Problem 1: Inventory Management System

Description: Develop an inventory management system for an e-commerce platform.

Requirements:

- Use a structure to define an item with fields: itemID, itemName, price, and quantity.
- Use an array of structures to store the inventory.
- Implement functions to add new items, update item details (call by reference), and display the entire inventory (call by value).
- Use a loop to iterate through the inventory.
- Use static to keep track of the total number of items added.

Output Expectations:

- Display the updated inventory after each addition or update.
- Show the total number of items.

Sol:
```
#include <stdio.h>
#define MAX_ITEMS 100

// Typedef for Item structure
typedef struct {
    int itemID;
    char itemName[50];
    float price;
    int quantity;
} Item;

// Function prototypes
void addItem(Item inventory[], int *totalItems);
void updateItem(Item *item);
void displayInventory(Item inventory[], int totalItems);
```

```
int main() {
    Item inventory[MAX_ITEMS];
    static int totalItems = 0;

    addItem(inventory, &totalItems);
    addItem(inventory, &totalItems);
    displayInventory(inventory, totalItems);

    if (totalItems > 0) {
        updateItem(&inventory[0]);
        displayInventory(inventory, totalItems);
    }

    return 0;
}

void addItem(Item inventory[], int *totalItems) {
    if (*totalItems >= MAX_ITEMS) {
        printf("Inventory is full.\n");
        return;
    }

    printf("Enter Item ID: ");
    scanf("%d", &inventory[*totalItems].itemID);
    printf("Enter Item Name: ");
    scanf("%s", inventory[*totalItems].itemName);
    printf("Enter Price: ");
    scanf("%f", &inventory[*totalItems].price);
    printf("Enter Quantity: ");
```

```
    scanf("%d", &inventory[*totalItems].quantity);

    (*totalItems)++;
}

void updateItem(Item *item) {
    printf("\nUpdating item: %s\n", item->itemName);
    printf("Enter new Price: ");
    scanf("%f", &item->price);
    printf("Enter new Quantity: ");
    scanf("%d", &item->quantity);
}

void displayInventory(Item inventory[], int totalItems) {
    printf("\nInventory:\n");
    printf("ID\tName\tPrice\tQuantity\n");
    for (int i = 0; i < totalItems; i++) {
        printf("%d\t%s\t%.2f\t%d\n", inventory[i].itemID, inventory[i].itemName,
inventory[i].price, inventory[i].quantity);
    }
    printf("Total items: %d\n", totalItems);
}
```

O/p:

```
Enter Item ID: 1
Enter Item Name: laptop
Enter Price: 7000
Enter Quantity: 10
Enter Item ID: 2
Enter Item Name: phone
```

Enter Price: 600

Enter Quantity: 5

Inventory:

ID	Name	Price	Quantity
1	laptop	7000.00	10
2	phone	600.00	5

Total items: 2

Updating item: laptop

Enter new Price: phone 10000

Enter new Quantity: 11

Inventory:

ID	Name	Price	Quantity
1	laptop	10000.00	11
2	phone	600.00	5

Total items: 2

Problem 2: Order Processing System

Description: Create an order processing system that calculates the total order cost and applies discounts.

Requirements:

- Use a structure for Order containing fields for orderID, customerName, items (array), and totalCost.
- Use const for the discount rate.
- Implement functions for calculating the total cost (call by value) and applying the discount (call by reference).
- Use a loop to process multiple orders.

Output Expectations:

- Show the total cost before and after applying the discount for each order.

Sol:
```
#include <stdio.h>
#define MAX_ITEMS 5
#define DISCOUNT_RATE 0.1f  // 10% discount

// Typedef for Item and Order structures
typedef struct {
    int itemID;
    char itemName[30];
    float price;
    int quantity;
} Item;

typedef struct {
    int orderID;
    char customerName[30];
    Item items[MAX_ITEMS];
    int itemCount;
    float totalCost;
} Order;

// Function prototypes
float calculateTotalCost(Order order);
void applyDiscount(Order *order);
void processOrder(Order *order);

int main() {
    Order order;
```

```
    processOrder(&order);

    printf("\nOrder Summary:\n");
    printf("Order ID: %d\nCustomer Name: %s\n", order.orderID,
order.customerName);
    printf("Total Cost Before Discount: %.2f\n", calculateTotalCost(order));
    applyDiscount(&order);
    printf("Total Cost After Discount: %.2f\n", order.totalCost);

    return 0;
}

void processOrder(Order *order) {
    printf("Enter Order ID: ");
    scanf("%d", &order->orderID);
    printf("Enter Customer Name: ");
    scanf("%s", order->customerName);

    printf("Enter number of items: ");
    scanf("%d", &order->itemCount);

    for (int i = 0; i < order->itemCount; i++) {
        printf("\nItem %d:\n", i + 1);
        printf("Enter Item ID: ");
        scanf("%d", &order->items[i].itemID);
        printf("Enter Item Name: ");
        scanf("%s", order->items[i].itemName);
        printf("Enter Price: ");
        scanf("%f", &order->items[i].price);
```

```
        printf("Enter Quantity: ");
        scanf("%d", &order->items[i].quantity);
    }
    order->totalCost = calculateTotalCost(*order);
}

float calculateTotalCost(Order order) {
    float total = 0;
    for (int i = 0; i < order.itemCount; i++) {
        total += order.items[i].price * order.items[i].quantity;
    }
    return total;
}

void applyDiscount(Order *order) {
    order->totalCost -= order->totalCost * DISCOUNT_RATE;
}
```

O/p:

```
Enter Order ID: 1
Enter Customer Name: likitha
Enter number of items: 2

Item 1:
Enter Item ID: 101
Enter Item Name: books
Enter Price: 7000
Enter Quantity: 5

Item 2:
```

Enter Item ID: 102

Enter Item Name: mobile

Enter Price: 5000

Enter Quantity: 2

Order Summary:

Order ID: 1

Customer Name: likitha

Total Cost Before Discount: 45000.00

Total Cost After Discount: 40500.00

Problem 3: Customer Feedback System

Description: Develop a feedback system that categorizes customer feedback based on ratings.

Requirements:

- Use a structure to define Feedback with fields for customerID, feedbackText, and rating.
- Use a switch case to categorize feedback (e.g., Excellent, Good, Average, Poor).
- Store feedback in an array.
- Implement functions to add feedback and display feedback summaries using loops.

Output Expectations:

- Display categorized feedback summaries.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_FEEDBACK 5

// Typedef for Feedback structure

typedef struct {

```
    int customerID;
    char feedbackText[100];
    int rating;  // Rating from 1 to 5
} Feedback;

// Function prototypes
void addFeedback(Feedback feedbacks[], int *totalFeedbacks);
void displayFeedbackSummary(const Feedback feedbacks[], int totalFeedbacks);
void categorizeFeedback(int rating);

int main() {
    Feedback feedbacks[MAX_FEEDBACK];
    int totalFeedbacks = 0;

    int choice;
    do {
        printf("\n1. Add Feedback\n2. Display Feedback Summary\n3. Exit\nChoose an option: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                addFeedback(feedbacks, &totalFeedbacks);
                break;
            case 2:
                displayFeedbackSummary(feedbacks, totalFeedbacks);
                break;
            case 3:
                printf("Exiting the program.\n");
                break;
```

```
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 3);

    return 0;
}

void addFeedback(Feedback feedbacks[], int *totalFeedbacks) {
    if (*totalFeedbacks >= MAX_FEEDBACK) {
        printf("Feedback storage is full.\n");
        return;
    }

    printf("Enter Customer ID: ");
    scanf("%d", &feedbacks[*totalFeedbacks].customerID);
    printf("Enter Feedback Text: ");
    getchar();  // Clear newline character
    fgets(feedbacks[*totalFeedbacks].feedbackText,
sizeof(feedbacks[*totalFeedbacks].feedbackText), stdin);
    strtok(feedbacks[*totalFeedbacks].feedbackText, "\n");  // Remove newline
character
    printf("Enter Rating (1 to 5): ");
    scanf("%d", &feedbacks[*totalFeedbacks].rating);

    (*totalFeedbacks)++;
}

void displayFeedbackSummary(const Feedback feedbacks[], int totalFeedbacks) {
    printf("\nFeedback Summary:\n");
```

```
    for (int i = 0; i < totalFeedbacks; i++) {
        printf("Customer ID: %d\n", feedbacks[i].customerID);
        printf("Feedback: %s\n", feedbacks[i].feedbackText);
        printf("Rating: %d\nCategory: ", feedbacks[i].rating);
        categorizeFeedback(feedbacks[i].rating);
        printf("\n");
    }
}

void categorizeFeedback(int rating) {
    switch (rating) {
        case 5:
            printf("Excellent");
            break;
        case 4:
            printf("Good");
            break;
        case 3:
            printf("Average");
            break;
        case 2:
            printf("Poor");
            break;
        case 1:
            printf("Very Poor");
            break;
        default:
            printf("Invalid Rating");
    }
```

}
O/p:

1. Add Feedback
2. Display Feedback Summary
3. Exit
Choose an option: 1
Enter Customer ID: 101
Enter Feedback Text: good service
Enter Rating (1 to 5): 5

1. Add Feedback
2. Display Feedback Summary
3. Exit
Choose an option: 1
Enter Customer ID: 102
Enter Feedback Text: Not satisfied
Enter Rating (1 to 5): 3

1. Add Feedback
2. Display Feedback Summary
3. Exit
Choose an option: 2

Feedback Summary:
Customer ID: 101
Feedback: good service
Rating: 5
Category: Excellent

Customer ID: 102

Feedback: Not satisfied

Rating: 3

Category: Average

1. Add Feedback

2. Display Feedback Summary

3. Exit

Choose an option:

Problem 4: Payment Method Selection

Description: Write a program that handles multiple payment methods and calculates transaction charges.

Requirements:

- Use a structure for Payment with fields for method, amount, and transactionCharge.
- Use const for fixed transaction charges.
- Use a switch case to determine the transaction charge based on the payment method.
- Implement functions for processing payments and updating transaction details (call by reference).

Output Expectations:

- Show the payment details including the method and transaction charge.

Sol: #include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX_ITEMS 100

#define DISCOUNT_RATE 0.1

#define CREDIT_CARD_CHARGE 0.02

```
#define DEBIT_CARD_CHARGE 0.01
#define PAYPAL_CHARGE 0.03

typedef struct {
    char method[20];
    float amount;
    float transactionCharge;
} Payment;

// Function to convert string to lowercase for case-insensitive comparison
void toLowerCase(char *str) {
    for (int i = 0; str[i]; i++) {
        str[i] = tolower(str[i]);
    }
}

void processPayment(Payment *payment) {

    printf("Enter Payment Method (Credit Card, Debit Card, PayPal): ");
    fgets(payment->method, sizeof(payment->method), stdin);
    payment->method[strcspn(payment->method, "\n")] = '\0';  // Remove the
newline character if any

    toLowerCase(payment->method);  // Convert to lowercase for case-insensitive
comparison

    printf("Enter Payment Amount: ");
    scanf("%f", &payment->amount);

    if (payment->amount < 0) {
```

```
        printf("Invalid amount. Amount cannot be negative.\n");
        return;
    }

    if (strcmp(payment->method, "credit card") == 0) {
        payment->transactionCharge = payment->amount *
CREDIT_CARD_CHARGE;
    } else if (strcmp(payment->method, "debit card") == 0) {
        payment->transactionCharge = payment->amount *
DEBIT_CARD_CHARGE;
    } else if (strcmp(payment->method, "paypal") == 0) {
        payment->transactionCharge = payment->amount * PAYPAL_CHARGE;
    } else {
        printf("Invalid Payment Method. No charge applied.\n");
        payment->transactionCharge = 0;
    }
    printf("Payment Method: %s\n", payment->method);
    printf("Transaction Charge: %.2f\n", payment->transactionCharge);
}

int main() {
    Payment payment;

    processPayment(&payment);

    return 0;
}
```

O/p:

Enter Payment Method (Credit Card, Debit Card, PayPal): Py ayPal

Enter Payment Amount: 5000

Payment Method: PayPal

Transaction Charge: 150.00

Problem 5: Shopping Cart System

Description: Implement a shopping cart system that allows adding, removing, and viewing items.

Requirements:

- Use a structure for CartItem with fields for itemID, itemName, price, and quantity.
- Use an array to store the cart items.
- Implement functions to add, remove (call by reference), and display items (call by value).
- Use loops for iterating through cart items.

Output Expectations:

- Display the updated cart after each operation.

Sol:

```
#include <stdio.h>

#define MAX_ITEMS 10

// Structure for Cart Item
typedef struct {
    int itemID;
    char itemName[50];
    float price;
    int quantity;
} CartItem;

// Function to add item to cart
void addItem(CartItem cart[], int *totalItems) {
    if (*totalItems < MAX_ITEMS) {
        printf("\nEnter Item ID: ");
```

```
        scanf("%d", &cart[*totalItems].itemID);
        getchar();  // Consume newline
        printf("Enter Item Name: ");
        fgets(cart[*totalItems].itemName, sizeof(cart[*totalItems].itemName), stdin);
        cart[*totalItems].itemName[strcspn(cart[*totalItems].itemName, "\n")] = '\0';
// Remove newline
        printf("Enter Item Price: ");
        scanf("%f", &cart[*totalItems].price);
        printf("Enter Item Quantity: ");
        scanf("%d", &cart[*totalItems].quantity);
        (*totalItems)++;
    } else {
        printf("Cart is full!\n");
    }
}

// Function to remove item from cart (by reference)
void removeItem(CartItem cart[], int *totalItems) {
    int itemID;
    printf("Enter Item ID to remove: ");
    scanf("%d", &itemID);
    int found = 0;
    for (int i = 0; i < *totalItems; i++) {
        if (cart[i].itemID == itemID) {
            for (int j = i; j < *totalItems - 1; j++) {
                cart[j] = cart[j + 1];  // Shift items left
            }
            (*totalItems)--;  // Decrease the item count
            found = 1;
```

```
            break;
        }
    }
    if (!found) {
        printf("Item not found in cart.\n");
    }
}

// Function to display cart items
void displayCart(CartItem cart[], int totalItems) {
    printf("\nShopping Cart:\n");
    for (int i = 0; i < totalItems; i++) {
        printf("Item ID: %d, Name: %s, Price: %.2f, Quantity: %d\n",
               cart[i].itemID, cart[i].itemName, cart[i].price, cart[i].quantity);
    }
}

int main() {
    CartItem cart[MAX_ITEMS];
    int totalItems = 0;
    int choice;

    while (1) {
        printf("\n1. Add Item\n2. Remove Item\n3. Display Cart\n4. Exit\nEnter
choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

```
            addItem(cart, &totalItems);
            break;
        case 2:
            removeItem(cart, &totalItems);
            break;
        case 3:
            displayCart(cart, totalItems);
            break;
        case 4:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice, try again.\n");
      }
   }
}
```

O/p:

```
1. Add Item
2. Remove Item
3. Display Cart
4. Exit
Enter choice: 1

Enter Item ID: 101
Enter Item Name: gold
Enter Item Price: 6000
Enter Item Quantity: 2
```

1. Add Item
2. Remove Item
3. Display Cart
4. Exit
Enter choice: 1

Enter Item ID: 102
Enter Item Name: silver
Enter Item Price: 7000
Enter Item Quantity: 3

1. Add Item
2. Remove Item
3. Display Cart
4. Exit
Enter choice: 3

Shopping Cart:
Item ID: 101, Name: gold, Price: 6000.00, Quantity: 2
Item ID: 102, Name: silver, Price: 7000.00, Quantity: 3

1. Add Item
2. Remove Item
3. Display Cart
4. Exit
Enter choice: 2
Enter Item ID to remove: 102

1. Add Item

2. Remove Item

3. Display Cart

4. Exit

Enter choice: 3

Shopping Cart:

Item ID: 101, Name: gold, Price: 6000.00, Quantity: 2

1. Add Item

2. Remove Item

3. Display Cart

4. Exit

Enter choice:

Problem 6: Product Search System

Description: Create a system that allows searching for products by name or ID.

Requirements:

- Use a structure for Product with fields for productID, productName, category, and price.
- Store products in an array.
- Use a loop to search for a product.
- Implement functions for searching by name (call by value) and updating details (call by reference).

Output Expectations:

- Display product details if found or a message indicating the product is not found.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_PRODUCTS 10

```
// Structure for Product
typedef struct {
    int productID;
    char productName[50];
    float price;
    char category[50];
} Product;

// Function to search product by name
void searchByName(Product products[], int totalProducts, char *name) {
    int found = 0;
    for (int i = 0; i < totalProducts; i++) {
        if (strstr(products[i].productName, name)) {
            printf("Product found: ID: %d, Name: %s, Price: %.2f, Category: %s\n",
                   products[i].productID, products[i].productName,
                   products[i].price, products[i].category);
            found = 1;
        }
    }
    if (!found) {
        printf("No product found with the name \"%s\"\n", name);
    }
}

// Function to search product by ID
void searchByID(Product products[], int totalProducts, int id) {
    for (int i = 0; i < totalProducts; i++) {
        if (products[i].productID == id) {
```

```
            printf("Product found: ID: %d, Name: %s, Price: %.2f, Category: %s\n",
                products[i].productID, products[i].productName,
                products[i].price, products[i].category);
            return;
        }
    }
    printf("Product with ID %d not found.\n", id);
}

int main() {
    Product products[MAX_PRODUCTS] = {
        {1, "Laptop", 1500.50, "Electronics"},
        {2, "Smartphone", 800.00, "Electronics"},
        {3, "Chair", 150.75, "Furniture"},
        {4, "Table", 250.30, "Furniture"}
    };

    int choice, id;
    char name[50];
    int totalProducts = 4;  // Initial number of products

    printf("Search products by:\n1. Name\n2. ID\nEnter choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            getchar();  // Consume newline
            printf("Enter product name to search: ");
            fgets(name, sizeof(name), stdin);
```

```
            name[strcspn(name, "\n")] = '\0';  // Remove newline character
            searchByName(products, totalProducts, name);
            break;
        case 2:
            printf("Enter product ID to search: ");
            scanf("%d", &id);
            searchByID(products, totalProducts, id);
            break;
        default:
            printf("Invalid choice.\n");
    }

    return 0;
}
```

O/p:

Search products by:

1. Name

2. ID

Enter choice: 1

Enter product name to search: Smartphone

Product found: ID: 2, Name: Smartphone, Price: 800.00, Category: Electronics

Problem 7: Sales Report Generator

Description: Develop a system that generates a sales report for different categories.

Requirements:

- Use a structure for Sale with fields for saleID, productCategory, amount, and date.
- Store sales in an array.
- Use a loop and switch case to categorize and summarize sales.

- Implement functions to add sales data and generate reports.

Output Expectations:

- Display summarized sales data by category.

Sol:

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

#define MAX_SALES 10

// Structure for Sale
typedef struct {
    int saleID;
    char productCategory[50];
    float amount;
    char date[20];
} Sale;

// Function to generate sales report by category
void generateReport(Sale sales[], int totalSales) {
    float electronicsSales = 0, furnitureSales = 0, totalSalesAmount = 0;
    for (int i = 0; i < totalSales; i++) {
        if (strcmp(sales[i].productCategory, "Electronics") == 0) {
            electronicsSales += sales[i].amount;
        } else if (strcmp(sales[i].productCategory, "Furniture") == 0) {
            furnitureSales += sales[i].amount;
        }
        totalSalesAmount += sales[i].amount;
    }

    printf("\nSales Report:\n");
```

```
    printf("Electronics Sales: %.2f\n", electronicsSales);
    printf("Furniture Sales: %.2f\n", furnitureSales);
    printf("Total Sales: %.2f\n", totalSalesAmount);
}

int main() {
    Sale sales[MAX_SALES] = {
        {1, "Electronics", 1500.50, "2025-01-01"},
        {2, "Furniture", 800.00, "2025-01-02"},
        {3, "Electronics", 200.75, "2025-01-03"},
        {4, "Furniture", 1000.30, "2025-01-04"}
    };

    int totalSales = 4;  // Initial number of sales

    generateReport(sales, totalSales);

    return 0;
}
```

O/p:

Sales Report:

Electronics Sales: 1701.25

Furniture Sales: 1800.30

Total Sales: 3501.55

Problem 8: Customer Loyalty Program

Description: Implement a loyalty program that rewards customers based on their total purchase amount.

Requirements:

- Use a structure for Customer with fields for customerID, name, totalPurchases, and rewardPoints.
- Use const for the reward rate.
- Implement functions to calculate and update reward points (call by reference).
- Use a loop to process multiple customers.

Output Expectations:

- Display customer details including reward points after updating.

Sol:

```
#include <stdio.h>

#define REWARD_RATE 0.05  // 5% reward rate

// Structure for Customer
typedef struct {
    int customerID;
    char name[50];
    float totalPurchases;
    float rewardPoints;
} Customer;

// Function to calculate and update reward points (call by reference)
void updateRewardPoints(Customer *customer) {
    customer->rewardPoints = customer->totalPurchases * REWARD_RATE;
}

int main() {
    Customer customer;

    // Input customer details
    printf("Enter Customer ID: ");
```

```
    scanf("%d", &customer.customerID);
    getchar();  // Consume newline character
    printf("Enter Customer Name: ");
    fgets(customer.name, sizeof(customer.name), stdin);
    customer.name[strcspn(customer.name, "\n")] = '\0';  // Remove newline
    printf("Enter Total Purchases: ");
    scanf("%f", &customer.totalPurchases);

    // Update reward points based on total purchases
    updateRewardPoints(&customer);

    // Output the updated customer details
    printf("\nCustomer Details:\n");
    printf("Customer ID: %d\n", customer.customerID);
    printf("Name: %s\n", customer.name);
    printf("Total Purchases: %.2f\n", customer.totalPurchases);
    printf("Reward Points: %.2f\n", customer.rewardPoints);

    return 0;
}
```

O/p: Enter Customer ID: 1
Enter Customer Name: likitha
Enter Total Purchases: 100

Customer Details:
Customer ID: 1
Name: likitha
Total Purchases: 100.00
Reward Points: 5.00

Problem 9: Warehouse Management System

Description: Create a warehouse management system to track stock levels of different products.

Requirements:

- Use a structure for WarehouseItem with fields for itemID, itemName, currentStock, and reorderLevel.
- Use an array to store warehouse items.
- Implement functions to update stock levels (call by reference) and check reorder status (call by value).
- Use a loop for updating stock.

Output Expectations:

- Display the stock levels and reorder status for each item.

Sol:

```
#include <stdio.h>

#define MAX_ITEMS 10

// Structure for Warehouse Item
typedef struct {
    int itemID;
    char itemName[50];
    int currentStock;
    int reorderLevel;
} WarehouseItem;

// Function to update stock levels
void updateStock(WarehouseItem *item) {
    printf("Enter new stock for item %s (ID: %d): ", item->itemName, item->itemID);
    scanf("%d", &item->currentStock);
```

```
}

// Function to check reorder status
void checkReorderStatus(WarehouseItem *item) {
    if (item->currentStock <= item->reorderLevel) {
        printf("Item %s (ID: %d) needs to be reordered.\n", item->itemName, item->itemID);
    } else {
        printf("Item %s (ID: %d) stock level is sufficient.\n", item->itemName, item->itemID);
    }
}

int main() {
    WarehouseItem warehouse[MAX_ITEMS] = {
        {1, "Laptop", 50, 10},
        {2, "Smartphone", 30, 5},
        {3, "Chair", 100, 15},
        {4, "Table", 60, 10}
    };

    int totalItems = 4;  // Initial number of items
    int choice;

    while (1) {
        printf("\nWarehouse Management System\n");
        printf("1. Update Stock\n2. Check Reorder Status\n3. Exit\nEnter choice: ");
        scanf("%d", &choice);

        switch (choice) {
```

```
case 1: {
    int itemID;
    printf("Enter Item ID to update stock: ");
    scanf("%d", &itemID);
    int found = 0;
    for (int i = 0; i < totalItems; i++) {
        if (warehouse[i].itemID == itemID) {
            updateStock(&warehouse[i]);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Item with ID %d not found.\n", itemID);
    }
    break;
}
case 2: {
    int itemID;
    printf("Enter Item ID to check reorder status: ");
    scanf("%d", &itemID);
    int found = 0;
    for (int i = 0; i < totalItems; i++) {
        if (warehouse[i].itemID == itemID) {
            checkReorderStatus(&warehouse[i]);
            found = 1;
            break;
        }
    }
```

```
            if (!found) {
                printf("Item with ID %d not found.\n", itemID);
            }
            break;
        }
        case 3:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice, try again.\n");
    }
  }
}
```

O/p:

Problem 10: Discount Management System

Description: Design a system that manages discounts for different product categories.

Requirements:

- Use a structure for Discount with fields for category, discountPercentage, and validTill.
- Use const for predefined categories.
- Use a switch case to apply discounts based on the category.
- Implement functions to update and display discounts (call by reference).

Output Expectations:

- Show the updated discount details for each category.

Sol:

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

#define ELECTRONICS_DISCOUNT 0.10  // 10% discount for Electronics
```

```
#define FURNITURE_DISCOUNT 0.15    // 15% discount for Furniture
#define CLOTHING_DISCOUNT 0.20     // 20% discount for Clothing

// Structure for Discount
typedef struct {
    char category[50];
    float discountPercentage;
} Discount;

// Function to apply discount based on category
void applyDiscount(Discount *discount) {
    if (strcmp(discount->category, "Electronics") == 0) {
        discount->discountPercentage = ELECTRONICS_DISCOUNT;
    } else if (strcmp(discount->category, "Furniture") == 0) {
        discount->discountPercentage = FURNITURE_DISCOUNT;
    } else if (strcmp(discount->category, "Clothing") == 0) {
        discount->discountPercentage = CLOTHING_DISCOUNT;
    } else {
        printf("No discount available for this category.\n");
        discount->discountPercentage = 0;
    }
}

int main() {
    Discount discount;

    printf("Enter product category (Electronics, Furniture, Clothing): ");
    fgets(discount.category, sizeof(discount.category), stdin);
    discount.category[strcspn(discount.category, "\n")] = '\0';  // Remove newline
```

```
    applyDiscount(&discount);

    printf("\nDiscount Details:\n");
    printf("Category: %s\n", discount.category);
    printf("Discount Percentage: %.2f%%\n", discount.discountPercentage * 100);

    return 0;
}
```

O/p:

Enter product category (Electronics, Furniture, Clothing): Electronics

Discount Details:

Category: Electronics

Discount Percentage: 10.00%

Problem 1: Union for Mixed Data

Description: Create a union that can store an integer, a float, or a character. Write a program that assigns values to each member and displays them.

Sol: #include <stdio.h>

```
// Define a union
union Data {
    int i;
    float f;
    char c;
};

int main() {
    union Data data;  // Declare a union variable
```

```
    // Assign and display integer value
    data.i = 42;
    printf("Integer: %d\n", data.i);

    // Assign and display float value
    data.f = 3.14;
    printf("Float: %.2f\n", data.f);

    // Assign and display character value
    data.c = 'A';
    printf("Character: %c\n", data.c);
      return 0;
}
```

O/p: Integer: 42

Float: 3.14

Character: A

Problem 2: Student Data with Union

Description: Define a union to store either a student's roll number (integer) or name (string). Write a program to input and display student details using the union.

Sol: #include <stdio.h>

```
// Define a union for student data
union Student {
    int rollNumber;
    char name[50];
};
```

```
int main() {
    union Student student;  // Declare a union variable
    int choice;

    printf("Enter 1 to input roll number, 2 to input name: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter roll number: ");
        scanf("%d", &student.rollNumber);
        printf("Student Roll Number: %d\n", student.rollNumber);
    } else if (choice == 2) {
        printf("Enter name: ");
        scanf("%s", student.name);
        printf("Student Name: %s\n", student.name);
    } else {
        printf("Invalid choice.\n");
    }
    return 0;
}
```

O/p:

```
Enter 1 to input roll number, 2 to input name: 1
Enter roll number: 70
Student Roll Number: 70
Enter 1 to input roll number, 2 to input name:
Enter name: likitha
Student Name: likitha
```

Problem 3: Union for Measurement Units

Description: Create a union that can store a distance in either kilometers (float) or miles (float). Write a program to convert and display the distance in both units.

Sol:

```
#include <stdio.h>
// Define a union for distance
union Distance {
    float kilometers;
    float miles;
};
int main() {
    union Distance distance;
    int choice;
    float conversionFactor = 0.621371;  // 1 kilometer = 0.621371 miles

    printf("Enter 1 to input distance in kilometers, 2 for miles: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter distance in kilometers: ");
        scanf("%f", &distance.kilometers);
        distance.miles = distance.kilometers * conversionFactor;
        printf("Distance: %.2f kilometers = %.2f miles\n", distance.kilometers,
distance.miles);
    } else if (choice == 2) {
        printf("Enter distance in miles: ");
        scanf("%f", &distance.miles);
        distance.kilometers = distance.miles / conversionFactor;
        printf("Distance: %.2f miles = %.2f kilometers\n", distance.miles,
distance.kilometers);
    } else {
        printf("Invalid choice.\n");
```

```
    }
    return 0;
}
```

O/p:

Enter 1 to input distance in kilometers, 2 for miles: 1

Enter distance in kilometers: 70

Distance: 43.50 kilometers = 43.50 miles

Enter 1 to input distance in kilometers, 2 for miles: 2

Enter distance in miles: 287

Distance: 461.88 miles = 461.88 kilometers

Problem 4: Union for Shape Dimensions

Description: Define a union to store dimensions of different shapes: a radius (float) for a circle, length and width (float) for a rectangle. Write a program to calculate and display the area based on the selected shape.

Sol: #include <stdio.h>

```
// Define a union for shape dimensions
union ShapeDimensions {
    float radius;
    struct {
        float length;
        float width;
    } rectangle;
};

int main() {
    union ShapeDimensions shape;
    int choice;
    float area;
```

```
    printf("Choose the shape to calculate area:\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("Enter your choice (1 or 2): ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter the radius of the circle: ");
        scanf("%f", &shape.radius);
        area = 3.14159 * shape.radius * shape.radius;
        printf("Area of the circle: %.2f\n", area);
    } else if (choice == 2) {
        printf("Enter the length and width of the rectangle: ");
        scanf("%f %f", &shape.rectangle.length, &shape.rectangle.width);
        area = shape.rectangle.length * shape.rectangle.width;
        printf("Area of the rectangle: %.2f\n", area);
    } else {
        printf("Invalid choice.\n");
    }

    return 0;
}
```

O/p:

```
Choose the shape to calculate area:
1. Circle
2. Rectangle
Enter your choice (1 or 2): 1
Enter the radius of the circle: 35
```

Area of the circle: 3848.45

Choose the shape to calculate area:

1. Circle

2. Rectangle

Enter your choice (1 or 2): 2

Enter the length and width of the rectangle: 12 30

Area of the rectangle: 360.00

Problem 5: Union for Employee Data

Description: Create a union to store either an employee's ID (integer) or salary (float). Write a program to input and display either ID or salary based on user choice.

Sol: #include <stdio.h>

```
// Define a union for employee data
union Employee {
    int id;
    float salary;
};

int main() {
    union Employee employee;
    int choice;

    printf("Enter 1 to input employee ID, 2 to input salary: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter employee ID: ");
        scanf("%d", &employee.id);
        printf("Employee ID: %d\n", employee.id);
    } else if (choice == 2) {
```

```
        printf("Enter employee salary: ");
        scanf("%f", &employee.salary);
        printf("Employee Salary: %.2f\n", employee.salary);
    } else {
        printf("Invalid choice.\n");
    }

    printf("\nNote: In a union, only one member is valid at a time.\n");
    return 0;
}
```

O/p: Enter 1 to input employee ID, 2 to input salary: 1

Enter employee ID: 1001

Employee ID: 1001

Enter 1 to input employee ID, 2 to input salary: 2

Enter employee salary: 25000

Employee Salary: 25000.00

Problem 6: Union for Sensor Data

Description: Define a union to store sensor data, either temperature (float) or pressure (float). Write a program to simulate sensor readings and display the data.

Sol:

```
#include <stdio.h>

// Define a union for sensor data
union SensorData {
    float temperature;
    float pressure;
};

int main() {
    union SensorData sensor;
```

```
    int choice;

    printf("Enter 1 to simulate temperature reading, 2 for pressure reading: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter temperature reading (in Celsius): ");
        scanf("%f", &sensor.temperature);
        printf("Temperature Reading: %.2f°C\n", sensor.temperature);
    } else if (choice == 2) {
        printf("Enter pressure reading (in Pascals): ");
        scanf("%f", &sensor.pressure);
        printf("Pressure Reading: %.2f Pa\n", sensor.pressure);
    } else {
        printf("Invalid choice.\n");
    }
    return 0;
}
```

O/p: Enter 1 to simulate temperature reading, 2 for pressure reading: 1

Enter temperature reading (in Celsius): 34

Temperature Reading: 34.00°C

Enter 1 to simulate temperature reading, 2 for pressure reading: 2

Enter pressure reading (in Pascals): 121

Pressure Reading: 121.00 Pa

Problem 7: Union for Bank Account Information

Description: Create a union to store either a bank account number (integer) or balance (float). Write a program to input and display either the account number or balance based on user input.

Sol: #include <stdio.h>

```
// Define a union for bank account information
union BankAccount {
    int accountNumber;
    float balance;
};

int main() {
    union BankAccount account;
    int choice;

    printf("Enter 1 to input account number, 2 to input balance: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter bank account number: ");
        scanf("%d", &account.accountNumber);
        printf("Bank Account Number: %d\n", account.accountNumber);
    } else if (choice == 2) {
        printf("Enter account balance: ");
        scanf("%f", &account.balance);
        printf("Account Balance: %.2f\n", account.balance);
    } else {
        printf("Invalid choice.\n");
    }

    printf("\nNote: In a union, only one member is valid at a time.\n");
    return 0;
}
```

O/p: Enter 1 to input account number, 2 to input balance: 1

Enter bank account number: 3627138

Bank Account Number: 3627138

Enter 1 to input account number, 2 to input balance: 2

Enter account balance: 70000

Account Balance: 70000.00

Problem 8: Union for Vehicle Information

Description: Define a union to store either the vehicle's registration number (integer) or fuel capacity (float). Write a program to input and display either the registration number or fuel capacity.

Sol: #include <stdio.h>

```
// Define a union for vehicle information
union Vehicle {
    int registrationNumber;
    float fuelCapacity;
};

int main() {
    union Vehicle vehicle;
    int choice;

    printf("Enter 1 to input vehicle registration number, 2 to input fuel capacity: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter vehicle registration number: ");
        scanf("%d", &vehicle.registrationNumber);
        printf("Vehicle Registration Number: %d\n", vehicle.registrationNumber);
    } else if (choice == 2) {
```

```
        printf("Enter fuel capacity (in liters): ");
        scanf("%f", &vehicle.fuelCapacity);
        printf("Fuel Capacity: %.2f liters\n", vehicle.fuelCapacity);
    } else {
        printf("Invalid choice.\n");
    }
    return 0;
}
```

O/p: 1Enter 1 to input vehicle registration number, 2 to input fuel capacity: 1

Enter vehicle registration number: 2627165

Vehicle Registration Number: 2627165

Enter 1 to input vehicle registration number, 2 to input fuel capacity: 2

Enter fuel capacity (in liters): 456

Fuel Capacity: 456.00 liters

Problem 9: Union for Exam Results

Description: Create a union to store either a student's marks (integer) or grade (char). Write a program to input marks or grade and display the corresponding value.

Sol: #include <stdio.h>

```
// Define a union for exam results
union ExamResult {
    int marks;
    char grade;
};

int main() {
    union ExamResult result;
    int choice;
```

```
    printf("Enter 1 to input marks, 2 to input grade: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter marks: ");
        scanf("%d", &result.marks);
        printf("Marks: %d\n", result.marks);
    } else if (choice == 2) {
        printf("Enter grade (A, B, C, etc.): ");
        scanf(" %c", &result.grade);  // Space before %c to consume newline
        printf("Grade: %c\n", result.grade);
    } else {
        printf("Invalid choice.\n");
    }
    return 0;
}
```

O/p: Enter 1 to input marks, 2 to input grade: 1

Enter marks: 97

Marks: 97

Enter 1 to input marks, 2 to input grade: 2

Enter grade (A, B, C, etc.): A

Grade: A

Problem 10: Union for Currency Conversion

Description: Define a union to store currency values in either USD (float) or EUR (float). Write a program to input a value in one currency and display the equivalent in the other.

Sol: #include <stdio.h>

```
// Define a union for currency
union Currency {
    float usd;
    float eur;
};

int main() {
    union Currency currency;
    int choice;
    float conversionRateToEUR = 0.85;  // Example conversion rate
    float conversionRateToUSD = 1.18;  // Example conversion rate

    printf("Enter 1 to input USD, 2 to input EUR: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter amount in USD: ");
        scanf("%f", &currency.usd);
        currency.eur = currency.usd * conversionRateToEUR;
        printf("Equivalent Amount: %.2f USD = %.2f EUR\n", currency.usd,
currency.eur);
    } else if (choice == 2) {
        printf("Enter amount in EUR: ");
        scanf("%f", &currency.eur);
        currency.usd = currency.eur * conversionRateToUSD;
```

```
        printf("Equivalent Amount: %.2f EUR = %.2f USD\n", currency.eur,
currency.usd);
    } else {
        printf("Invalid choice.\n");
    }    return 0;
}
```

O/p:

Enter 1 to input USD, 2 to input EUR: 1

Enter amount in USD: 456

Equivalent Amount: 387.60 USD = 387.60 EUR

Problem 1: Aircraft Fleet Management

Description: Develop a system to manage a fleet of aircraft, tracking their specifications and operational status.

Requirements:

- Define a struct for Aircraft with fields: aircraftID, model, capacity, and status.
- Use an array of Aircraft structures.
- Implement functions to add new aircraft (call by reference), update status, and display fleet details (call by value).
- Use static to track the total number of aircraft.
- Utilize a switch case to manage different operational statuses.
- Employ loops to iterate through the fleet.

Output Expectations:

- Display updated fleet information after each operation.

Sol:
```
#include <stdio.h>
#include <string.h>

#define MAX_FLEET_SIZE 100
```

```
// Aircraft struct definition
typedef struct {
    int aircraftID;
    char model[50];
    int capacity;
    char status[20]; // Operational status like "Available", "In Service", "Under Maintenance"
} Aircraft;

// Static variable to keep track of total number of aircraft
static int totalAircraft = 0;

// Function to add new aircraft
void addAircraft(Aircraft* aircraft, int aircraftID, const char* model, int capacity, const char* status) {
    aircraft->aircraftID = aircraftID;
    strcpy(aircraft->model, model);
    aircraft->capacity = capacity;
    strcpy(aircraft->status, status);
    totalAircraft++;
}

// Function to update aircraft status
void updateStatus(Aircraft* aircraft) {
    int choice;
    printf("\nSelect a new status for Aircraft ID %d:\n", aircraft->aircraftID);
    printf("1. Available\n2. In Service\n3. Under Maintenance\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
```

```
    switch (choice) {
        case 1:
            strcpy(aircraft->status, "Available");
            break;
        case 2:
            strcpy(aircraft->status, "In Service");
            break;
        case 3:
            strcpy(aircraft->status, "Under Maintenance");
            break;
        default:
            printf("Invalid choice.\n");
            break;
    }
}

// Function to display fleet details
void displayFleet(Aircraft fleet[]) {
    printf("\nFleet Details:\n");
    for (int i = 0; i < totalAircraft; i++) {
        printf("Aircraft ID: %d, Model: %s, Capacity: %d, Status: %s\n",
               fleet[i].aircraftID, fleet[i].model, fleet[i].capacity, fleet[i].status);
    }
}

int main() {
    Aircraft fleet[MAX_FLEET_SIZE];

    // Add new aircraft to the fleet
```

```
    addAircraft(&fleet[0], 101, "Boeing 737", 200, "Available");
    addAircraft(&fleet[1], 102, "Airbus A320", 180, "In Service");

    // Display fleet details
    displayFleet(fleet);

    // Update status of the first aircraft
    updateStatus(&fleet[0]);

    // Display updated fleet details
    displayFleet(fleet);

    return 0;
}
```

O/p:

```
Fleet Details:
Aircraft ID: 101, Model: Boeing 737, Capacity: 200, Status: Available
Aircraft ID: 102, Model: Airbus A320, Capacity: 180, Status: In Service

Select a new status for Aircraft ID 101:
1. Available
2. In Service
3. Under Maintenance
Enter your choice: 1

Fleet Details:
Aircraft ID: 101, Model: Boeing 737, Capacity: 200, Status: Available
Aircraft ID: 102, Model: Airbus A320, Capacity: 180, Status: In Service
```

Fleet Details:

Aircraft ID: 101, Model: Boeing 737, Capacity: 200, Status: Available

Aircraft ID: 102, Model: Airbus A320, Capacity: 180, Status: In Service

Select a new status for Aircraft ID 101:

1. Available

2. In Service

3. Under Maintenance

Enter your choice: 3

Fleet Details:

Aircraft ID: 101, Model: Boeing 737, Capacity: 200, Status: Under Maintenance

Aircraft ID: 102, Model: Airbus A320, Capacity: 180, Status: In Service

Problem 2: Satellite Data Processing

Description: Create a system to process and analyze satellite data.

Requirements:

- Define a union for SatelliteData to store either image data (array) or telemetry data (nested structure).
- Use struct to define Telemetry with fields: temperature, velocity, and altitude.
- Implement functions to process image and telemetry data (call by reference).
- Use const for fixed telemetry limits.
- Employ loops to iterate through data points.

Output Expectations:

- Display processed image or telemetry data based on user input.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_IMAGE_SIZE 5

```
#define MAX_TELEMETRY_DATA 3

// Struct for telemetry data
typedef struct {
    float temperature;   // Temperature in Celsius
    float velocity;      // Velocity in km/s
    float altitude;      // Altitude in km
} Telemetry;

// Union to store either image data or telemetry data
typedef union {
    int image[MAX_IMAGE_SIZE];   // Image data array
    Telemetry telemetryData;     // Telemetry data
} SatelliteData;

// Function to process image data
void processImageData(SatelliteData* data) {
    printf("Processing Image Data:\n");
    for (int i = 0; i < MAX_IMAGE_SIZE; i++) {
        printf("Pixel %d: %d\n", i+1, data->image[i]);
    }
}

// Function to process telemetry data
void processTelemetryData(SatelliteData* data) {
    printf("Processing Telemetry Data:\n");
    printf("Temperature: %.2f°C\n", data->telemetryData.temperature);
    printf("Velocity: %.2f km/s\n", data->telemetryData.velocity);
    printf("Altitude: %.2f km\n", data->telemetryData.altitude);
```

```
}

int main() {
    SatelliteData data;
    int choice;

    // Get user choice for type of data to process
    printf("Select the type of satellite data:\n");
    printf("1. Image Data\n2. Telemetry Data\n");
    printf("Enter your choice (1 or 2): ");
    scanf("%d", &choice);

    // Process based on user choice
    if (choice == 1) {
        // Input image data
        for (int i = 0; i < MAX_IMAGE_SIZE; i++) {
            printf("Enter pixel %d value: ", i+1);
            scanf("%d", &data.image[i]);
        }
        processImageData(&data);
    } else if (choice == 2) {
        // Input telemetry data
        printf("Enter temperature (°C): ");
        scanf("%f", &data.telemetryData.temperature);
        printf("Enter velocity (km/s): ");
        scanf("%f", &data.telemetryData.velocity);
        printf("Enter altitude (km): ");
        scanf("%f", &data.telemetryData.altitude);
        processTelemetryData(&data);
```

```
    } else {
        printf("Invalid choice.\n");
    }

    return 0;
}
```

O/p: Select the type of satellite data:

1. Image Data

2. Telemetry Data

Enter your choice (1 or 2): 1

Enter pixel 1 value: 150

Enter pixel 2 value: 300

Enter pixel 3 value: 200

Enter pixel 4 value: 234

Enter pixel 5 value: 120

Processing Image Data:

Pixel 1: 150

Pixel 2: 300

Pixel 3: 200

Pixel 4: 234

Pixel 5: 120

Select the type of satellite data:

1. Image Data

2. Telemetry Data

Enter your choice (1 or 2): 2

Enter temperature (°C): 21

Enter velocity (km/s): 45

Enter altitude (km): 546

Processing Telemetry Data:

Temperature: 21.00°C

Velocity: 45.00 km/s

Altitude: 546.00 km

Problem 3: Mission Control System

Description: Develop a mission control system to manage spacecraft missions.

Requirements:

- Define a struct for Mission with fields: missionID, name, duration, and a nested union for payload (either crew details or cargo).
- Implement functions to add missions (call by reference), update mission details, and display mission summaries (call by value).
- Use static to count total missions.
- Use loops and switch case for managing different mission types.

Output Expectations:

- Provide detailed mission summaries including payload information.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_MISSIONS 5

// Struct for Crew

typedef struct {

    char name[50];

    int age;

    char role[50];

} Crew;

// Struct for Cargo

typedef struct {

    char description[50];

    float weight;

```
} Cargo;

// Union for Payload (Crew or Cargo)
typedef union {
    Crew crew;
    Cargo cargo;
} Payload;

// Struct for Mission
typedef struct {
    int missionID;
    char name[50];
    int duration;
    Payload payload;
    int isCrewMission;  // 1 for crew, 0 for cargo
} Mission;

// Static variable for total missions
static int totalMissions = 0;

// Function to add mission
void addMission(Mission* mission, int missionID, const char* name, int duration,
int isCrewMission) {
    mission->missionID = missionID;
    strcpy(mission->name, name);
    mission->duration = duration;
    mission->isCrewMission = isCrewMission;
    totalMissions++;
}
```

```
// Function to display mission summary
void displayMission(Mission mission) {
    printf("\nMission ID: %d\n", mission.missionID);
    printf("Mission Name: %s\n", mission.name);
    printf("Duration: %d days\n", mission.duration);

    if (mission.isCrewMission) {
        printf("Crew Member: %s, Age: %d, Role: %s\n",
mission.payload.crew.name, mission.payload.crew.age,
mission.payload.crew.role);
    } else {
        printf("Cargo Description: %s, Weight: %.2f kg\n",
mission.payload.cargo.description, mission.payload.cargo.weight);
    }
}

int main() {
    Mission missions[MAX_MISSIONS];

    // Add some missions
    addMission(&missions[0], 1, "Mars Exploration", 180, 1);
    strcpy(missions[0].payload.crew.name, "John Doe");
    missions[0].payload.crew.age = 35;
    strcpy(missions[0].payload.crew.role, "Pilot");

    addMission(&missions[1], 2, "Cargo Delivery", 120, 0);
    strcpy(missions[1].payload.cargo.description, "Satellite");
    missions[1].payload.cargo.weight = 500.0;
```

```
    // Display all missions
    for (int i = 0; i < totalMissions; i++) {
        displayMission(missions[i]);
    }

    return 0;
}
```

O/p:

Mission ID: 1
Mission Name: Mars Exploration
Duration: 180 days
Crew Member: John Doe, Age: 35, Role: Pilot

Mission ID: 2
Mission Name: Cargo Delivery
Duration: 120 days
Cargo Description: Satellite, Weight: 500.00 kg

Problem 4: Aircraft Maintenance Tracker

Description: Create a tracker for aircraft maintenance schedules and logs.

Requirements:

- Use a struct for MaintenanceLog with fields: logID, aircraftID, date, and a nested union for maintenance type (routine or emergency).
- Implement functions to add maintenance logs (call by reference) and display logs (call by value).
- Use const for maintenance frequency.
- Employ loops to iterate through maintenance logs.

Output Expectations:

- Display maintenance logs categorized by type.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_LOGS 10

#define MAINTENANCE_FREQUENCY 30 // Days between routine maintenance

// Structs for different types of maintenance

typedef struct {

    char description[100];

} RoutineMaintenance;

typedef struct {

    char issue[100];

    char action[100];

} EmergencyMaintenance;

// Union to store either Routine or Emergency maintenance

typedef union {

    RoutineMaintenance routine;

    EmergencyMaintenance emergency;

} MaintenanceType;

// Struct for Maintenance Log

typedef struct {

    int logID;

    int aircraftID;

    char date[20];

    MaintenanceType maintenance;

```
    int isEmergency;  // 1 for emergency, 0 for routine
} MaintenanceLog;

// Static variable for total logs
static int totalLogs = 0;

// Function to add a maintenance log
void addMaintenanceLog(MaintenanceLog* log, int logID, int aircraftID, const
char* date, int isEmergency) {
    log->logID = logID;
    log->aircraftID = aircraftID;
    strcpy(log->date, date);
    log->isEmergency = isEmergency;
    totalLogs++;
}

// Function to display a maintenance log
void displayMaintenanceLog(MaintenanceLog log) {
    printf("\nLog ID: %d\n", log.logID);
    printf("Aircraft ID: %d\n", log.aircraftID);
    printf("Date: %s\n", log.date);

    if (log.isEmergency) {
        printf("Maintenance Type: Emergency\n");
        printf("Issue: %s\n", log.maintenance.emergency.issue);
        printf("Action: %s\n", log.maintenance.emergency.action);
    } else {
        printf("Maintenance Type: Routine\n");
        printf("Description: %s\n", log.maintenance.routine.description);
```

```
    }
}

int main() {
    MaintenanceLog logs[MAX_LOGS];

    // Add maintenance logs
    addMaintenanceLog(&logs[0], 1, 101, "2025-01-10", 0);
    strcpy(logs[0].maintenance.routine.description, "Routine inspection and oil
change");

    addMaintenanceLog(&logs[1], 2, 102, "2025-01-12", 1);
    strcpy(logs[1].maintenance.emergency.issue, "Engine failure");
    strcpy(logs[1].maintenance.emergency.action, "Replaced faulty engine");

    // Display all maintenance logs
    for (int i = 0; i < totalLogs; i++) {
        displayMaintenanceLog(logs[i]);
    }

    return 0;
}
```

O/p:

Log ID: 1

Aircraft ID: 101

Date: 2025-01-10

Maintenance Type: Routine

Description: Routine inspection and oil change

Log ID: 2

Aircraft ID: 102

Date: 2025-01-12

Maintenance Type: Emergency

Issue: Engine failure

Action: Replaced faulty engine

Problem 5: Spacecraft Navigation System

Description: Develop a navigation system for spacecraft to track their position and velocity.

Requirements:

- Define a struct for NavigationData with fields: position, velocity, and a nested union for navigation mode (manual or automatic).
- Implement functions to update navigation data (call by reference) and display the current status (call by value).
- Use static to count navigation updates.
- Use loops and switch case for managing navigation modes.

Output Expectations:

- Show updated position and velocity with navigation mode details.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_UPDATES 5

// Struct for manual mode navigation data

typedef struct {

  float x, y, z; // Position coordinates

} ManualNavigation;

// Struct for automatic mode navigation data

```
typedef struct {
    float x, y, z;  // Position coordinates
    float velocity; // Velocity
} AutomaticNavigation;

// Union for storing either manual or automatic navigation data
typedef union {
    ManualNavigation manual;
    AutomaticNavigation automatic;
} NavigationMode;

// Struct for navigation data
typedef struct {
    int updateID;
    NavigationMode navMode;
    char mode[20]; // "Manual" or "Automatic"
} NavigationData;

// Static variable to count navigation updates
static int totalUpdates = 0;

// Function to update navigation data
void updateNavigationData(NavigationData* data, int updateID, const char*
mode, float x, float y, float z, float velocity) {
    data->updateID = updateID;
    strcpy(data->mode, mode);

    if (strcmp(mode, "Manual") == 0) {
        data->navMode.manual.x = x;
```

```
        data->navMode.manual.y = y;
        data->navMode.manual.z = z;
    } else if (strcmp(mode, "Automatic") == 0) {
        data->navMode.automatic.x = x;
        data->navMode.automatic.y = y;
        data->navMode.automatic.z = z;
        data->navMode.automatic.velocity = velocity;
    }

    totalUpdates++;
}

// Function to display navigation data
void displayNavigationData(NavigationData data) {
    printf("\nUpdate ID: %d\n", data.updateID);
    printf("Navigation Mode: %s\n", data.mode);

    if (strcmp(data.mode, "Manual") == 0) {
        printf("Position (Manual Mode): x = %.2f, y = %.2f, z = %.2f\n",
               data.navMode.manual.x, data.navMode.manual.y,
data.navMode.manual.z);
    } else if (strcmp(data.mode, "Automatic") == 0) {
        printf("Position (Automatic Mode): x = %.2f, y = %.2f, z = %.2f, Velocity =
%.2f\n",
               data.navMode.automatic.x, data.navMode.automatic.y,
data.navMode.automatic.z, data.navMode.automatic.velocity);
    }
}

int main() {
```

```
    NavigationData updates[MAX_UPDATES];

    // Update navigation data
    updateNavigationData(&updates[0], 1, "Manual", 10.0, 15.0, 20.0, 0.0);
    updateNavigationData(&updates[1], 2, "Automatic", 12.0, 18.0, 25.0, 30.0);

    // Display all navigation updates
    for (int i = 0; i < totalUpdates; i++) {
        displayNavigationData(updates[i]);
    }

    return 0;
}
```

O/p:

```
Update ID: 1
Navigation Mode: Manual
Position (Manual Mode): x = 10.00, y = 15.00, z = 20.00

Update ID: 2
Navigation Mode: Automatic
Position (Automatic Mode): x = 12.00, y = 18.00, z = 25.00, Velocity = 30.00
```

Problem 6: Flight Simulation Control

Description: Create a control system for flight simulations with different aircraft models.

Requirements:

- Define a struct for Simulation with fields: simulationID, aircraftModel, duration, and a nested union for control settings (manual or automated).

- Implement functions to start simulations (call by reference), update settings, and display simulation results (call by value).
- Use const for fixed simulation parameters.
- Utilize loops to run multiple simulations and a switch case for selecting control settings.

Output Expectations:

- Display simulation results with control settings.

Sol:

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

#define MAX_SIMULATIONS 3

// Struct for manual control settings
typedef struct {
    float throttle;  // Throttle level
    float pitch;     // Pitch angle
    float roll;      // Roll angle
} ManualControl;

// Struct for automated control settings
typedef struct {
    float altitude;  // Altitude
    float speed;     // Speed
} AutomatedControl;

// Union for storing either manual or automated control settings
typedef union {
    ManualControl manual;
    AutomatedControl automated;
} ControlSettings;
```

```
// Struct for simulation data
typedef struct {
    int simulationID;
    char aircraftModel[50];
    int duration;  // Duration of the simulation in minutes
    ControlSettings controlSettings;
    char mode[20]; // "Manual" or "Automated"
} Simulation;

// Function to start a simulation
void startSimulation(Simulation* sim, int simulationID, const char* model, int
duration, const char* mode, float param1, float param2, float param3) {
    sim->simulationID = simulationID;
    strcpy(sim->aircraftModel, model);
    sim->duration = duration;
    strcpy(sim->mode, mode);

    if (strcmp(mode, "Manual") == 0) {
        sim->controlSettings.manual.throttle = param1;
        sim->controlSettings.manual.pitch = param2;
        sim->controlSettings.manual.roll = param3;
    } else if (strcmp(mode, "Automated") == 0) {
        sim->controlSettings.automated.altitude = param1;
        sim->controlSettings.automated.speed = param2;
    }
}

// Function to display simulation results
```

```
void displaySimulationResults(Simulation sim) {
    printf("\nSimulation ID: %d\n", sim.simulationID);
    printf("Aircraft Model: %s\n", sim.aircraftModel);
    printf("Simulation Duration: %d minutes\n", sim.duration);
    printf("Control Mode: %s\n", sim.mode);

    if (strcmp(sim.mode, "Manual") == 0) {
        printf("Manual Control Settings - Throttle: %.2f, Pitch: %.2f, Roll: %.2f\n",
            sim.controlSettings.manual.throttle, sim.controlSettings.manual.pitch,
sim.controlSettings.manual.roll);
    } else if (strcmp(sim.mode, "Automated") == 0) {
        printf("Automated Control Settings - Altitude: %.2f, Speed: %.2f\n",
            sim.controlSettings.automated.altitude,
sim.controlSettings.automated.speed);
    }
}

int main() {
    Simulation simulations[MAX_SIMULATIONS];

    // Start simulations with different control modes
    startSimulation(&simulations[0], 1, "Boeing 747", 30, "Manual", 80.0, 5.0,
10.0);
    startSimulation(&simulations[1], 2, "Airbus A320", 45, "Automated", 30000.0,
550.0, 0.0);
    startSimulation(&simulations[2], 3, "Cessna 172", 60, "Manual", 50.0, 10.0,
20.0);

    // Display simulation results
    for (int i = 0; i < MAX_SIMULATIONS; i++) {
```

```
        displaySimulationResults(simulations[i]);
    }

    return 0;
}
```

Op:

```
Simulation ID: 1
Aircraft Model: Boeing 747
Simulation Duration: 30 minutes
Control Mode: Manual
Manual Control Settings - Throttle: 80.00, Pitch: 5.00, Roll: 10.00

Simulation ID: 2
Aircraft Model: Airbus A320
Simulation Duration: 45 minutes
Control Mode: Automated
Automated Control Settings - Altitude: 30000.00, Speed: 550.00

Simulation ID: 3
Aircraft Model: Cessna 172
Simulation Duration: 60 minutes
Control Mode: Manual
Manual Control Settings - Throttle: 50.00, Pitch: 10.00, Roll: 20.00
```

Problem 7: Aerospace Component Testing

Description: Develop a system for testing different aerospace components.

Requirements:

- Use a struct for ComponentTest with fields: testID, componentName, and a nested union for test data (physical or software).
- Implement functions to record test results (call by reference) and display summaries (call by value).
- Use static to count total tests conducted.
- Employ loops and switch case for managing different test types.

Output Expectations:

- Display test results categorized by component type.

Sol:

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

#define MAX_TESTS 3

// Struct for physical test data
typedef struct {
    float temperature;
    float pressure;
} PhysicalTestData;

// Struct for software test data
typedef struct {
    int errorCode;
    char status[20];
} SoftwareTestData;

// Union for test data (physical or software)
typedef union {
    PhysicalTestData physical;
    SoftwareTestData software;
} TestData;
```

```
// Struct for component test
typedef struct {
    int testID;
    char componentName[50];
    TestData data;
    char testType[20];
} ComponentTest;

// Static variable to track total tests
static int totalTests = 0;

// Function to record test results
void recordTestResult(ComponentTest *test, int testID, const char *component,
const char *type, float param1, float param2, const char *status) {
    test->testID = testID;
    strcpy(test->componentName, component);
    strcpy(test->testType, type);

    if (strcmp(type, "Physical") == 0) {
        test->data.physical.temperature = param1;
        test->data.physical.pressure = param2;
    } else if (strcmp(type, "Software") == 0) {
        test->data.software.errorCode = (int)param1;
        strcpy(test->data.software.status, status);
    }

    totalTests++;
}
```

```
// Function to display test result
void displayTestResult(const ComponentTest test) {
    printf("\nTest ID: %d\n", test.testID);
    printf("Component: %s\n", test.componentName);
    printf("Test Type: %s\n", test.testType);

    if (strcmp(test.testType, "Physical") == 0) {
        printf("Physical Test - Temperature: %.2f, Pressure: %.2f\n",
test.data.physical.temperature, test.data.physical.pressure);
    } else if (strcmp(test.testType, "Software") == 0) {
        printf("Software Test - Error Code: %d, Status: %s\n",
test.data.software.errorCode, test.data.software.status);
    }
}

int main() {
    ComponentTest tests[MAX_TESTS];

    // Record some tests
    recordTestResult(&tests[0], 1, "Engine", "Physical", 100.5, 15.2, NULL);
    recordTestResult(&tests[1], 2, "Flight Software", "Software", 0, 0, "Passed");
    recordTestResult(&tests[2], 3, "Wing Structure", "Physical", 80.0, 12.0, NULL);

    // Display all test results
    for (int i = 0; i < totalTests; i++) {
        displayTestResult(tests[i]);
    }

    return 0;
```

}

O/p:

Test ID: 1

Component: Engine

Test Type: Physical

Physical Test - Temperature: 100.50, Pressure: 15.20

Test ID: 2

Component: Flight Software

Test Type: Software

Software Test - Error Code: 0, Status: Passed

Test ID: 3

Component: Wing Structure

Test Type: Physical

Physical Test - Temperature: 80.00, Pressure: 12.00

Problem 8: Space Station Crew Management

Description: Create a system to manage crew members aboard a space station.

Requirements:

- Define a struct for CrewMember with fields: crewID, name, role, and a nested union for role-specific details (engineer or scientist).
- Implement functions to add crew members (call by reference), update details, and display crew lists (call by value).
- Use const for fixed role limits.
- Use loops to iterate through the crew list and a switch case for role management.

Output Expectations:

- Show updated crew information including role-specific details.

Sol: #include <stdio.h>
#include <string.h>

#define MAX_CREW 3

// Struct for engineer role-specific details
typedef struct {
    char specialty[50];
    int yearsExperience;
} Engineer;

// Struct for scientist role-specific details
typedef struct {
    char field[50];
    int publications;
} Scientist;

// Union for role-specific details
typedef union {
    Engineer engineer;
    Scientist scientist;
} RoleDetails;

// Struct for crew member data
typedef struct {
    int crewID;
    char name[50];
    char role[20];
    RoleDetails roleDetails;

```
} CrewMember;

// Function to add crew member details
void addCrewMember(CrewMember* member, int crewID, const char* name,
const char* role, const char* specialty_or_field, int years_or_publications) {
    member->crewID = crewID;
    strcpy(member->name, name);
    strcpy(member->role, role);

    if (strcmp(role, "Engineer") == 0) {
        strcpy(member->roleDetails.engineer.specialty, specialty_or_field);
        member->roleDetails.engineer.yearsExperience = years_or_publications;
    } else if (strcmp(role, "Scientist") == 0) {
        strcpy(member->roleDetails.scientist.field, specialty_or_field);
        member->roleDetails.scientist.publications = years_or_publications;
    }
}

// Function to display crew member details
void displayCrewMember(CrewMember member) {
    printf("\nCrew ID: %d\n", member.crewID);
    printf("Name: %s\n", member.name);
    printf("Role: %s\n", member.role);

    if (strcmp(member.role, "Engineer") == 0) {
        printf("Specialty: %s\n", member.roleDetails.engineer.specialty);
        printf("Years of Experience: %d\n",
member.roleDetails.engineer.yearsExperience);
    } else if (strcmp(member.role, "Scientist") == 0) {
        printf("Field: %s\n", member.roleDetails.scientist.field);
```

```
        printf("Publications: %d\n", member.roleDetails.scientist.publications);
    }
}

int main() {
    CrewMember crew[MAX_CREW];

    // Add crew members
    addCrewMember(&crew[0], 1, "Alice", "Engineer", "Propulsion", 10);
    addCrewMember(&crew[1], 2, "Bob", "Scientist", "Astrophysics", 5);
    addCrewMember(&crew[2], 3, "Charlie", "Engineer", "Robotics", 8);

    // Display crew member details
    for (int i = 0; i < MAX_CREW; i++) {
        displayCrewMember(crew[i]);
    }

    return 0;
}
```

O/p:

```
Crew ID: 1
Name: Alice
Role: Engineer
Specialty: Propulsion
Years of Experience: 10

Crew ID: 2
Name: Bob
```

Role: Scientist

Field: Astrophysics

Publications: 5

Crew ID: 3

Name: Charlie

Role: Engineer

Specialty: Robotics

Years of Experience: 8

Problem 9: Aerospace Research Data Analysis

Description: Develop a system to analyze research data from aerospace experiments.

Requirements:

- Use a struct for ResearchData with fields: experimentID, description, and a nested union for data type (numerical or qualitative).
- Implement functions to analyze data (call by reference) and generate reports (call by value).
- Use static to track the number of analyses conducted.
- Employ loops and switch case for managing different data types.

Output Expectations:

- Provide detailed reports of analyzed data.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_ANALYSES 3

// Struct for numerical data

typedef struct {

  float value1;

```
    float value2;
} NumericalData;

// Struct for qualitative data
typedef struct {
    char observation[100];
} QualitativeData;

// Union for data type (numerical or qualitative)
typedef union {
    NumericalData numerical;
    QualitativeData qualitative;
} DataType;

// Struct for research data
typedef struct {
    int experimentID;
    char description[100];
    DataType data;
    char dataType[20];
} ResearchData;

// Static variable to track the number of analyses conducted
static int totalAnalyses = 0;

// Function to analyze research data
void analyzeData(ResearchData* data, int experimentID, const char* description,
const char* type, float value1, float value2, const char* observation) {
    data->experimentID = experimentID;
```

```
    strcpy(data->description, description);
    strcpy(data->dataType, type);

    if (strcmp(type, "Numerical") == 0) {
        data->data.numerical.value1 = value1;
        data->data.numerical.value2 = value2;
    } else if (strcmp(type, "Qualitative") == 0) {
        strcpy(data->data.qualitative.observation, observation);
    }

    totalAnalyses++;
}

// Function to generate a report of the analyzed data
void generateReport(const ResearchData data) {
    printf("\nExperiment ID: %d\n", data.experimentID);
    printf("Description: %s\n", data.description);
    printf("Data Type: %s\n", data.dataType);

    if (strcmp(data.dataType, "Numerical") == 0) {
        printf("Numerical Data - Value 1: %.2f, Value 2: %.2f\n",
data.data.numerical.value1, data.data.numerical.value2);
    } else if (strcmp(data.dataType, "Qualitative") == 0) {
        printf("Qualitative Data - Observation: %s\n",
data.data.qualitative.observation);
    }
}

int main() {
    ResearchData analyses[MAX_ANALYSES];
```

```
    // Analyze some research data
    analyzeData(&analyses[0], 101, "Pressure Test", "Numerical", 120.5, 35.0, NULL);
    analyzeData(&analyses[1], 102, "Material Strength Test", "Qualitative", 0, 0, "Material showed excellent strength under stress.");
    analyzeData(&analyses[2], 103, "Temperature Test", "Numerical", 100.0, 25.5, NULL);

    // Generate reports for all analyses
    for (int i = 0; i < totalAnalyses; i++) {
        generateReport(analyses[i]);
    }

    return 0;
}
```

O/p:

```
Experiment ID: 101
Description: Pressure Test
Data Type: Numerical
Numerical Data - Value 1: 120.50, Value 2: 35.00

Experiment ID: 102
Description: Material Strength Test
Data Type: Qualitative
Qualitative Data - Observation: Material showed excellent strength under stress.

Experiment ID: 103
Description: Temperature Test
```

Data Type: Numerical

Numerical Data - Value 1: 100.00, Value 2: 25.50

Problem 10: Rocket Launch Scheduler

Description: Create a scheduler for managing rocket launches.

Requirements:

- Define a struct for Launch with fields: launchID, rocketName, date, and a nested union for launch status (scheduled or completed).
- Implement functions to schedule launches (call by reference), update statuses, and display launch schedules (call by value).
- Use const for fixed launch parameters.
- Use loops to iterate through launch schedules and a switch case for managing status updates.

Output Expectations:

- Display detailed launch schedules and statuses.

Sol: #include <stdio.h>

#include <string.h>

#define MAX_LAUNCHES 5

// Struct for launch date

typedef struct {

  int year, month, day; // Launch date

} LaunchDate;

// Union for launch status (either scheduled or completed)

typedef union {

  char scheduled[20]; // "Scheduled"

  char completed[20]; // "Completed"

} LaunchStatus;

```
// Struct for launch details
typedef struct {
    int launchID;
    char rocketName[50];
    LaunchDate date;
    LaunchStatus status;  // Union to hold launch status
    char statusType[20];  // "Scheduled" or "Completed"
} Launch;

// Function to schedule a launch
void scheduleLaunch(Launch* launch, int launchID, const char* rocketName, int
year, int month, int day, const char* status) {
    launch->launchID = launchID;
    strcpy(launch->rocketName, rocketName);
    launch->date.year = year;
    launch->date.month = month;
    launch->date.day = day;
    strcpy(launch->statusType, status);

    if (strcmp(status, "Scheduled") == 0) {
        strcpy(launch->status.scheduled, "Scheduled");
    } else if (strcmp(status, "Completed") == 0) {
        strcpy(launch->status.completed, "Completed");
    }
}

// Function to update the launch status
void updateLaunchStatus(Launch* launch, const char* status) {
```

```
    strcpy(launch->statusType, status);
    if (strcmp(status, "Scheduled") == 0) {
        strcpy(launch->status.scheduled, "Scheduled");
    } else if (strcmp(status, "Completed") == 0) {
        strcpy(launch->status.completed, "Completed");
    }
}

// Function to display launch schedule
void displayLaunchSchedule(Launch launch) {
    printf("\nLaunch ID: %d\n", launch.launchID);
    printf("Rocket Name: %s\n", launch.rocketName);
    printf("Launch Date: %d-%d-%d\n", launch.date.year, launch.date.month,
launch.date.day);
    printf("Launch Status: %s\n", launch.statusType);
}

int main() {
    Launch launches[MAX_LAUNCHES];

    // Schedule rocket launches
    scheduleLaunch(&launches[0], 1, "Falcon 9", 2025, 5, 20, "Scheduled");
    scheduleLaunch(&launches[1], 2, "Starship", 2025, 6, 15, "Scheduled");
    scheduleLaunch(&launches[2], 3, "Atlas V", 2025, 7, 10, "Scheduled");

    // Display all scheduled launches
    for (int i = 0; i < 3; i++) {
        displayLaunchSchedule(launches[i]);
    }
```

```
    // Update the status of the first launch to "Completed"
    updateLaunchStatus(&launches[0], "Completed");

    // Display updated launch schedules
    printf("\nUpdated Launch Schedules:\n");
    for (int i = 0; i < 3; i++) {
        displayLaunchSchedule(launches[i]);
    }

    return 0;
}
```

O/p:

```
Launch ID: 1
Rocket Name: Falcon 9
Launch Date: 2025-5-20
Launch Status: Scheduled

Launch ID: 2
Rocket Name: Starship
Launch Date: 2025-6-15
Launch Status: Scheduled

Launch ID: 3
Rocket Name: Atlas V
Launch Date: 2025-7-10
Launch Status: Scheduled

Updated Launch Schedules:
```

Launch ID: 1
Rocket Name: Falcon 9
Launch Date: 2025-5-20
Launch Status: Completed

Launch ID: 2
Rocket Name: Starship
Launch Date: 2025-6-15
Launch Status: Scheduled

Launch ID: 3
Rocket Name: Atlas V
Launch Date: 2025-7-10
Launch Status: Scheduled