
- Hard Disk – 3 TB
- Operating System – 64-bit Open source Linux or its derivative

- **Programming tools recommended: -** Open Source Python like JupyterNotebook, Pycharm, Spyder, G++ /GCC

Institute Vision

"Empowerment through knowledge"

Institute Mission

To educate students to transform them as professionally competent and quality conscious engineers by providing conducive environment for teaching, learning and overall personality development, culminating the institute into an international seat of excellence.

Vision of Department

Developing highly skilled and competent IT professional for sustainable growth in the field of Artificial Intelligence and Data science

Mission of Department

1. To empower students for developing intelligent systems and innovative products for societal problems.

2. To build strong foundation in Data computation, Intelligent Systems that enables self-development entrepreneurship and Intellectual property.
3. To develop competent and skilled IT professional by imparting global skills and technologies for serving society.

Program Specific Outcomes:

1. **Professional Skills-**The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, networking, artificial intelligence and data science for efficient design of computer-based systems of varying complexities.

2. **Problem-Solving Skills-** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

3. **Successful Career and Entrepreneurship-** The ability to employ modern computer languages, environments and platforms in creating innovative career paths to be an entrepreneur and to have a zest for higher studies**.**

Program Educational Objectives:

1) ***Core Competence****:* To provide students with a solid foundation in mathematical, scientific and engineering fundamentals required to solve engineering problems and also to pursue higher studies.

2) ***Breadth****:* To train students with good scientific and engineering breadth so as to comprehend, analyse, design, and create novel products and solutions for the real-life problems.

3) ***Professionalism****:* To inculcate in students professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, and an ability to relate engineering issues to broader social context.

4) ***Learning Environment****:* To provide student with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the life-long learning needed for a successful professional career.

5) ***Attainment***: To prepare students to excel in research or to succeed in industry/technical profession through global, rigorous education and research and to become future entrepreneurs.

Sr. No.	Course Objectives
1	To understand the standard and abstract data representation methods.
2	To acquaint with the structural constraints and advantages in usage of the data.
3	To understand various data structures, operations on it and the memory requirements
4	To understand various data searching and sorting methods.
5	To understand various algorithmic strategies to approach the problem solution.

Course Outcomes	
After successful completion of this course the student will be able to:	
CO1	**Design** the algorithms to solve the programming problems, **identify** appropriate algorithmic strategy for specific application, and **analyze** the time and space complexity.
CO2	**Discriminate** the usage of various structures, **Design/Program/Implement** the appropriate data structures; use them in implementations of abstract data types and Identity the appropriate data structure in approaching the problem solution.
CO3	**Demonstrate** use of sequential data structures- Array and Linked lists to store and process data.
CO4	**Understand** the computational efficiency of the principal algorithms for searching and sorting and choose the most efficient one for the application.
CO5	**Compare** and **contrast** different implementations of data structures (dynamic and static).
CO6	**Understand, Implement and apply** principles of data structures-stack and queue to solve computational problems.

GENERAL INSTRUCTIONS FOR STUDENTS

DO'S

- Students should enter into the Laboratory with prior permission.
- Students should come in proper uniform.
- Students should come with Practical note book to the laboratory.
- Students should maintain silence inside the laboratory.
- After completing the laboratory exercise, make sure to shut down the system and arrange chairs properly.

DONT'S

- Students bringing the bags inside the laboratory.
- Students using mobile phones inside the laboratory.
- Students using the computers in an improper way.
- Students scribbling on the desk and mishandling the chairs.
- Students making noise inside the laboratory.

Course Category	**Program** **Core Course2**	**Course Code**	**AD124PC302**
Course Title		**Fundamentals of Data**	

				Structure Lab			
Teaching Scheme				**Evaluation Scheme**			
L (Hr)	**T (Hr)**	**P (Hr)**	**Cr**	**Exam**	**Lab % Marks**		
					Max %	**Min marks for Passing**	
-	-	2	1	**CCE**	50	20	40
Total Hours:26				**ESE**	50	20	

Prerequisites: Programming and Problem Solving ,OOP		
Course Objective: After successful completion of the course the student will be able to:		
1. **Provide** a strong foundation in fundamental data structures and algorithm design techniques to enable students to effectively analyze, implement, and optimize solutions for computational problems using Python and C++ languages. 2. **Analyze and apply** the fundamental concepts of data structures like arrays, linked lists, stacks, and queues, using Python and C++, to solve computational problems efficiently. 3. **Develop** algorithmic problem-solving skills by implementing various techniques such as searching, sorting, and recursion for practical and real-world scenarios. 4. **Analyze and optimize** algorithms in terms of time and space complexity, ensuring efficient implementation of data structure-based solutions in diverse applications.		
Course Outcomes: After successful completion of the course the student will		
CO1	**Comprehend and Demonstrate** the basic data structure operations, including arrays, strings, linked lists, stacks, and queues.	**BTL2**
CO2	**Apply** different algorithms and techniques to solve computational problems.	**BTL3**
CO3	**Analyze** and **Implement** the time and space complexity of algorithms and compare their efficiency using appropriate metrics.	**BTL4**
CO4	**Compare and Implement** data structures using different algorithms in real-world applications like expression conversion, sparse matrix operations, and circular queue simulations.	**BTL4**

Guidelines

Course Design and Assessment:

- The assignments are divided into groups (A, B, C, and D), with specific implementation requirements.
- Group A and B assignments are to be implemented using Python, focusing on fundamental operations without using built-in methods for core functionalities.
- Group C and D assignments are to be implemented using C++, emphasizing advanced structures and real-world problem applications. a minimum of 9 assignments must be

completed, covering at least 2 assignments from group A, Group B & group C respectively and 3 assignments from group D.

Laboratory Journal Submission:

Students must maintain a laboratory journal with a structured format:

- Title, Objective, Problem Statement, and Outcomes.
- Theory (Concepts and Algorithms), Flowchart, and Test Cases.
- Program Code, Sample Output, Conclusion, and Analysis.
- Journals must be handwritten for problem-solving write-ups but may include soft copies of code and outputs to reduce paper usage.

Evaluation and Assessment:
Continuous evaluation based on:

- Timely submission of assignments.
- Code efficiency and innovation.
- Problem-solving and debugging skills.
- Punctuality and active participation.

Practical examination must include problem-solving demonstrations, viva voce, and code walkthroughs to assess conceptual clarity.

List of Practicals

Syllabus	
	GROUP A
PR1	Implement a python program to store the marks scored by students in a particular subject. Write functions to compute Average, Maximum score, Minimum score and total count of absent students using lists.
PR2	Implement following operations on **Two Dimensional Arrays** a. Addition of Two matrices b. Subtraction of Two matrices c. Multiplication of Matrices d. Transpose of Matrix
PR3	Implement a python program to compute following operation on string: - A. Count the Number of Vowels and Consonants from the input string B. Reverse the Words in the string without changing their order. C. Find the Longest Word in the string
	GROUP B
PR4	Implement a python program to store employee IDs in an array those who attended the training program in random order. Perform following tasks A. Search whether a particular employee attended a training or not using linear search B. Search whether a particular employee attended a training or not using binary search
PR5	Implement a python program to store the percentage of students in an array. write function for sorting array of floating points numbers in ascending order using A. Selection sort and Insertion sort B. Quick sort and bubble Sort
PR6	You are a software engineer tasked with developing a backend system for an online library. The library maintains a vast database of books, each with attributes like **Book ID**, **Title**, **Author**, **Publication Year**, and **Category**. The library system should allow users to: 1. **Search** for a book based on specific attributes (e.g., Book ID, Title) using sentinel search and indexed sequential search **2.** **Sort** books for display based on criteria like alphabetical order of titles, publication year, or author's name using insertion sort and **Shell sort**
	GROUP C

PR 7	The Department of AI & DS has a student's club named 'ISA'. Students of second and third year of department can be granted membership on request. Similarly one may cancel the membership of a club. First node is reserved for the president of the club and the last node is reserved for the secretary of the club. Write C++ program to maintain club member‘s information using Singly linked lists. Store student PRN and Name. Write functions to: A. Add and delete the members as well as president or even secretary. B. Compute total number of members of clubDisplay members C. Two linked lists exist for two divisions. D. Concatenate two lists
PR8	A company's employee data is stored in a doubly linked list sorted by employee ID. When a new employee is added, the data structure needs to maintain its sorted order. Write a C++ program for following tasks A. Insert a node into a sorted doubly linked list while maintaining the sorted order B. Count the number of employees ID stored in doubly Linked list C. Delete any one of them
PR9	Develop a **Ride-Sharing Service Queue Management System** using linked list data structures to handle the following requirements: Maintain a **singly linked list** to store ride requests, where each request contains a unique ride ID, pickup location, drop-off location, rider name, and status (e.g., pending, in-progress, completed). Allow insertion of new ride requests at any position (e.g., based on priority) and deletion upon ride completion or cancellation.
	GROUP D
PR10	Implement a C++ program to check if a mathematical expression is well-parenthesized using a stack. The program should: A. Accept an expression containing parentheses (), curly braces {}, and square brackets []. B. Push opening brackets onto the stack. C. Pop and match with closing brackets to ensure they are balanced and properly nested. Display whether the expression is balanced or not.
PR11	A restaurant accepts maximum M orders. Orders are served on a first come first served basis.Order once placed cannot be cancelled. Write C++ program to simulate the system using a circular queue using arrays

PR12	Implement a C++ program to simulate an online ticket reservation system using a double-ended queue (deque). The program should support the following operations: A. Add a customer at the rear for regular ticket booking. B. Add a VIP customer at the front for priority ticket booking. C. Serve customers from the front of the queue. D. Display the current queue of customers waiting for tickets. Check if the queue is full or empty.
PR13	Design an **Airport Baggage Handling System** using a circular queue to manage the movement of luggage on a conveyor belt. Implement the following operations: 1. Add Luggage: Place new luggage on the conveyor belt. 2. Remove Luggage: Remove luggage once it reaches the pickup point. 3. Overflow Handling: Ensure the system prevents adding luggage when the conveyor belt is full. 4. Underflow Handling: Display a message when the conveyor is empty.

Text Books:

1. Horowitz and Sahani, “Fundamentals of Data Structures in C++”, University Press, ISBN 10:0716782928 /ISBN 13: 9780716782926.
2. Brassard & Bratley, “Fundamentals of Algorithms”, Prentice Hall India/Pearson Education, ISBN 13-9788120311312.

Reference Books:

1. Steven S S. Skiena, “The Algorithm Design Manual”, Springer, 2nd ed. 2008 Edition, ISBN- 13: 978-1849967204, ISBN-10: 1849967202.
2. Allen Downey, Jeffery Elkner, Chris Meyers, “How to think like a Computer Scientist: Learning with Python”, Dreamtech Press, ISBN: 9789351198147.
3. M. Weiss, “Data Structures and Algorithm Analysis in C++”, 2nd edition, Pearson Education, 2002, ISBN-81-7808-670-0.

Journal Papers:

1. B. Park and D. T. Ahmed, "Abstracting Learning Methods for Stack and Queue Data Structures in Video Games," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 1051-1054, doi: 10.1109/CSCI.2017.183.

Vlab:

1. **Quick Sort Experiment,** https://ds1-iiith.vlabs.ac.in/exp/quick-sort/index.html
2. **Stacks and Queues**,https://ds1-iiith.vlabs.ac.in/exp/stacks-queues/index.html

   Linked List,https://ds1-iiith.vlabs.ac.in/exp/linked-list/index.html
3. **Polynomial Arithmetic,** https://ds1-iiith.vlabs.ac.in/exp/poly-arithmetic/index.html

MOOCs:

1. https://nptel.ac.in/courses/106/102/106102064/ (Introduction to Data Structures and Algorithms, IIT Delhi , Prof. Naveen Garg, 40 hrs)
2. https://nptel.ac.in/courses/106/105/106105085 (Programming & Data structure ,IIT Kharagpur , Dr.P.P.Chakraborty ,40 hrs)
3. https://onlinecourses.nptel.ac.in/noc22_cs26/preview (Programming, Data Structures And Algorithms Using Python, Chennai Mathematical Institute,By Prof. Madhavan Mukund ,38 hrs)

Practical No. 01

AIM-Implement a python program to store the marks scored by students in a particular subject. Write functions to compute Average, Maximum score, Minimum score and total count of absent students using lists.

OBJECTIVES:

- To acquire the technique of problem solving through the design of algorithms and programs.
- To design a solution by following the aforementioned constraints
- To create a list having input 'n' resulting in an output of average, highest and lowest marks operations.

THEORY:

Python List

A list in Python is used to store the sequence of various types of data. Python lists are mutable types; it means we can modify its element after it created. However, Python consists of six data-types that are capable to store the sequences, but the most common and reliable type is the list.

A **list** is a built-in data structure in Python that allows us to store multiple values in a single variable.

Characteristics of Lists are:

- **Ordered**: Elements have a defined order and can be accessed using indices.
- **Mutable**: Elements can be changed after the list is created.
- **Heterogeneous**: Lists can contain elements of different data types (e.g., integers, strings, etc.).
- **Dynamic**: The size of a list can grow or shrink during runtime.

In the context of our lab assignment (student marks processing), lists are ideal for storing the marks of multiple students efficiently and performing operations like average, maximum, and minimum calculations.

Creating Lists:

Examples of lists are as follows-

L1 = ["John", 102, "USA"]

L2 = [1, 2, 3, 4, 5, 6]

Lists are created using square brackets [ ] and elements are separated by commas:

marks = [78, 85, -1, 67, 90]

This list contains **integers** (marks of present students) and -1 indicating an absent student.

ALGORITHM:

1. **Input** the number of students.
2. For each student, **input the mark or "-1"** (for absent).

3. Store the marks in a list.
4. Define functions to:
    - Calculate the **average** of present student marks.
    - Find the **maximum** and **minimum** marks.
    - Count the number of **absentees**.
5. Display the results.

CODE:

OUTPUT:

CONCLUSION:

This assignment demonstrates the use of functions to manipulate lists and reusable functions in Python, which is essential for handling real-world data.

DISCUSSION AND VIVA VOCE:

- What is the difference between list and tuples in Python?
- What is the purpose of function?
- What are the key features of Python?
- When would you use a list vs dictionary?
- Does a list need to be homogeneous?
- What is the difference between append and extend?

Practical No. 02

AIM-Implement following operations on **Two Dimensional Arrays**
a. Addition of Two matrices
b. Subtraction of Two matrices
c. Multiplication of Matrices
d. Transpose of Matrices

OBJECTIVES:

The aim is to understand how two-dimensional arrays (matrices) can be manipulated using nested lists in Python.

THEORY:

What is Matrix ?

Matrix is depicted as an array of numbers (real or complex) that are arranged in rows(horizontal lines) and columns (vertical lines). A rectangular representation of mn numbers (complex or real) in the form of m rows and n columns is named as a matrix of order m × n. Any m × n matrix is represented as,

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}_{m \times n}$$

Or, It can also be represented as $A = [a_{ij}]_{m \times n}$

where $1 \le i \le m$ and $1 \le j \le n$.

A two-dimensional (2D) array is a data structure used to store data in rows and columns, like a table or matrix.
A nested list is simply a list that contains other lists as its elements. Each inner list represents a row in the 2D array.
Structure of a 2D Array Using Nested Lists

```
matrix = [
   [1, 2, 3],   # Row 0
   [4, 5, 6],   # Row 1
   [7, 8, 9]    # Row 2
]
```

In this example:

- matrix[0] gives [1, 2, 3] (the first row)
- matrix[1][2] gives 6 (element at 2nd row, 3rd column)

Creating a 2D Array Dynamically

```
rows = 3  cols = 3 matrix = []

for i in range(rows):

   row = []

   for j in range(cols):

      row.append(int(input(f"Enter element at {i},{j}: ")))

   matrix.append(row)
```

This creates a 3×3 matrix by reading user input.

ALGORITHM:

1. Matrix Addition / Subtraction:

- Input matrices A and B of same dimensions (m x n)
- Create result matrix C with same dimensions
- For each i in rows and j in columns:
    - C[i][j] = A[i][j] ± B[i][j]

2. Matrix Multiplication:

- Input matrix A (m x n) and B (n x p)
- Create result matrix C (m x p) initialized with zeros
- For each i in A, j in B:
    - Compute: C[i][j] = sum(A[i][k] * B[k][j])

3. Transpose:

- Input matrix A (m x n)
- Create result matrix B (n x m)
- For each i, j: B[j][i] = A[i][j]

CODE:

OUTPUT:

CONCLUSION:

Thus we have implemented a 2- D array for matrix and matrix operation using **nested lists** in Python.

DISCUSSION AND VIVA VOCE:

- What is a two-dimensional array?
- How do you represent a 2D array in Python?
- What is the difference between 1D and 2D arrays?
- What are the conditions for matrix addition and subtraction?
- What are the conditions for matrix multiplication?
- Why is NumPy preferred for matrix operations in real-world applications?
- What is the time complexity of matrix multiplication?

Practical No. 03

AIM: Implement a python program to store employee IDs in an array those who attended the training program in random order. Perform following tasks

A. Search whether a particular employee attended a training or not using linear search
B. Search whether a particular employee attended a training or not using binary search

OBJECTIVES:

- To **develop** algorithmic problem-solving skills by implementing various techniques such as searching, sorting, and recursion for practical and real-world scenarios
- To acquire different searching manipulation techniques to be used with python array.

THEORY:

Linear search is also called as sequential search algorithm. It is the simplest searching algorithm. In Linear search, we simply traverse the list completely and match each element of the list with the item whose location is to be found. If the match is found, then the location of the item is returned; otherwise, the algorithm returns NULL. It is widely used to search an element from the unordered list, i.e., the list in which items are not sorted. The worst-case time complexity of linear search is O(n).

Binary searching

Binary search is the search technique that works efficiently on sorted lists. Hence, to search an element into some list using the binary search technique, we must ensure that the list is sorted. Binary search follows the divide and conquer approach in which the list is divided into two halves, and the item is compared with the middle element of the list. If the match is found then, the location of the middle element is returned. Otherwise, we search into either of the halves depending upon the result produced through the match.

ALGORITHM:

Function for Linear Search

```
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return True
    return False
```

Function for Binary Search (array must be sorted)

```
def binary_search(arr, target):
    low = 0
```

```
high = len(arr) - 1

    while low <= high:

    mid = (low + high) // 2

    if arr[mid] == target:

        return True

    elif arr[mid] < target:

        low = mid + 1

    else:

        high = mid - 1

return False
```

CODE:

OUTPUT:

CONCLUSION: By this way, we can perform different searching operations on any data.

CONCLUSION:

DISCUSSION AND VIVA VOCE:

What is the time complexity of linear search?
What is the time complexity of binary search?
Why do we need sorting before binary search?
Which search method is better?

Where are these search techniques used in real life?

Practical No. 04

AIM: Implement a python program to store the percentage of students in an array. write function for sorting array of floating points numbers in ascending order using

A. Selection sort and Insertion sort
B. Quick sort and bubble Sort

OBJECTIVES:

- To **develop** algorithmic problem-solving skills by implementing various techniques such as searching, sorting, and recursion for practical and real-world scenarios
- To acquire different searching manipulation techniques to be used with python array.

THEORY:

Write a short theory along with algorithms related to these sorting algorithm

- **Selection Sort**
- **Insertion Sort**
- **Bubble Sort**
- **Quick Sort**

ALGORITHM:

Algorithm Overview

1. **Selection Sort**
    - Repeatedly selects the smallest element from the unsorted part and places it at the beginning.
    - Time Complexity: **O(n²)**
2. **Insertion Sort**
    - Builds the sorted list one element at a time by inserting each new element into its proper place.
    - Time Complexity: **O(n²)**
3. **Bubble Sort**
    - Repeatedly swaps adjacent elements if they are in the wrong order.
    - Time Complexity: **O(n²)**
4. **Quick Sort**
    - A divide-and-conquer algorithm that partitions the array around a pivot and recursively sorts subarrays.
    - Time Complexity: **O(n log n)** (average case)

CODE:

OUTPUT:

CONCLUSION: By this way, we can perform different searching operations on any data.

CONCLUSION:

DISCUSSION AND VIVA VOCE:

Q1. What is the difference between selection sort and insertion sort?
Q2. Why is bubble sort not efficient?
Q3. What is the best sorting algorithm among the four implemented?
Q4. What does “in-place sorting” mean?
Q5. Where are sorting algorithms used in real life?