INVENTORY MANAGEMENT SYSTEM

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_ITEMS 100

typedef struct {
    int id;
    char name[50];
    int quantity;
    float price;
} Item;

void addItem(Item items[], int *count) {
    if (*count >= MAX_ITEMS) {
        printf("Inventory full!\n");
        return;
    }

    printf("Enter ID: ");
    scanf("%d", &items[*count].id);
    printf("Enter Name: ");
    scanf(" %[^\n]", items[*count].name);
```

```
    printf("Enter Quantity: ");
    scanf("%d", &items[*count].quantity);
    printf("Enter Price: ");
    scanf("%f", &items[*count].price);
    (*count)++;
    printf("Item added successfully!\n");
}

void viewItems(Item items[], int count) {
    if (count == 0) {
        printf("No items to display.\n");
        return;
    }

    printf("\n%-5s %-20s %-10s %-10s\n", "ID", "Name", "Quantity", "Price");
    for (int i = 0; i < count; i++) {
        printf("%-5d %-20s %-10d %-10.2f\n", items[i].id, items[i].name, items[i].quantity,
items[i].price);
    }
}

void searchItem(Item items[], int count) {
    int id, found = 0;
    printf("Enter item ID to search: ");
    scanf("%d", &id);
```

```
    for (int i = 0; i < count; i++) {
        if (items[i].id == id) {
            printf("Item found:\n");
            printf("ID: %d\nName: %s\nQuantity: %d\nPrice: %.2f\n",
                items[i].id, items[i].name, items[i].quantity, items[i].price);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Item not found.\n");
    }
}

void updateItem(Item items[], int count) {
    int id, found = 0;
    printf("Enter item ID to update: ");
    scanf("%d", &id);

    for (int i = 0; i < count; i++) {
        if (items[i].id == id) {
            printf("Enter new name: ");
            scanf(" %[^\n]", items[i].name);
            printf("Enter new quantity: ");
            scanf("%d", &items[i].quantity);
```

```
            printf("Enter new price: ");
            scanf("%f", &items[i].price);
            printf("Item updated successfully!\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Item not found.\n");
    }
}

void deleteItem(Item items[], int *count) {
    int id, found = 0;
    printf("Enter item ID to delete: ");
    scanf("%d", &id);

    for (int i = 0; i < *count; i++) {
        if (items[i].id == id) {
            for (int j = i; j < *count - 1; j++) {
                items[j] = items[j + 1];
            }
            (*count)--;
            printf("Item deleted successfully!\n");
            found = 1;
```

```
            break;
        }
    }

    if (!found) {
        printf("Item not found.\n");
    }
}

int main() {
    Item items[MAX_ITEMS];
    int count = 0;
    int choice;

    do {
        printf("\n=== Inventory Management System ===\n");
        printf("1. Add Item\n");
        printf("2. View Items\n");
        printf("3. Search Item\n");
        printf("4. Update Item\n");
        printf("5. Delete Item\n");
        printf("0. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
```

```
            case 1: addItem(items, &count); break;
            case 2: viewItems(items, count); break;
            case 3: searchItem(items, count); break;
            case 4: updateItem(items, count); break;
            case 5: deleteItem(items, &count); break;
            case 0: printf("Exiting...\n"); break;
            default: printf("Invalid choice.\n");
        }
    } while (choice != 0);

    return 0;
}
```

INVENTORY MANAGEMENT SYSTEM

1. Introduction The Inventory Management System developed in C is a console-based application designed to help businesses and individuals keep track of stock items efficiently. This system facilitates inventory control by allowing users to manage product records including item names, quantities, and prices. It serves as a practical implementation of fundamental programming concepts in C.

2. Objectives

To design a system that helps in maintaining accurate and up-to-date inventory records.

To implement basic Add, View, Search, Update, Delete operations.

To apply structured programming principles using the C language.

3. System Features

Add New Item: Users can input item details such as ID, name, quantity, and price. This function simplifies the tracking of stock levels and item information.

View All Items: Displays a complete list of inventory items. It is helpful for reviewing the current stock and verifying quantities.

Search Item: Allows searching for a specific item using its ID. This makes it easy to find product information without browsing the entire list.

Update Item: Enables modification of item details. It ensures data accuracy and helps in keeping the inventory up-to-date.

Delete Item: Permits removal of an item from the inventory. This feature is essential for discarding discontinued or obsolete products.

4. Technical Details

Programming Language: C, chosen for its simplicity and control over system-level operations.

Compiler Used: GCC / Turbo C, both widely used in academic environments.

Key Concepts: Structures for grouping related data, Conditional Statements for decision making, Looping for repetition of tasks, and Functions for organizing the code into manageable sections.

Development Environment: Compatible with most C compilers and suitable for cross-platform use.

5. Problem and Complexity Although the system achieves its basic objectives, it currently lacks advanced features. There is no data persistence, user interface, or advanced error handling. These limitations make it unsuitable for large-scale or commercial applications. The current implementation also faces constraints related to manual data entry and lack of real-time updates. The complexity of the system increases with the number of items, as data must be handled manually and sequentially, which could result in performance issues

6. Future Improvement

Introduce file handling for storing and retrieving data permanently.

Develop a user-friendly interface to improve usability and accessibility.

Add database support for efficient and scalable data management.

Enhance input validation and implement robust error handling mechanisms.

Integrate reporting and analytics features to extract insights from inventory data.

Allow sorting and filtering options for easier inventory navigation.

Consider network features for multi-user access in a shared environment.