# **Exercise 1: Inventory Management System**

#### Scenario:

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

## Steps:

#### 1. Understand the Problem:

- o Explain why data structures and algorithms are essential in handling large inventories.
- Discuss the types of data structures suitable for this problem.

#### 2. Setup:

Create a new project for the inventory management system.

### 3. Implementation:

- o Define a class Product with attributes like **productId**, **productName**, **quantity**, and **price**.
- o Choose an appropriate data structure to store the products (e.g., ArrayList, HashMap).
- o Implement methods to add, update, and delete products from the inventory.

#### 4. Analysis:

- Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.
- Discuss how you can optimize these operations.

## **Exercise 2: E-commerce Platform Search Function**

#### Scenario:

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

## Steps:

### 1. Understand Asymptotic Notation:

- o Explain Big O notation and how it helps in analyzing algorithms.
- o Describe the best, average, and worst-case scenarios for search operations.

### 2. Setup:

 Create a class Product with attributes for searching, such as productId, productName, and category.

#### 3. Implementation:

- o Implement linear search and binary search algorithms.
- o Store products in an array for linear search and a sorted array for binary search.

### 4. Analysis:

- o Compare the time complexity of linear and binary search algorithms.
- Discuss which algorithm is more suitable for your platform and why.

# **Exercise 3: Sorting Customer Orders**

#### Scenario:

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritizing high-value orders.

### Steps:

#### 1. Understand Sorting Algorithms:

o Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).

### 2. Setup:

o Create a class **Order** with attributes like **orderId**, **customerName**, and **totalPrice**.

### 3. Implementation:

- o Implement **Bubble Sort** to sort orders by **totalPrice**.
- Implement Quick Sort to sort orders by totalPrice.

#### 4. Analysis:

- o Compare the performance (time complexity) of Bubble Sort and Quick Sort.
- o Discuss why Quick Sort is generally preferred over Bubble Sort.

# **Exercise 4: Employee Management System**

#### Scenario:

You are developing an employee management system for a company. Efficiently managing employee records is crucial.

#### Steps:

### 1. Understand Array Representation:

Explain how arrays are represented in memory and their advantages.

#### 2. Setup:

Create a class Employee with attributes like employeeld, name, position, and salary.

#### 3. Implementation:

- Use an array to store employee records.
- o Implement methods to add, search, traverse, and delete employees in the array.

## 4. Analysis:

- o Analyze the time complexity of each operation (add, search, traverse, delete).
- o Discuss the limitations of arrays and when to use them.

# **Exercise 5: Task Management System**

#### Scenario:

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

### Steps:

#### 1. Understand Linked Lists:

o Explain the different types of linked lists (Singly Linked List, Doubly Linked List).

## 2. Setup:

o Create a class **Task** with attributes like **taskId**, **taskName**, and **status**.

### 3. Implementation:

- Implement a singly linked list to manage tasks.
- o Implement methods to add, search, traverse, and delete tasks in the linked list.

### 4. Analysis:

- Analyze the time complexity of each operation.
- o Discuss the advantages of linked lists over arrays for dynamic data.

# **Exercise 6: Library Management System**

#### Scenario:

You are developing a library management system where users can search for books by title or author.

# Steps:

#### 1. Understand Search Algorithms:

Explain linear search and binary search algorithms.

#### 2. Setup:

o Create a class **Book** with attributes like **bookId**, **title**, and **author**.

### 3. Implementation:

- Implement linear search to find books by title.
- o Implement binary search to find books by title (assuming the list is sorted).

## 4. Analysis:

- o Compare the time complexity of linear and binary search.
- o Discuss when to use each algorithm based on the data set size and order.

# **Exercise 7: Financial Forecasting**

#### Scenario:

You are developing a financial forecasting tool that predicts future values based on past data.

#### Steps:

## 1. Understand Recursive Algorithms:

Explain the concept of recursion and how it can simplify certain problems.

#### 2. Setup:

• Create a method to calculate the future value using a recursive approach.

### 3. Implementation:

o Implement a recursive algorithm to predict future values based on past growth rates.

## 4. Analysis:

- o Discuss the time complexity of your recursive algorithm.
- o Explain how to optimize the recursive solution to avoid excessive computation.