**ARTIFICIAL INTELLIGENCE DEPARTMENT**

# Total Marks:

**Obtained Marks:**

**PROJECT NO#1**

**REPORT**

Object Oriented Programming Techniques (LAB)

**INVENTORY MANAGEMENT SYSTEM**

# Submitted To: Sir Hamza Imran

**Student Name:** **M.Muneeb, Muddasir, Adnan**

**Section Name:** **BSAI-2A**

# Reg. Number: 22108190, 22108251,22108166

**ARTIFICIAL INTELLIGENCE DEPARTMENT**

**Introduction: -**

An Inventory Management System (IMS) is a tool that helps businesses and organizations track goods across various stages along the supply chain—from acquisition from vendors to storage in a warehouse and finally to the point of sale. It plays a pivotal role in maintaining the optimum stock levels to ensure smooth operations and minimize costs. An Inventory Management System designed using Object-Oriented Programming (OOP) techniques in C++ is not just a step towards efficient code organization but also towards creating a scalable, reusable, and robust application. Object-Oriented principles like Encapsulation, Inheritance, and Polymorphism can significantly contribute to the code's structure and readability.

**Procedure: -**

The procedure for using the Inventory Management System program can vary slightly depending on the specific features and functions of the software, but here is a general overview of the steps involved:

**Launch the Inventory Management System Program:**

Open a code editor or integrated development environment (IDE) on your computer or device.

**Copy and Paste Code:**

Copy the provided C++ code for the Inventory Management System and paste it into your code editor or IDE.

**Compile the Code:**

Compile the code using the appropriate build or compile command in your chosen development environment. Make sure you have a C++ compiler (such as g++) installed on your system.

**Run the Program:**

Once the code is successfully compiled without errors, you can run the program by executing it in your code editor or IDE.

**Interact with the Program:**

After running the program, you will see a menu displayed in the console, which allows you to interact with the Inventory Management System. The menu will include the following options:

1. Add Item
2. Display Items
3. Delete Item
4. Exit

**Choose an Option:**

Use the numerical options provided in the menu to choose what action you want to perform in the Inventory Management System.

**Add Items:**

If you choose option 1, you can add items to the inventory. The program will prompt you to enter the name, quantity, and price of the item you want to add.

**Display Items:**

If you choose option 2, the program will display the list of items in your inventory, including their names, quantities, and prices.

**Delete Items:**

If you choose option 3, you can delete an item from the inventory by entering the name of the item you want to delete.

**Exit the Program:**

If you choose option 4, the program will exit and display a message indicating that you are exiting the system.

**Repeat or Terminate:**

You can repeat these steps to continue managing your inventory or choose option 4 to exit the program when you are done.

This procedure outlines how to run and interact with the Inventory Management System program on your computer or device. It allows you to add, display, and delete items in your inventory efficiently.

**Objective: -**

The objective of the inventory management system implemented in the provided code is to create a simple console-based tool for managing an inventory of items. This system allows users to perform functions.

**Scope of Inventory Management System: -**

1. **Small Retail Business**: This could be the base for a simple stock-keeping unit (SKU) management system for a small store or an individual seller.
2. **Personal Inventory**: Someone might use this for keeping track of personal collections, like books, digital assets, etc.
3. **Educational Purposes**: It serves as a good introductory example for learning object-oriented programming and C++ specifically.
4. **Expansion Possibility**: This could be further developed into a full-fledged Inventory Management System with additional features like search, sorting, batch updates, etc.
5. **Integration with IoT**: With modifications, this could be part of a more complex system integrated with IoT devices for real-time tracking.
6. **Limitations**: Due to its basic nature, this would not be suitable for large scale operations that require high-concurrency and persistence, unless it is significantly extended and optimized.

**Future Scope of Inventory Management System: -**

1. **Database Integration**: The current system uses an in-memory array to store inventory items. In the future, it can be extended to interact with a relational or NoSQL database for persistent storage.
2. **RESTful API Support**: Extending the system to have API support will make it easier to integrate with web and mobile applications.
3. **Machine Learning for Inventory Prediction**: The system could be enhanced to predict when inventory for a specific item may run out, helping in more efficient stock management.
4. **User Authentication & Role-Based Access**: Implementing user authentication and defining roles could allow for secure and differentiated access levels to the inventory system.
5. **Batch Operations**: Future implementations could provide functionality for performing batch operations, like bulk insert, update, or deletion of items.
6. **Real-Time Monitoring and Notifications**: The system could provide real-time notifications when stock reaches a certain limit or changes in some other critical way.

**Features of Inventory Management System: -**

The **InventoryManager** class offers four main functionalities:

1. **Add Item:** Allows users to add new items to the inventory by specifying the name, quantity, and price.
2. **Display Items:** Shows all items currently in the inventory along with their quantities and prices.
3. **Delete Item:** Allows users to remove an item from the inventory by providing its name.
4. **Exit**: Exits the program

**Disadvantages of Inventory Management System: -**

Your C++ code for an Inventory Management System looks clean and straightforward. However, you asked about the disadvantages of Inventory Management Systems in general. Here are some:

1. **Cost**: Initial setup and maintenance can be expensive, especially for small businesses.
2. **Complexity**: With numerous items and various categories, the system can become complex and difficult to manage.
3. **Data Errors**: Mistakes in inputting data can lead to incorrect inventory levels, affecting business decisions.
4. **Dependence on Technical Infrastructure**: Power outages, system crashes, or network issues can disrupt the inventory system.
5. **Limited Customization**: Off-the-shelf systems may not meet the specific needs of a business, requiring additional customization that can be time-consuming and costly.

**Topics Related to OOPs: -**

1. **Encapsulation**: The concept of bundling data and methods that operate on that data within a single unit or object.
2. **Inheritance**: The mechanism where a new class (derived class) inherits attributes and behaviors (methods) from an existing class (base class).
3. **Polymorphism**: The ability of different objects to respond to the same method call in a way that is specific to their individual types.
4. **Abstraction**: Hiding the complex reality while exposing only the essential features of an object.

### **Classes**

1. **Constructor**: Special method in a class used for initializing new objects.
2. **Destructor**: Method that is automatically invoked when an object is destroyed to clean up resources.
3. **Class Variables**: Variables that are shared across all instances of a class.
4. **Instance Variables**: Variables that are unique to each instance of a class.
5. **Static Methods**: Methods that belong to the class rather than any particular object instance.
6. **Class Methods**: Methods that are bound to the class and not the instance of the class.
7. **Overloading**: The ability to define the same method multiple times with different parameters.
8. **Overriding**: The ability of a subclass to provide its own implementation for a method that is already defined in its superclass.

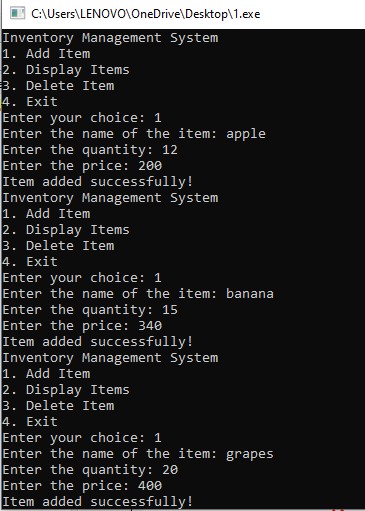
### **Structures**

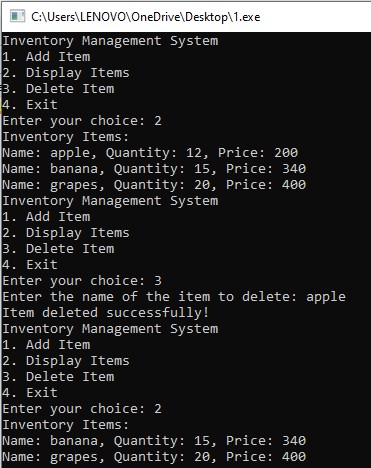
1. **Declaration**: Defining a new structure type with keyword **struct** (C, C++) or **class** with public access (C++).
2. **Initialization**: Creating instances of the structure.
3. **Nested Structures**: Structures within structures.
4. **Arrays of Structures**: Using arrays to store multiple structure elements.
5. **Structure Padding**: Compiler automatically adds padding to ensure memory alignment.

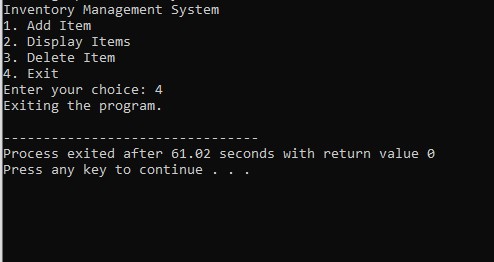
### **Functions**

1. **Function Declaration**: Specifies a function's name, return type, and parameters.
2. **Function Definition**: Provides the actual body of the function.
3. **Function Overloading**: Defining multiple functions with the same name but different parameters.
4. **Function Overriding**: Providing a new implementation for a function in a subclass.
5. **Recursion**: Functions calling themselves.
6. **Lambda Functions**: Anonymous functions often used for short, throwaway functionalities.
7. **Variadic Functions**: Functions that accept an indefinite number of arguments.
8. **Pure Functions**: Functions where the output is determined only by its input, without any observable side effects.

**Output: -**

****

****

****

**Explanation: -**

1. Include the necessary header files: <iostream> for input-output operations and <string> for using the string data type.
2. Use the standard namespace. It allows you to use classes and functions in the std namespace without qualifying them with std::.
3. Define a class named InventoryManager.
4. Public Section: - Declare a constructor, destructor, and a public method run ().
5. Private Section: - Define a nested private struct named Item with three fields
6. Declare a pointer named inventory for dynamic allocation of an array of Item objects, and an integer size to keep track of the number of items
7. Constructor: - Initialize inventory pointer to nullptr and size to 0.
8. Destructor: - Free the dynamically allocated memory if inventory is not null.
9. Contains the main loop for running the inventory system.
10. Display the menu options.
11. Adds a new item to the inventory.
12. Displays all items in the inventory.
13. Deletes an item from the inventory based on its name.
14. Creates an object of InventoryManager and runs the system.

***\*\*The End\*\****