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.

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

typedef struct
{
   int itemID;
   char itemName[50];
   float price;
   int quantity;
} Item;
```

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

static int totalItemsAdded = 0;

int main()
{
   Item inventory[100];
   int itemCount = 0;
   int option;
   do
   {
      printf("\nInventory Management System\n");
      printf("1. Add item\n");
      printf("2. Update item\n");
      printf("3. Display inventory\n");
      printf("4. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: addItem(inventory, &itemCount);
                 break;
         case 2: int itemID, found = 0;
                 printf("Enter item ID to update: ");
```

```
                scanf("%d", &itemID);
                for (int i = 0; i < itemCount; i++)
                {
                    if (inventory[i].itemID == itemID)
                    {
                        updateItem(&inventory[i]);
                        found = 1;
                        break;
                    }
                }
                if (!found)
                    printf("Item with ID %d not found\n", itemID);
                break;
            case 3: displayInventory(inventory, itemCount);
                break;
            case 4: printf("Exit the program\n");
                break;
            default:printf("Invalid option\n");
        }
    } while (option != 4);
    return 0;
}

// Function to add a new item to the inventory
void addItem(Item inventory[], int *itemCount)
{
    if (*itemCount >= 100)
```

```
    {
        printf("Inventory is full\n");
        return;
    }

    Item *newItem = &inventory[*itemCount];
    printf("Enter item ID: ");
    scanf("%d", &newItem->itemID);

    printf("Enter item name: ");
    scanf(" %[^\n]", newItem->itemName);

    printf("Enter item price: ");
    scanf("%f", &newItem->price);

    printf("Enter item quantity: ");
    scanf("%d", &newItem->quantity);

    (*itemCount)++;
    totalItemsAdded++;

    printf("Item added successfully\n");
    displayInventory(inventory, *itemCount);
    printf("Total items added: %d\n", totalItemsAdded);
}

// Function to update details of an item (call by reference)
```

```
void updateItem(Item *item)
{
    printf("Update details for item ID: %d\n", item->itemID);

    printf("Enter new item name: ");
    scanf(" %[^\n]", item->itemName);

    printf("Enter new item price: ");
    scanf("%f", &item->price);

    printf("Enter new item quantity: ");
    scanf("%d", &item->quantity);

    printf("Item updated successfully\n");
}

// Function to display the inventory (call by value)
void displayInventory(Item inventory[], int itemCount)
{
    printf("\nCurrent inventory:\n");
    printf("ID\tName\t\tPrice\t\tQuantity\n");

    for (int i = 0; i < itemCount; i++)
    {
        printf("%d\t%s\t\t%.2f\t\t%d\n", inventory[i].itemID,
               inventory[i].itemName, inventory[i].price,
               inventory[i].quantity);
```

```
    }
}
```

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.

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

typedef struct
{
    int orderID;
    char customerName[50];
    int itemCount;
    float itemPrices[10];
    float totalCost;
} Order;
```

```
// Function prototypes
float calculateTotalCost(Order order);
void applyDiscount(Order *order);
void processOrders(Order orders[], int orderCount);

int main()
{
    Order orders[50];
    int orderCount;

    printf("Enter the number of orders to process: ");
    scanf("%d", &orderCount);

    for (int i = 0; i < orderCount; i++)
    {
        printf("\nEnter details for order %d\n", i + 1);

        printf("Enter order ID: ");
        scanf("%d", &orders[i].orderID);

        printf("Enter customer name: ");
        scanf(" %[^\n]", orders[i].customerName);

        printf("Enter number of items: ");
        scanf("%d", &orders[i].itemCount);

        for (int j = 0; j < orders[i].itemCount; j++)
```

```
        {
            printf("Enter price of item %d: ", j + 1);
            scanf("%f", &orders[i].itemPrices[j]);
        }

        // Call the function to calculate total cost
        orders[i].totalCost = calculateTotalCost(orders[i]);

        // Call the function to apply discount
        applyDiscount(&orders[i]);
    }

    printf("\nProcessed orders\n");
    printf("OrderID\tCustomer\tTotal cost(Before dis.)\tTotal cost(After dis.)\n");

    for (int i = 0; i < orderCount; i++)
    {
        float totalBeforeDiscount = 0.0;
        for (int j = 0; j < orders[i].itemCount; j++)
            totalBeforeDiscount += orders[i].itemPrices[j];

        printf("%d\t%s\t\t%.2f\t\t\t%.2f\n", orders[i].orderID,
               orders[i].customerName, totalBeforeDiscount,
               orders[i].totalCost);
    }
    return 0;
}
```

```
// Function to calculate total cost
float calculateTotalCost(Order order)
{
    float total = 0.0;
    for (int i = 0; i < order.itemCount; i++)
        total += order.itemPrices[i];
    return total;
}

// Function to apply discount
void applyDiscount(Order *order)
{
    order->totalCost -= order->totalCost * 0.2; // 20% discount
}
```

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.

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

typedef struct
{
    int customerID;
    char feedbackText[300];
    int rating;
} Feedback;

// Function prototypes
void addFeedback(Feedback feedbacks[], int *feedbackCount);
void displayFeedbackSummary(Feedback feedbacks[], int feedbackCount);

int main() {
    Feedback feedbacks[100];
    int feedbackCount = 0;
    int option;
    do
    {
        printf("\nCustomer Feedback System\n");
        printf("1. Add feedback\n");
        printf("2. Display feedback summary\n");
        printf("3. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);
```

```
        switch (option)
        {
            case 1: addFeedback(feedbacks, &feedbackCount);
                    break;
            case 2: displayFeedbackSummary(feedbacks, feedbackCount);
                    break;
            case 3: printf("Exit the program\n");
                    break;
            default:printf("Invalid option\n");
        }
    } while (option != 3);
    return 0;
}

// Function to add feedback
void addFeedback(Feedback feedbacks[], int *feedbackCount)
{
    if (*feedbackCount >= 100)
    {
        printf("Feedback storage is full\n");
        return;
    }

    Feedback *newfeedback = &feedbacks[*feedbackCount];
    printf("\nEnter customer ID: ");
    scanf("%d", &newfeedback->customerID);
```

```
    printf("Enter Feedback Text: ");
    scanf(" %[^\n]", newfeedback->feedbackText);
    do
    {
        printf("Enter rating (1 to 4): ");
        scanf("%d", &newfeedback->rating);

        if (newfeedback->rating < 1 || newfeedback->rating > 5)
            printf("Invalid rating\n");
    } while (newfeedback->rating < 1 || newfeedback->rating > 5);

    (*feedbackCount)++;
    printf("Feedback added successfully\n");
}

// Function to display feedback summary
void displayFeedbackSummary(Feedback feedbacks[], int feedbackCount)
{
    printf("\nFeedback summary\n");
    for (int i = 0; i < feedbackCount; i++)
    {
        printf("\nCustomer ID: %d\n", feedbacks[i].customerID);
        printf("Feedback: %s\n", feedbacks[i].feedbackText);

        printf("Rating: %d - ", feedbacks[i].rating);
        switch (feedbacks[i].rating)
        {
```

```
            case 4: printf("Excellent\n");
                    break;
            case 3: printf("Good\n");
                    break;
            case 2: printf("Average\n");
                    break;
            case 1: printf("Poor\n");
                    break;
            default:printf("Unknown category\n");
        }
    }
}
```

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.

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

```
#include <string.h>

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

// Function prototype
void processAndUpdatePayment(Payment *payment);

int main()
{
   Payment payment;
   char option;
   do
   {
      printf("\nPayment Processing System\n");
      printf("Enter payment method (Card, UPI, Wallet): ");
      scanf("%s", payment.method);

      printf("Enter payment amount: ");
      scanf("%f", &payment.amount);

      // Call the function to process and update the payment
      processAndUpdatePayment(&payment);
```

```
        printf("\nPayment Details:\n");
        printf("Payment method: %s\n", payment.method);
        printf("Payment amount: %.2f\n", payment.amount);
        printf("Transaction charge: %.2f\n", payment.transactionCharge);
        printf("Total amount (Including charges): %.2f\n", payment.amount + payment.transactionCharge);
        printf("\nDo you want to process another payment? (y/n): ");
        scanf(" %c", &option);
    } while (option == 'y' || option == 'Y');
    printf("Exit the system\n");
    return 0;
}

// Function to process and update payment
void processAndUpdatePayment(Payment *payment)
{
    if (strcmp(payment->method, "Card") == 0)
        payment->transactionCharge = payment->amount * 0.03; // 3% for credit/debit cards
    else if (strcmp(payment->method, "UPI") == 0)
        payment->transactionCharge = payment->amount * 0.02;   // 2% for UPI
    else if (strcmp(payment->method, "Wallet") == 0)
        payment->transactionCharge = payment->amount * 0.01; // 1% for wallets
    else
    {
        printf("Invalid payment method\n");
```

```
        payment->transactionCharge = 0;
    }
}
```

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.

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

typedef struct
{
    int itemID;
    char itemName[50];
    float price;
    int quantity;
} CartItem;
```

```
// Function prototypes
void addItem(CartItem cart[], int *cartCount);
void removeItem(CartItem cart[], int *cartCount);
void displayCart(CartItem cart[], int cartCount);

int main()
{
   CartItem cart[50];
   int cartCount = 0;
   int option;
   do
   {
      printf("\nShopping Cart System\n");
      printf("1. Add item\n");
      printf("2. Remove item\n");
      printf("3. View cart\n");
      printf("4. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: addItem(cart, &cartCount);
                 break;
         case 2: removeItem(cart, &cartCount);
                 break;
         case 3: displayCart(cart, cartCount);
                 break;
```

```
            case 4: printf("Exit the system\n");
                    break;
            default:printf("Invalid option\n");
        }
    } while (option != 4);
    return 0;
}

// Function to add an item to the cart
void addItem(CartItem cart[], int *cartCount)
{
    if (*cartCount >= 50)
    {
        printf("Cart is full\n");
        return;
    }
    CartItem *newItem = &cart[*cartCount];
    printf("\nEnter item ID: ");
    scanf("%d", &newItem->itemID);

    printf("Enter item name: ");
    scanf(" %[^\n]", newItem->itemName);

    printf("Enter item price: ");
    scanf("%f", &newItem->price);

    printf("Enter quantity: ");
```

```
    scanf("%d", &newItem->quantity);

    (*cartCount)++;
    printf("Item added successfully\n");
    displayCart(cart, *cartCount);
}

// Function to remove an item from the cart
void removeItem(CartItem cart[], int *cartCount)
{
    if (*cartCount == 0)
    {
        printf("Cart is empty\n");
        return;
    }
    int itemID, found = 0;
    printf("\nEnter item ID to remove: ");
    scanf("%d", &itemID);

    for (int i = 0; i < *cartCount; i++)
    {
        if (cart[i].itemID == itemID)
        {
            found = 1;
            // Shift remaining items to the left
            for (int j = i; j < *cartCount - 1; j++)
                cart[j] = cart[j + 1];
```

```
            (*cartCount)--;
            printf("Item removed successfully\n");
            break;
        }
    }
    if (!found)
        printf("Item with ID %d not found in the cart\n", itemID);
    displayCart(cart, *cartCount);
}

// Function to display the cart
void displayCart(CartItem cart[], int cartCount)
{
    if (cartCount == 0)
    {
        printf("\nCart is empty\n");
        return;
    }
    printf("\nCurrent cart:\n");
    printf("ID\tName\t\tPrice\tQuantity\tTotal\n");
    for (int i = 0; i < cartCount; i++)
    {
        float total = cart[i].price * cart[i].quantity;
        printf("%d\t%s\t\t%.2f\t%d\t\t%.2f\n", cart[i].itemID, cart[i].itemName,
               cart[i].price, cart[i].quantity, total);
    }
}
```

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.

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

typedef struct
{
    int productID;
    char productName[50];
    char category[30];
    float price;
} Product;

// Function prototypes
void searchByName(Product products[], int productCount);
void updateProduct(Product products[], int productCount);
```

```
int main()
{
   Product products[50];
   int productCount = 0;
   int option;

   products[0] = (Product){101, "Laptop", "Electronics", 800.49};
   products[1] = (Product){102, "Smartphone", "Electronics", 699.99};
   products[2] = (Product){103, "Desk Chair", "Furniture", 250.99};
   productCount = 3;
   do
   {
      printf("\nProduct Search System\n");
      printf("1. Search product by name\n");
      printf("2. Update product details\n");
      printf("3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: searchByName(products, productCount);
                 break;
         case 2: updateProduct(&products[0], productCount);
                 break;
         case 3: printf("Exit the system\n");
                 break;
         default:printf("Invalid option\n");
```

```
        }
    } while (option != 4);
    return 0;
}

// Function to search for a product by name
void searchByName(Product products[], int productCount)
{
    char searchName[50];
    int found = 0;
    printf("\nEnter product name to search: ");
    scanf(" %[^\n]", searchName);
    for (int i = 0; i < productCount; i++)
    {
        if (strcmp(products[i].productName, searchName) == 0)
        {
            found = 1;
            printf("\nProduct found:\n");
            printf("ID: %d\nName: %s\nCategory: %s\nPrice: %.2f\n",
                   products[i].productID, products[i].productName,
                   products[i].category, products[i].price);
            break;
        }
    }
    if (!found)
        printf("\nProduct with name '%s' not found\n", searchName);
}
```

```
// Function to update product details
void updateProduct(Product products[], int productCount)
{
    int searchID, found = 0;
    printf("\nEnter product ID to update: ");
    scanf("%d", &searchID);
    for (int i = 0; i < productCount; i++)
    {
        if (products[i].productID == searchID)
        {
            found = 1;
            printf("\nProduct found:\n");
            printf("ID: %d\nName: %s\nCategory: %s\nPrice: %.2f\n",
                   products[i].productID, products[i].productName,
                   products[i].category, products[i].price);

            printf("\nEnter new name: ");
            scanf(" %[^\n]", products[i].productName);
            printf("Enter new category: ");
            scanf(" %[^\n]", products[i].category);
            printf("Enter new price: ");
            scanf("%f", &products[i].price);
            printf("\nProduct updated successfully\n");
            break;
        }
    }
    if (!found)
```

```
        printf("\nProduct with ID %d not found\n", searchID);
    }
```

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.

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

typedef struct
{
   int saleID;
   char productCategory[30];
   float amount;
   char date[15];
} Sale;

// Function prototypes
```

```
void addSale(Sale sales[], int *saleCount);
void generateReport(Sale sales[], int saleCount);

int main()
{
   Sale sales[100];
   int saleCount = 0;
   int option;
   do
   {
      printf("\nSales Report Generator\n");
      printf("1. Add sale\n");
      printf("2. Generate sales report\n");
      printf("3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: addSale(sales, &saleCount);
                 break;
         case 2: generateReport(sales, saleCount);
                 break;
         case 3: printf("Exit the system\n");
                 break;
         default:printf("Invalid option\n");
      }
   } while (option != 3);
```

```
    return 0;
}

// Function to add a sale
void addSale(Sale sales[], int *saleCount)
{
    if (*saleCount >= 100)
    {
        printf("Maximum capacity reached\n");
        return;
    }
    Sale *newSale = &sales[*saleCount];
    printf("\nEnter sale ID: ");
    scanf("%d", &newSale->saleID);

    printf("Enter product category: ");
    scanf(" %[^\n]", newSale->productCategory);

    printf("Enter sale amount: ");
    scanf("%f", &newSale->amount);

    printf("Enter sale date (YYYY-MM-DD): ");
    scanf(" %[^\n]", &newSale->date);

    (*saleCount)++;
    printf("Sale added successfully\n");
}
```

```
// Function to generate a sales report
void generateReport(Sale sales[], int saleCount)
{
    if (saleCount == 0)
    {
        printf("\nNo sales data available\n");
        return;
    }
     float electronicsTotal = 0.0, furnitureTotal = 0.0, groceriesTotal = 0.0, othersTotal = 0.0;
    printf("\nGenerating Sales Report\n");
    for (int i = 0; i < saleCount; i++)
    {
        if (strcmp(sales[i].productCategory, "Electronics") == 0)
            electronicsTotal += sales[i].amount;
        else if (strcmp(sales[i].productCategory, "Furniture") == 0)
            furnitureTotal += sales[i].amount;
        else if (strcmp(sales[i].productCategory, "Groceries") == 0)
            groceriesTotal += sales[i].amount;
        else
            othersTotal += sales[i].amount;
    }
    printf("\nSales report by category\n");
    printf("Electronics: %.2f\n", electronicsTotal);
    printf("Furniture: %.2f\n", furnitureTotal);
    printf("Groceries: %.2f\n", groceriesTotal);
    printf("Others: %.2f\n", othersTotal);
}
```

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.

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

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

// Function prototypes
void addCustomer(Customer customers[], int *customerCount);
void updateRewards(Customer customers[], int customerCount);
void displayCustomers(Customer customers[], int customerCount);
```

```
int main()
{
    Customer customers[50];
    int customerCount = 0;
    int option;
    do
    {
        printf("\nCustomer Loyalty Program\n");
        printf("1. Add customer\n");
        printf("2. Update reward points\n");
        printf("3. Display customers\n");
        printf("4. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);
        switch (option)
        {
            case 1: addCustomer(customers, &customerCount);
                    break;
            case 2: updateRewards(&customers[0], customerCount);
                    break;
            case 3: displayCustomers(customers, customerCount);
                    break;
            case 4: printf("Exit the program.\n");
                    break;
            default: printf("Invalid option\n");
        }
    } while (option != 4);
```

```
    return 0;
}

// Function to add a new customer
void addCustomer(Customer customers[], int *customerCount)
{
    if (*customerCount >= 50)
    {
        printf("Maximum capacity reached\n");
        return;
    }
    printf("\nEnter customer ID: ");
    scanf("%d", &customers[*customerCount].customerID);

    printf("Enter customer name: ");
    scanf(" %[^\n]", customers[*customerCount].name);

    printf("Enter total purchases: ");
    scanf("%f", &customers[*customerCount].totalPurchases);

    customers[*customerCount].rewardPoints = 0;
    (*customerCount)++;
    printf("Customer added successfully\n");
}

// Function to update reward points
void updateRewards(Customer customers[], int customerCount)
```

```
{
    if (customerCount == 0)
    {
        printf("\nNo customers available\n");
        return;
    }
    for (int i = 0; i < customerCount; i++)
        customers[i].rewardPoints = (int)(customers[i].totalPurchases * 0.1); // 1
reward point for every 10 spent
    printf("\nReward points updated successfully\n");
}

// Function to display all customers
void displayCustomers(Customer customers[], int customerCount)
{
    if (customerCount == 0)
    {
        printf("\nNo customer data available\n");
        return;
    }
    printf("\nCustomer details:\n");
    printf("ID\tName\tTotal purchases\tReward points\n");
    for (int i = 0; i < customerCount; i++) {
        printf("%d\t%s\t%.2f\t\t%d\n", customers[i].customerID,
                customers[i].name, customers[i].totalPurchases,
                customers[i].rewardPoints);
    }
}
```

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.

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

typedef struct
{
   int itemID;
   char itemName[50];
   int currentStock;
   int reorderLevel;
} WarehouseItem;

// Function prototypes
void updateStock(WarehouseItem items[], int *itemCount);
void checkReorderStatus(WarehouseItem items[], int itemCount);
void displayStock(WarehouseItem items[], int itemCount);
```

```
int main()
{
    WarehouseItem items[50];
    int itemCount = 0;
    int option;
    do
    {
        printf("\nWarehouse Management System\n");
        printf("1. Add/Update stock\n");
        printf("2. Check reorder status\n");
        printf("3. Display all stock\n");
        printf("4. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);
        switch (option)
        {
            case 1: updateStock(items, &itemCount);
                    break;
            case 2: checkReorderStatus(items, itemCount);
                    break;
            case 3: displayStock(items, itemCount);
                    break;
            case 4: printf("Exit the program\n");
                    break;
            default:printf("Invalid option\n");
        }
    } while (option != 4);
```

```
    return 0;
}

// Function to update stock
void updateStock(WarehouseItem items[], int *itemCount)
{
    if (*itemCount >= 50)
    {
        printf("Maximum capacity reached\n");
        return;
    }
    int itemID, stockChange;
    printf("\nEnter the item ID to update stock: ");
    scanf("%d", &itemID);
    int found = 0;
    for (int i = 0; i < *itemCount; i++)
    {
        if (items[i].itemID == itemID)
        {
            found = 1;
            printf("Enter the change in stock (+ for addition, - for removal): ");
            scanf("%d", &stockChange);
            items[i].currentStock += stockChange;
                printf("Stock updated for item: %s (Current stock: %d)\n",
items[i].itemName, items[i].currentStock);
            break;
        }
    }
```

```
    if (!found)
    {
        printf("Item ID not found\n");
        printf("Enter item name: ");
        scanf(" %[^\n]", items[*itemCount].itemName);
        items[*itemCount].itemID = itemID;
        printf("Enter current stock: ");
        scanf("%d", &items[*itemCount].currentStock);
        printf("Enter reorder level: ");
        scanf("%d", &items[*itemCount].reorderLevel);

        (*itemCount)++;
        printf("New item added successfully\n");
    }
}

// Function to check reorder status
void checkReorderStatus(WarehouseItem items[], int itemCount)
{
    if (itemCount == 0)
    {
        printf("\nNo items in the warehouse\n");
        return;
    }
    printf("\nReorder status for all items:\n");
    for (int i = 0; i < itemCount; i++)
    {
```

```
        if (items[i].currentStock <= items[i].reorderLevel)
        {
            printf("Item: %s (ID: %d) is below reorder level. Current stock: %d  Reorder level: %d\n",
                   items[i].itemName, items[i].itemID, items[i].currentStock, items[i].reorderLevel);
        } else {
            printf("Item: %s (ID: %d) is above reorder level. Current stock: %d  Reorder level: %d\n",
                   items[i].itemName, items[i].itemID, items[i].currentStock, items[i].reorderLevel);
        }
    }
}

// Function to display all stock levels
void displayStock(WarehouseItem items[], int itemCount)
{
    if (itemCount == 0)
    {
        printf("\nNo items to display\n");
        return;
    }
    printf("\nWarehouse Stock Details:\n");
    printf("ID\tItem Name\tCurrent Stock\tReorder Level\n");
    for (int i = 0; i < itemCount; i++)
        printf("%d\t%s\t\t%d\t\t%d\n", items[i].itemID, items[i].itemName,
            items[i].currentStock, items[i].reorderLevel);
}
```

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.

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

typedef struct
{
   char category[50];
   float discountPercentage;
   char validTill[20];
} Discount;

const char* categories[3] = { "Electronics", "Clothing", "Groceries"};

// Function prototypes
void updateDiscount(Discount discounts[], int *discountCount);
void displayDiscounts(Discount discounts[], int discountCount);
```

```
int main()
{
    Discount discounts[3];
    int discountCount = 0;
    int option;
    do
    {
        printf("\nDiscount Management System\n");
        printf("1. Update discount\n");
        printf("2. Display discounts\n");
        printf("3. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);
        switch (option)
        {
            case 1: updateDiscount(discounts, &discountCount);
                    break;
            case 2: displayDiscounts(discounts, discountCount);
                    break;
            case 3: printf("Exit the program\n");
                    break;
            default:printf("Invalid option\n");
        }
    } while (option != 3);
    return 0;
}
```

```
// Function to update discount
void updateDiscount(Discount discounts[], int *discountCount)
{
    if (*discountCount >= 3)
    {
        printf("Maximum number of categories reached\n");
        return;
    }
    char category[50];
    printf("\nEnter the product category: ");
    scanf(" %[^\n]", category);
    int categoryIndex = -1;
    for (int i = 0; i < 3; i++)
    {
        if (strcmp(categories[i], category) == 0)
        {
            categoryIndex = i;
            break;
        }
        if(categoryIndex == -1)
        {
            printf("Invalid category\n");
            return;
        }
    }
    for (int i = 0; i < *discountCount; i++) {
        if (strcmp(discounts[i].category, category) == 0)
```

```
        {
            printf("Discount already set for %s, updating the existing discount\n",
category);
            printf("Enter the new discount percentage: ");
            scanf("%f", &discounts[i].discountPercentage);
            printf("Enter the discount validity date: ");
            scanf(" %[^\n]", discounts[i].validTill);
            return;
        }
    }
    strcpy(discounts[*discountCount].category, category);
    printf("Enter the discount percentage: ");
    scanf("%f", &discounts[*discountCount].discountPercentage);
    printf("Enter the discount validity date: ");
    scanf(" %[^\n]", discounts[*discountCount].validTill);

    (*discountCount)++;
    printf("Discount for %s added successfully\n", category);
}

// Function to display all discounts
void displayDiscounts(Discount discounts[], int discountCount)
{
    if (discountCount == 0)
    {
        printf("\nNo discounts available\n");
        return;
    }
```

```
    printf("\nDiscount Details for All Categories:\n");
    printf("Category\tDiscount%%\tValid Till\n");
    for (int i = 0; i < discountCount; i++)
        printf("%s\t%.2f%%\t\t%s\n", discounts[i].category,
            discounts[i].discountPercentage, discounts[i].validTill);
}
```

Unions

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.

```
#include <stdio.h>

union Data
{
    int i;
    float f;
    char c;
}data;

int main()
{
    data.i = 8;
    printf("Integer: %d\n", data.i);
```

```
    data.f = 1.1;
    printf("Float: %f\n", data.f);
    data.c = 'N';
    printf("Character: %c\n", data.c);
    return 0;
}
```

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.

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

union Student
{
    int rollNumber;
    char name[50];
} s;

int main()
{
    printf("Enter roll number: ");
    scanf("%d", &s.rollNumber);
    printf("Roll Number: %d\n", s.rollNumber);
```

```
    printf("Enter name: ");
    scanf(" %[^\n]", s.name);
    printf("Name: %s\n", s.name);
    return 0;
}
```

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.

```
#include <stdio.h>

union Distance
{
    float kilometers;
    float miles;
} d;

int main()
{
    char units;
    printf("Enter the unit of the distance(k or m): ");
    scanf(" %c", &units);
    if (units == 'k' || units == 'K')
    {
        printf("Enter the distance in kilometers: ");
```

```
        scanf("%f", &d.kilometers);
        printf("Converted distance\n");
        float m = d.kilometers * 0.621371;
        printf("%f kilometers = %f miles\n", d.kilometers, m);
    }
    else if (units == 'm' || units == 'M')
    {
        printf("Enter the distance in miles: ");
        scanf("%f", &d.miles);
        printf("\nConverted distance\n");
        float km = d.miles / 0.621371;
        printf("%f miles = %f kilometers\n", d.miles, km);
    }
    else
    {
        printf("Invalid unit\n");
        return 1;
    }
    return 0;
}
```

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.

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

```
union Shape
{
   float radius;
   struct
   {
      float length;
      float width;
   };
}s1;

int main()
{
   int option;
   printf("Select a shape to calculate the area\n");
   printf("1. Circle\n");
   printf("2. Rectangle\n");
   printf("Enter the option: ");
   scanf("%d", &option);
   switch (option)
   {
      case 1: printf("Enter the radius of the circle: ");
            scanf("%f", &s1.radius);
            float a = 3.14 * s1.radius * s1.radius;
            printf("Area of the circle = %f\n", a);
            break;
      case 2: printf("Enter the length and width of the rectangle: ");
            scanf("%f %f", &s1.length, &s1.width);
```

```
            float a1 = s1.length * s1.width;
            printf("Area of the rectangle = %f\n", a1);
            break;
        default: printf("Invalid option\n");
    }
    return 0;
}
```

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.

```
#include <stdio.h>

union EmployeeData
{
    int employeeID;
    float salary;
}emp;

int main()
{
    char option;
    printf("Select the data to input\n");
    printf("1. Employee ID\n");
    printf("2. Salary\n");
```

```
    printf("Enter the option: ");
    scanf("%d", &option);
    switch(option)
    {
        case 1: printf("Enter employee ID: ");
                scanf("%d", &emp.employeeID);
                printf("Employee ID: %d\n", emp.employeeID);
                break;
        case 2: printf("Enter the salary: ");
                scanf("%f", &emp.salary);
                printf("Salary: %f\n", emp.salary);
                break;
        default:printf("Invalid option\n");
    }
    return 0;
}
```

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.

```
#include <stdio.h>

union Sensor
{
    float temperature;
```

```
    float pressure;
};

int main()
{
    char option;
    printf("Select the type of sensor data to input (t or p): ");
    scanf(" %c", &option);
    if (option == 't' || option == 'T')
    {
        printf("Enter temperature: ");
        scanf("%f", &s.temperature);
        printf("Sensor temperature = %f\n", s.temperature);
    }
    else if (option == 'p' || option == 'P')
    {
        printf("Enter pressure: ");
        scanf("%f", &s.pressure);
        printf("Sensor pressure = %f\n", s.pressure);
    }
    else
        printf("Invalid option\n");
    return 0;
}
```

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.

```
#include <stdio.h>

union Account
{
   int accountNumber;
   float balance;
}a;

int main()
{
   int option;
   printf("Bank Account Information\n");
   printf("1. Enter cccount number\n");
   printf("2. Enter account balance\n");
   printf("Enter the option: ");
   scanf("%d", &option);

   switch (option)
   {
      case 1: printf("Enter cccount number: ");
            scanf("%d", &a.accountNumber);
            printf("Account number: %d\n", a.accountNumber);
            break;
```

```
        case 2: printf("Enter account balance: ");
                scanf("%f", &a.balance);
                printf("Account balance: %f\n", a.balance);
                break;
        default: printf("Invalid option\n");
    }
    return 0;
}
```

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.

```
#include <stdio.h>

union Information
{
    int regNumber;
    float fuelCapacity;
}i;

int main()
{
    int option;
    printf("Vehicle Information\n");
    printf("1. Registration number\n");
```

```
    printf("2. Fuel capacity\n");
    printf("Enter the option: ");
    scanf("%d", &option);
    switch(option)
    {
        case 1: printf("Enter the registration number: ");
            scanf("%d", &i.regNumber);
            printf("Vehicle registration number: %d\n", i.regNumber);
            break;
        case 2: printf("Enter the fuel capacity: ");
            scanf("%f", &i.fuelCapacity);
            printf("Vehicle fuel capacity: %f\n", i.fuelCapacity);
            break;
        default: printf("Invalid option\n");
    }
    return 0;
}
```

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.

```
#include <stdio.h>

union Results
{
```

```
    int marks;
    char grade;
} s;

int main()
{
    int option;
    printf("Exam Results\n");
    printf("1. Marks\n");
    printf("2. Grade\n");
    printf("Enter the option: ");
    scanf("%d", &option);
    switch(option)
    {
        case 1: printf("Enter the marks: ");
                scanf("%d", &s.marks);
                printf("Student's marks: %d\n", s.marks);
                break;
        case 2: printf("Enter the grade: ");
                scanf(" %c", &s.grade);
                printf("Student's grade: %c\n", s.grade);
                break;
        default: printf("Invalid option\n");
    }
    return 0;
}
```

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.

```
#include <stdio.h>

union Currency
{
   float usd;
   float eur;
}c;

int main()
{
   int option;
   const float conversion = 0.85;  // 1 USD = 0.85 EUR
   printf("Currency Conversion\n");
   printf("1. USD\n");
   printf("2. EUR\n");
   printf("Enter the option: ");
   scanf("%d", &option);
   switch(option)
   {
      case 1: printf("Enter the value in USD: ");
            scanf("%f", &c.usd);
            printf("Equivalent in EUR: %.2f\n", c.usd * conversion);
            break;
```

```
        case 2: printf("Enter the value in EUR: ");
                scanf("%f", &c.eur);
                printf("Equivalent in USD: %.2f\n", c.eur / conversion);
                break;
        default: printf("Invalid option\n");
    }
    return 0;
}
```

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.

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

```
struct Aircraft
{
    int aircraftID;
    char model[30];
    int capacity;
    char status[20];
};

// Function prototypes
void addAircraft(struct Aircraft *fleet, int *totalAircraft);
void updateStatus(struct Aircraft *fleet, int totalAircraft);
void displayFleet(struct Aircraft fleet[], int totalAircraft);

int main()
{
    struct Aircraft fleet[100];
    static int totalAircraft = 0;
    int option;
    while(1)
    {
        printf("\nAircraft Fleet Management System\n");
        printf("1. Add new aircraft\n");
        printf("2. Update aircraft status\n");
        printf("3. Display fleet details\n");
        printf("4. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);
```

```
        switch(option)
        {
            case 1: addAircraft(fleet, &totalAircraft);
                    break;
            case 2: updateStatus(fleet, totalAircraft);
                    break;
            case 3: displayFleet(fleet, totalAircraft);
                    break;
            case 4: printf("Exit the system\n");
                    return 0;
            default:printf("Invalid option\n");
        }
    }
    return 0;
}

// Function to add a new aircraft
void addAircraft(struct Aircraft *fleet, int *totalAircraft)
{
    struct Aircraft newAircraft;
    printf("\nEnter aircraft ID: ");
    scanf("%d", &newAircraft.aircraftID);
    printf("Enter aircraft model: ");
    scanf("%s", newAircraft.model);
    printf("Enter aircraft capacity: ");
    scanf("%d", &newAircraft.capacity);
```

```
    //Active, Maintenance, Inactive
    printf("Enter aircraft status: ");
    scanf("%s", newAircraft.status);

    fleet[*totalAircraft] = newAircraft;
    (*totalAircraft)++;

    printf("\nAircraft added successfully\n");
    displayFleet(fleet, *totalAircraft);
}

// Function to update the status of an aircraft
void updateStatus(struct Aircraft *fleet, int totalAircraft)
{
    int id, found = 0;
    printf("\nEnter aircraft ID to update status: ");
    scanf("%d", &id);
    for(int i = 0; i < totalAircraft; i++)
    {
        if(fleet[i].aircraftID == id)
        {
            found = 1;
            printf("Enter new status for aircraft ID %d: ", id);
            scanf("%s", fleet[i].status);
            printf("Status updated successfully\n");
            break;
        }
```

```
    }
    if(!found)
        printf("Aircraft ID %d not found in the fleet\n", id);
    displayFleet(fleet, totalAircraft);
}

// Function to display the details of the fleet
void displayFleet(struct Aircraft fleet[], int totalAircraft)
{
    if(totalAircraft == 0)
    {
        printf("\nNo aircraft in the fleet\n");
        return;
    }
    printf("\nAircraft Fleet 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);
    }
}
```

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.

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

union SatelliteData
{
   char imageData[100];
   struct Telemetry
   {
      float temperature;
      float velocity;
      float altitude;
   }tData;
}sData;

// Function prototypes
void processImageData(union SatelliteData *data);
void processTelemetryData(union SatelliteData *data);
void displayTelemetryData(union SatelliteData *data);
```

```
const float tLimit = -40.0;
const float vLimit = 2000.0;
const float aLimit = 100000.0;

int main()
{
   int option;
   while(1)
   {
      printf("\nSatellite Data Processing System\n");
      printf("1. Process image data\n");
      printf("2. Process telemetry data\n");
      printf("3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch(option)
      {
         case 1: processImageData(&sData);
                 break;
         case 2: processTelemetryData(&sData);
                 break;
         case 3: printf("Exit the system\n");
                 return 0;
         default: printf("Invalid option\n");
      }
   }
   return 0;
```

```
}

// Function to process image data
void processImageData(union SatelliteData *data)
{
    printf("\nEnter the image data: ");
    scanf(" %[^\n]", data->imageData);
    printf("Processed image data: %s\n", data->imageData);
}

// Function to process telemetry data
void processTelemetryData(union SatelliteData *data)
{
    printf("\nEnter the temperature: ");
    scanf("%f", &data->tData.temperature);
    printf("Enter the velocity: ");
    scanf("%f", &data->tData.velocity);
    printf("Enter the altitude: ");
    scanf("%f", &data->tData.altitude);
    displayTelemetryData(data);
}

// Function to display telemetry data and check limits
void displayTelemetryData(union SatelliteData *data)
{
    printf("\nTelemetry Data\n");
    printf("Temperature: %.2f°C\n", data->tData.temperature);
```

```
    printf("Velocity: %.2f km/h\n", data->tData.velocity);
    printf("Altitude: %.2f meters\n", data->tData.altitude);
    if (data->tData.temperature < tLimit)
        printf("Temperature exceeds the limit\n");
    else if (data->tData.velocity > vLimit)
        printf("Velocity exceeds the limit\n");
    else if (data->tData.altitude > aLimit)
        printf("Altitude exceeds the limit\n");
}
```

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.

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

struct Mission
{
```

```
    int missionID;
    char name[50];
    int duration;
    union Payload
    {
        struct Crew
        {
            char name[50];
            int age;
            char role[20];
        }c1;
        struct Cargo
        {
            char description[100];
            float weight;
        }c2;
    }p;
    int isCrew;  // 1 = crew, 0 = cargo
};

// Function prototypes
void addMission(struct Mission *missions, int *totalMissions);
void updateMission(struct Mission *missions, int totalMissions);
void displayMissionSummary(struct Mission mission);
void displayAllMissions(struct Mission *missions, int totalMissions);

static int totalMissions = 0;
```

```
int main()
{
   struct Mission missions[100];
   int option;
   while(1)
   {
      printf("\nMission Control System\n");
      printf("1. Add new mission\n");
      printf("2. Update mission details\n");
      printf("3. Display all mission summaries\n");
      printf("4. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch(option)
      {
         case 1: addMission(missions, &totalMissions);
                 break;
         case 2: updateMission(missions, totalMissions);
                 break;
         case 3: displayAllMissions(missions, totalMissions);
                 break;
         case 4: printf("Exit the system\n");
                 return 0;
         default:printf("Invalid option\n");
      }
   }
   return 0;
```

```
}

// Function to add a new mission
void addMission(struct Mission *missions, int *totalMissions)
{
    struct Mission newMission;
    printf("\nEnter mission ID: ");
    scanf("%d", &newMission.missionID);
    printf("Enter mission name: ");
    scanf(" %[^\n]", newMission.name);
    printf("Enter mission duration: ");
    scanf("%d", &newMission.duration);
    printf("Crew mission(1 = Yes/0 = No): ");
    scanf("%d", &newMission.isCrew);
    if (newMission.isCrew)
    {
        printf("Enter crew member name: ");
        scanf(" %[^\n]", newMission.p.c1.name);
        printf("Enter crew member age: ");
        scanf("%d", &newMission.p.c1.age);
        printf("Enter crew role: ");
        scanf(" %[^\n]", newMission.p.c1.role);
    }
    else
    {
        printf("Enter cargo description: ");
        scanf(" %[^\n]", newMission.p.c2.description);
```

```
        printf("Enter cargo weight: ");
        scanf("%f", &newMission.p.c2.weight);
    }
    missions[*totalMissions] = newMission;
    (*totalMissions)++;
    printf("\nMission added successfully\n");
    displayMissionSummary(newMission);
}

// Function to update mission details
void updateMission(struct Mission *missions, int totalMissions)
{
    int missionID, found = 0;
    printf("\nEnter mission ID to update: ");
    scanf("%d", &missionID);
    for (int i = 0; i < totalMissions; i++)
    {
        if (missions[i].missionID == missionID)
        {
            found = 1;
            printf("\nUpdating details for mission ID: %d\n", missionID);
            printf("Enter new mission name: ");
            scanf(" %[^\n]", missions[i].name);
            printf("Enter new mission duration: ");
            scanf("%d", &missions[i].duration);
            printf("Crew mission(1 = Yes/0 = No): ");
            scanf("%d", &missions[i].isCrew);
```

```
            if (missions[i].isCrew)
            {
                printf("Enter new crew member name: ");
                scanf(" %[^\n]s", missions[i].p.c1.name);
                printf("Enter new crew member age: ");
                scanf("%d", &missions[i].p.c1.age);
                printf("Enter new crew role: ");
                scanf(" %[^\n]s", missions[i].p.c1.role);
            }
            else
            {
                printf("Enter new cargo description: ");
                scanf(" %[^\n]s", missions[i].p.c2.description);
                printf("Enter new cargo weight: ");
                scanf("%f", &missions[i].p.c2.weight);
            }
            printf("Mission updated successfully\n");
            displayMissionSummary(missions[i]);
            break;
        }
    }
    if (!found)
        printf("Mission ID %d not found\n", missionID);
}

// Function to display mission summary
void displayMissionSummary(struct Mission mission)
```

```
{
    printf("\nMission Summary\n");
    printf("Mission ID: %d\n", mission.missionID);
    printf("Mission name: %s\n", mission.name);
    printf("Mission duration: %d days\n", mission.duration);
    if (mission.isCrew)
    {
        printf("Payload: Crew\n");
        printf("Crew member name: %s\n", mission.p.c1.name);
        printf("Crew member age: %d\n", mission.p.c1.age);
        printf("Crew member role: %s\n", mission.p.c1.role);
    }
    else
    {
        printf("Payload: Cargo\n");
        printf("Cargo description: %s\n", mission.p.c2.description);
        printf("Cargo Weight: %f kg\n", mission.p.c2.weight);
    }
}

// Function to display all mission summaries
void displayAllMissions(struct Mission *missions, int totalMissions)
{
    if (totalMissions == 0)
    {
        printf("\nNo missions available\n");
        return;
```

```
    }
    printf("\nAll Mission Summaries:\n");
    for (int i = 0; i < totalMissions; i++) {
        displayMissionSummary(missions[i]);
    }
}
```

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.

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

#define ROUTINE 0
#define EMERGENCY 1

typedef struct
{
```

```
    int logID;
    int aircraftID;
    char date[11];
    union
    {
        char routine[50];
        char emergency[50];
    } maintenanceType;
    int type; // 0 for routine, 1 for emergency
} MaintenanceLog;

//Function prototypes
void addMaintenanceLog(MaintenanceLog *logs, int *count);
void displayMaintenanceLogs(const MaintenanceLog *logs, int count);

int main()
{
    MaintenanceLog logs[100];
    int count = 0;
    int option;
    do
    {
        printf("\nAircraft Maintenance Tracker\n");
        printf("1. Add Maintenance Log\n");
        printf("2. Display logs\n");
        printf("3. Exit\n");
        printf("Enter the option: ");
```

```
        scanf("%d", &option);
        switch (option)
        {
            case 1: addMaintenanceLog(logs, &count);
                    break;
            case 2: displayMaintenanceLogs(logs, count);
                    break;
            case 3: printf("Exit the system\n");
                    break;
            default: printf("Invalid option\n");
        }
    } while (option != 3);
    return 0;
}

void addMaintenanceLog(MaintenanceLog *logs, int *count)
{
    printf("Enter log ID: ");
    scanf("%d", &logs[*count].logID);

    printf("Enter aircraft ID: ");
    scanf("%d", &logs[*count].aircraftID);

    printf("Enter date: ");
    scanf("%s", logs[*count].date);

    printf("Enter Ttype (0 for Routine/1 for Emergency): ");
```

```
    scanf("%d", &logs[*count].type);
    if (logs[*count].type == ROUTINE)
    {
        printf("Enter routine details: ");
        scanf(" %[^\n]", logs[*count].maintenanceType.routine);
    }
    else
    {
        printf("Enter emergency details: ");
        scanf(" %[^\n]", logs[*count].maintenanceType.emergency);
    }
    (*count)++;
}

void displayMaintenanceLogs(const MaintenanceLog *logs, int count)
{
    for (int i = 0; i < count; i++)
    {
        printf("Log ID: %d  Aircraft ID: %d  Date: %s  Type: %s\n",
               logs[i].logID, logs[i].aircraftID, logs[i].date,
               logs[i].type == ROUTINE ? "Routine" : "Emergency");
        if (logs[i].type == ROUTINE)
            printf("Routine details: %s\n", logs[i].maintenanceType.routine);
        else
         printf("Emergency details: %s\n", logs[i].maintenanceType.emergency);
    }
}
```

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.

```
#include <stdio.h>

typedef struct
{
   float position[3];
   float velocity[3];
   union
   {
      char manual[50];
      char automatic[50];
   } mode;
   int i; // 0 for manual, 1 for automatic
} NavigationData;

static int updateCount = 0;
```

```
void updateNavigationData(NavigationData *nData);
void displayNavigationData(const NavigationData nData);


int main()
{
   NavigationData nData;
   int option;
   do
   {
      printf("\nSpacecraft Navigation System\n");
       printf("\n1. Update navigation data\n2. Display navigation data\n3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: updateNavigationData(&nData);
                 break;
         case 2: displayNavigationData(nData);
                 break;
         case 3: printf("Exit the program\n");
                 break;
         default: printf("Invalid option\n");
      }
   } while (option != 3);
   return 0;
}
```

```
void updateNavigationData(NavigationData *nData)
{
   printf("Enter position: ");
    scanf("%f %f %f", &nData->position[0], &nData->position[1], &nData->position[2]);

   printf("Enter velocity: ");
    scanf("%f %f %f", &nData->velocity[0], &nData->velocity[1], &nData->velocity[2]);

   printf("Enter mode (0 for Manual/1 for Automatic): ");
   scanf("%d", &nData->i);
   if (nData->i)
   {
      printf("Enter automatic system details: ");
      scanf(" %[^\n]", nData->mode.automatic);
   }
   else
   {
      printf("Enter manual control details: ");
      scanf(" %[^\n]", nData->mode.manual);
   }
   updateCount++;
}

void displayNavigationData(const NavigationData nData)
{
```

```
        printf("Position: (%.2f, %.2f, %.2f)\n", nData.position[0], nData.position[1], nData.position[2]);
        printf("Velocity: (%.2f, %.2f, %.2f)\n", nData.velocity[0], nData.velocity[1], nData.velocity[2]);
    printf("Mode: %s\n", nData.i ? "Automatic" : "Manual");
    if (nData.i)
        printf("Automatic Details: %s\n", nData.mode.automatic);
    else
        printf("Manual details: %s\n", nData.mode.manual);
    printf("Update: %d\n", updateCount);
}
```

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.

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

```
typedef struct
{
   int simulationID;
   char aircraftModel[50];
   float duration;
   union
   {
      char manual[50];
      char automatic[50];
   } control;
   int i; // 0 for manual, 1 for automated
} Simulation;

void startSimulation(Simulation *s1);
void displaySimulationResults(const Simulation *s1);

int main()
{
   Simulation s[100];
   int count = 0;
   int option;
   do
   {
      printf("\nFlight Simulation Control");
      printf("\n1. Start simulation\n2. Display simulations\n3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
```

```
        switch (option)
        {
            case 1: if (count < 100)
                    {
                        startSimulation(&s[count]);
                        count++;
                    }
                    else
                         printf("Simulation limit reached\n");
                    break;
            case 2: for (int i = 0; i < count; i++)
                        displaySimulationResults(&s[i]);
                    break;
            case 3: printf("Exit\n");
                    break;
            default: printf("Invalid option\n");
        }
    } while (option != 3);
    return 0;
}

void startSimulation(Simulation *s1)
{
    printf("Enter simulation ID: ");
    scanf("%d", &s1->simulationID);

    printf("Enter aircraft model: ");
```

```
    scanf(" %[^\n]", s1->aircraftModel);

    printf("Enter duration: ");
    scanf("%f", &s1->duration);

    printf("Enter type(0 for Manual/1 for Automatic): ");
    scanf("%d", &s1->i);
    if (s1->i)
    {
        printf("Enter automated settings: ");
        scanf(" %[^\n]", s1->control.automatic);
    }
    else
    {
        printf("Enter manual settings: ");
        scanf(" %[^\n]", s1->control.manual);
    }
}

void displaySimulationResults(const Simulation *s1)
{
    printf("Simulation ID: %d\n", s1->simulationID);
    printf("Aircraft model: %s\n", s1->aircraftModel);
    printf("Duration: %.2f hours\n", s1->duration);
    printf("Control: %s\n", s1->i ? "Automatic" : "Manual");
    if (s1->i)
        printf("Automated settings: %s\n", s1->control.automatic);
```

```
    else
        printf("Manual Settings: %s\n", s1->control.manual);
}
```

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.

```
#include <stdio.h>

typedef struct
{
    int testID;
    char componentName[50];
    union
    {
        char physical[50];
        char software[50];
    }testData;
```

```
    int i;      // 0 for physical and 1 for software
}ComponentTest;

static int totalTests = 0;

//Function prototypes
void recordTestResults(ComponentTest *test);
void displaySummaries(const ComponentTest *test);

int main()
{
    ComponentTest tests[100];
    int count = 0, option;
    do
    {
        printf("\nAerospace Component Testing\n");
        printf("1. Record test results\n");
        printf("2. Display summaries\n");
        printf("3. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);
        switch (option)
        {
            case 1: if (count < 100)
                    {
                        recordTestResults(&tests[count]);
                        count++;
```

```
                }
                else
                    printf("Test limit reached\n");
                break;
            case 2: for (int i = 0; i < count; i++)
                    displaySummaries(&tests[i]);
                printf("Total tests conducted: %d\n", totalTests);
                break;
            case 3: printf("Exit the program\n");
                break;
            default: printf("Invalid option\n");
        }
    } while (option != 3);
    return 0;
}

void recordTestResults(ComponentTest *test)
{
    printf("Enter test ID: ");
    scanf("%d", &test->testID);

    printf("Enter component name: ");
    scanf(" %[^\n]", test->componentName);

    printf("Enter test type (0 for physical/1 for software): ");
    scanf("%d", &test->i);
    if (test->i)
```

```
    {
        printf("Enter software test details: ");
        scanf(" %[^\n]", test->testData.software);
    }
    else
    {
        printf("Enter Physical test details: ");
        scanf(" %[^\n]", test->testData.physical);
    }
    totalTests++;
}

void displaySummaries(const ComponentTest *test)
{
    printf("Test ID: %d\n", test->testID);
    printf("Component name: %s\n", test->componentName);
    printf("Test Type: %s\n", test->i ? "Software" : "Physical");
    if (test->i)
        printf("Software Details: %s\n", test->testData.software);
    else
        printf("Physical Details: %s\n", test->testData.physical);
}
```

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.

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

typedef struct
{
   int crewID;
   char name[50];
   char role[20];
   union
   {
      char engineer[50];
      char scientist[50];
   } roleDetails;
   int i; // 1 for engineer and0 for scientist
} CrewMember;

const int maxEngineers = 20;
const int maxScientists = 10;
```

```
void addCrewMember(CrewMember *crew, int *count, int *engineers, int *scientists);
void displayCrewList(const CrewMember *crew, int count);


int main()
{
   CrewMember crew[50];
   int count = 0, engineers = 0, scientists = 0, option;
   do
   {
      printf("\nSpace Station Crew Management\n");
      printf("\n1. Add crew member\n2. Display crew list\n3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: addCrewMember(crew, &count, &engineers, &scientists);
                 break;
         case 2: displayCrewList(crew, count);
                 break;
         case 3: printf("Exit the program\n");
                 break;
         default: printf("Invalid option\n");
      }
   } while (option != 3);
   return 0;
}
```

```
void addCrewMember(CrewMember *crew, int *count, int *engineers, int *scientists)
{
   if (*count >= 50)
   {
      printf("Crew capacity reached\n");
      return;
   }
   printf("Enter crew ID: ");
   scanf("%d", &crew[*count].crewID);

   printf("Enter name: ");
   scanf(" %[^\n]", crew[*count].name);

   printf("Enter role (1 for engineer/0 for scientist): ");
   scanf("%d", &crew[*count].i);
   if (crew[*count].i)
   {
      if (*engineers >= maxEngineers)
      {
         printf("Maximum engineers limit reached\n");
         return;
      }
      printf("Enter engineer specialty: ");
      scanf(" %[^\n]", crew[*count].roleDetails.engineer);
      strcpy(crew[*count].role, "Engineer");
      (*engineers)++;
   }
```

```
    else {
        if (*scientists >= maxScientists)
        {
            printf("Maximum scientists limit reached\n");
            return;
        }
        printf("Enter scientist field: ");
        scanf(" %[^\n]", crew[*count].roleDetails.scientist);
        strcpy(crew[*count].role, "Scientist");
        (*scientists)++;
    }
    (*count)++;
}

void displayCrewList(const CrewMember *crew, int count)
{
    printf("Crew list:\n");
    for (int i = 0; i < count; i++)
    {
        printf("Crew ID: %d  Name: %s  Role: %s\n",
               crew[i].crewID, crew[i].name, crew[i].role);
        if (crew[i].i)
            printf("Specialty: %s\n", crew[i].roleDetails.engineer);
        else
            printf("  Field: %s\n", crew[i].roleDetails.scientist);
    }
}
```

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.

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

typedef struct
{
   int experimentID;
   char description[100];
   union
   {
      float numerical[10];
      char qualitative[100];
   } data;
   int i; // 1 for numerical and 0 for qualitative
} ResearchData;
```

```
static int analysesConducted = 0;

void analyzeData(ResearchData *data);
void generateReport(const ResearchData *data);

int main()
{
   ResearchData experiments[50];
   int count = 0;
   int option;
   do
   {
      printf("\nAerospace Research Data Analysis\n");
      printf("\n1. Analyze data\n2. Generate reports\n3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: if (count < 50)
                 {
                    analyzeData(&experiments[count]);
                    count++;
                 }
                 else
                 printf("Experiment limit reached\n");
                 break;
         case 2: for (int i = 0; i < count; i++)
```

```
                generateReport(&experiments[i]);
                break;
            case 3: printf("Exit the program\n");
                break;
            default: printf("Invalid option\n");
        }
    } while (option != 3);
    return 0;
}

void analyzeData(ResearchData *data)
{
    printf("Enter experiment ID: ");
    scanf("%d", &data->experimentID);

    printf("Enter description: ");
    scanf(" %[^\n]", data->description);

    printf("Enter data type (1 for numerical/0 for qualitative): ");
    scanf("%d", &data->i);
    if (data->i)
    {
        printf("Enter 5 numerical data points: ");
        for (int i = 0; i < 5; i++)
            scanf("%f", &data->data.numerical[i]);
    }
    else
```

```
    {
        printf("Enter qualitative data: ");
        scanf(" %[^\n]", data->data.qualitative);
    }
    analysesConducted++;
}

void generateReport(const ResearchData *data)
{
    printf("Experiment ID: %d\n", data->experimentID);
    printf("Description: %s\n", data->description);
    if (data->i)
    {
        printf("Numerical data: ");
        for (int i = 0; i < 5; i++)
            printf("%f ", data->data.numerical[i]);
        printf("\n");
    }
    else
        printf("Qualitative data: %s\n", data->data.qualitative);
    printf("Total analyses conducted: %d\n", analysesConducted);
}
```

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.

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

typedef struct
{
   int launchID;
   char rocketName[50];
   char date[11];
   union
   {
      char scheduled[50];
      char completed[50];
   } status;
   int i; // 0 for scheduled and 1 for completed
} Launch;

void scheduleLaunch(Launch *launches, int *count);
void displayLaunches(const Launch *launches, int count);
```

```
int main()
{
   Launch launches[50];
   int count = 0;
   int option;
   do
   {
      printf("\nRocket Launch Scheduler");
      printf("\n1. Schedule launch\n2. Display launches\n3. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
      {
         case 1: scheduleLaunch(launches, &count);
                 break;
         case 2: displayLaunches(launches, count);
                 break;
         case 3: printf("Exit the program\n");
                 break;
         default: printf("Invalid option\n");
      }
   } while (option != 3);
   return 0;
}

void scheduleLaunch(Launch *launches, int *count)
{
```

```
    printf("Enter launch ID: ");
    scanf("%d", &launches[*count].launchID);

    printf("Enter rocket name: ");
    scanf(" %[^\n]", launches[*count].rocketName);

    printf("Enter date: ");
    scanf("%s", launches[*count].date);

    printf("Enter status (0 for scheduled/1 for completed): ");
    scanf("%d", &launches[*count].i);

    if (launches[*count].i)
    {
        printf("Enter completed details: ");
        scanf(" %[^\n]", launches[*count].status.completed);
    }
    else
    {
        printf("Enter scheduled details: ");
        scanf(" %[^\n]", launches[*count].status.scheduled);
    }
    (*count)++;
}

void displayLaunches(const Launch *launches, int count)
{
```

```
    for (int i = 0; i < count; i++)
    {
        printf("Launch ID: %d  Rocket name: %s  Date: %s  Status: %s\n",
               launches[i].launchID, launches[i].rocketName, launches[i].date,
               launches[i].i ? "Completed" : "Scheduled");
        if (launches[i].i)
            printf("Completed details: %s\n", launches[i].status.completed);
        else
            printf("Scheduled details: %s\n", launches[i].status.scheduled);
    }
}
```