**DATA STRUCTURES   
ASSIGNMENT 1  
Due Date: Monday 12th August 2024 at 11:00 PM**

***Note:***

***Make use of string data type is strictly prohibited, instead create your own string ADT or make use of String library if required.***

This problem statement outlines the basic functionality and requirements for an inventory management system using C++. The implementation should focus on using classes and object-oriented programming and data structures principles to manage and manipulate inventory data effectively.

**Objective:**  
Design and implement an inventory management system in C++ using classes. The system should allow users to manage inventory items, track stock levels, and perform essential operations like adding, removing, and searching inventory items.

**Requirements**

1. **Inventory Item Class:**
   * **Attributes:**
     + itemID (unique identifier)
     + itemName (name of the item)
     + quantity (current stock level)
     + price (price per item)
   * **Methods:**
     + Constructors (default and parameterized)
     + Getters and setters for each attribute
     + Overloading operators (where required)
     + Display item details
2. **Inventory Class:**
   * **Attributes:**
     + items (a collection of inventory items, e.g., Array as ADT)
   * **Methods:**
     + Add a new item
     + Remove last item
     + Remove an existing item by itemID
     + Search for an item by itemID or itemName
     + Display all items in the inventory
3. **User Interface:**
   * A menu-driven interface to allow users to perform various operations:
     + Add a new item
     + Remove last item
     + Remove any item
     + Search for an item
     + Display all items
     + Exit the application

**Implementation Details**

1. **Class Definitions:**
   * Define Item class with the specified attributes and methods.
   * Define Inventory class to manage a collection of Item objects.
2. **Data Structures:**
   * Use appropriate data structures (e.g., Your own created Arrays as discussed) to store and manage the inventory items.
3. **File Handling:**
   * Optional: Implement file handling to load inventory data from a file at the start and save it back to the file on exit.
4. **Error Checking:**
   * Implement error checking and exception handling to manage cases like invalid input, non-existent item IDs, etc.
5. **Modularity:**
   * Ensure the code is modular and well-organized, separating class definitions, implementations, and the user interface.

**Example Scenario**

1. **Add New Item:**
   * User adds a new item with itemID = 101, itemName = "Laptop", quantity = 10, price = 1500.00.
2. **Remove Item:**
   * User removes an item with itemID = 101.
3. **Search Item:**
   * User searches for an item by itemName = "Mouse".
4. **Display Items:**
   * User displays all items in the inventory.